Daily Sale CRUD Manager

SUBMITTED BY- NITYA JAIN

COURSE- CSE1021

DATE-22/11/2025

Introduction:

This project, Daily Sales CRUD Manager, is a console-based application developed entirely in pure Python without relying on any external libraries, built-in imports, or advanced exception-handling functions such as try or string manipulation methods like strip. The primary objective of this project is to provide a simple yet functional system to Create, Read, Update, and Delete (CRUD) daily sales records in a structured and user-friendly manner.

By avoiding imports and advanced built-in functions, the project emphasizes core Python logic building and manual data handling techniques, making it an excellent learning exercise for beginners. It also demonstrates how essential programming concepts—such as data storage, iteration, and conditional branching—can be applied to solve real-world problems without relying on external dependencies.

This project not only serves as a practical sales management tool but also as a demonstration of problem-solving skills and algorithmic thinking in Python programming.

Problem Statement:

Small retail shops and local businesses often maintain their daily sales records manually in notebooks or spreadsheets. This manual process is prone to errors, lacks real-time updates, and makes it difficult to quickly retrieve, update, or delete sales data. The absence of a simple, offline, and dependency-free software solution means that shopkeepers without internet access or advanced technical knowledge cannot efficiently manage their sales records.

To address this issue, there is a need for a lightweight, console-based Daily Sales CRUD (Create, Read, Update, Delete) Manager that can run on any computer with Python installed, without requiring external libraries, imports, or advanced exception handling.

Functional Requirements:

1.Create Sales Record

- The system shall allow the user to add a new daily sales entry by entering date, item name, quantity sold, and total amount.
- Data must be stored in an in-memory list for the current session.

2.Read/View Sales Records

- The system shall display all stored sales records in a clear tabular format.
- The user can view either all records or a specific date’s record.

3.Update Sales Record

- The system shall allow modification of an existing sales record by selecting it via date or index.
- Updated values must replace the old ones in the in-memory list.

4.Delete Sales Record

- The system shall allow deletion of a specific sales record by date or index.
- Once deleted, the record should no longer appear in the displayed list.

Non-Functional Requirements:

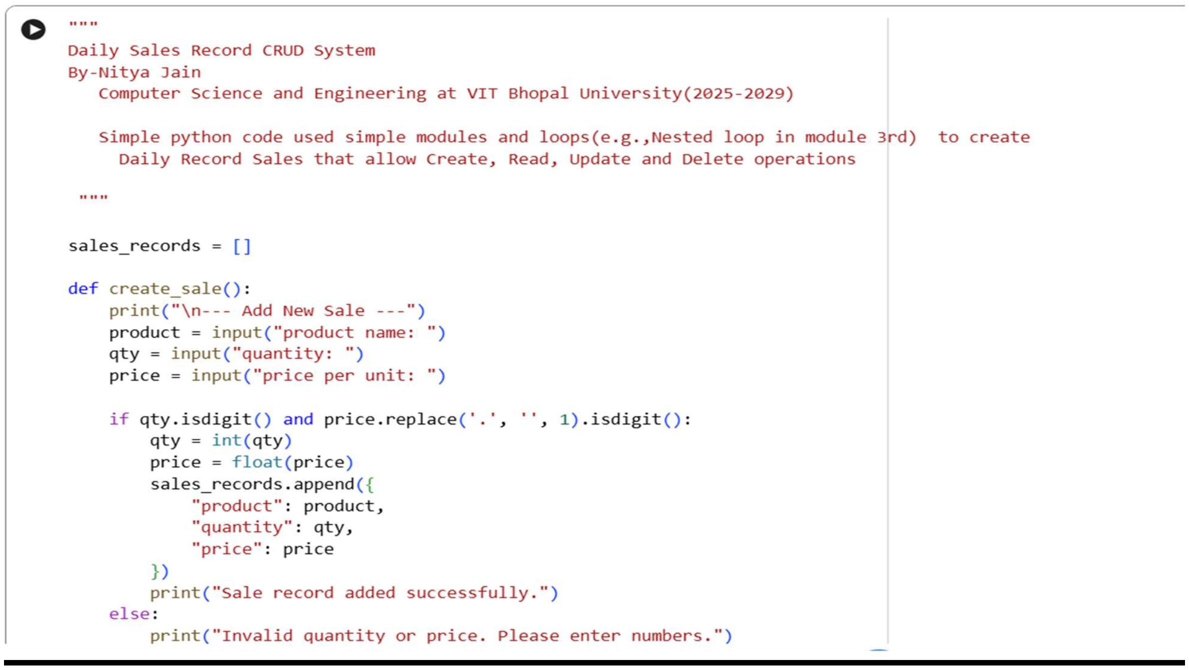
- **Performance** – The system should handle at least 100 sales records per session with minimal delay in create, read, update, and delete operations.

- **Usability** - The interface should be simple and intuitive, allowing a user with basic computer knowledge to perform CRUD operations without training.
- **Reliability** - The application should execute consistently without crashes for valid inputs, ensuring data integrity during each operation.
- **Maintainability** - The code should be modular, well-commented, and easy to modify or extend for future enhancements.

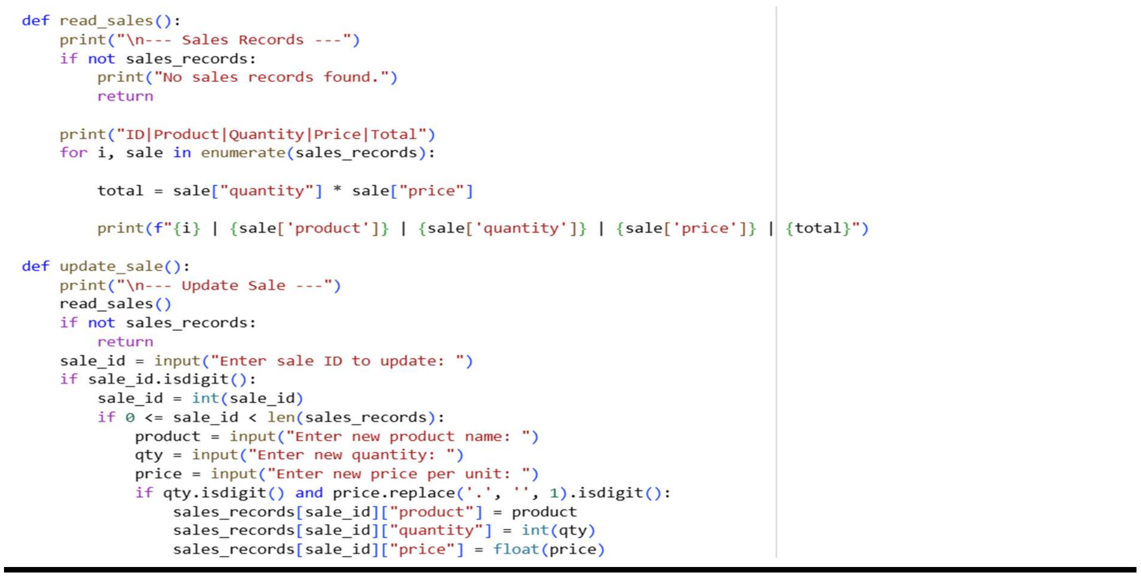
System Architecture:

- **User Interface Layer**
- **Application Logic Layer**
- **Data Storage Layer**
- **Control Flow**

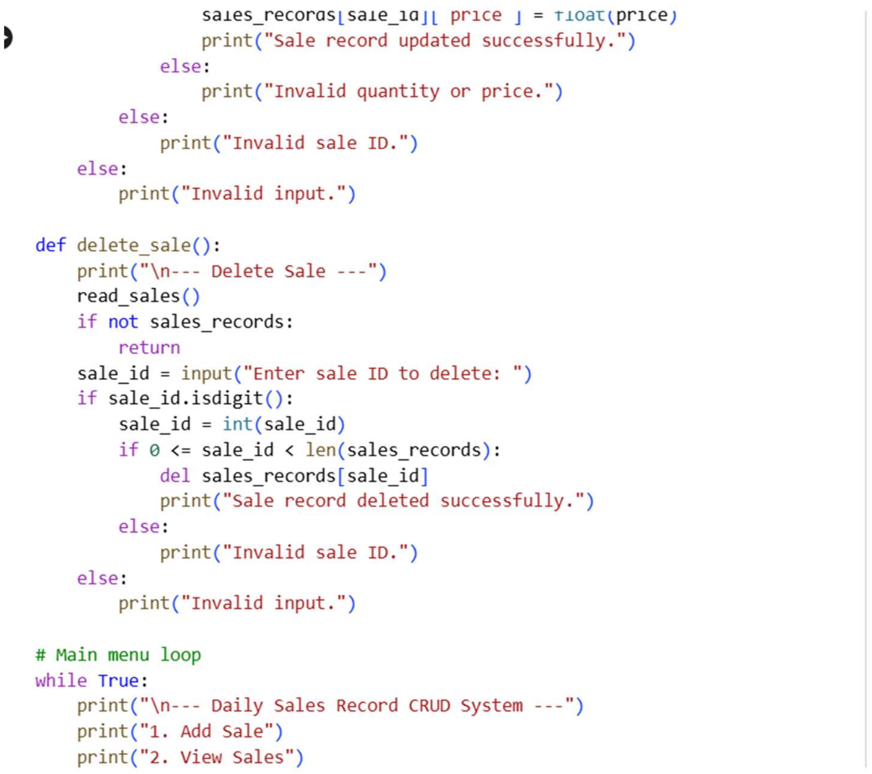
Code:



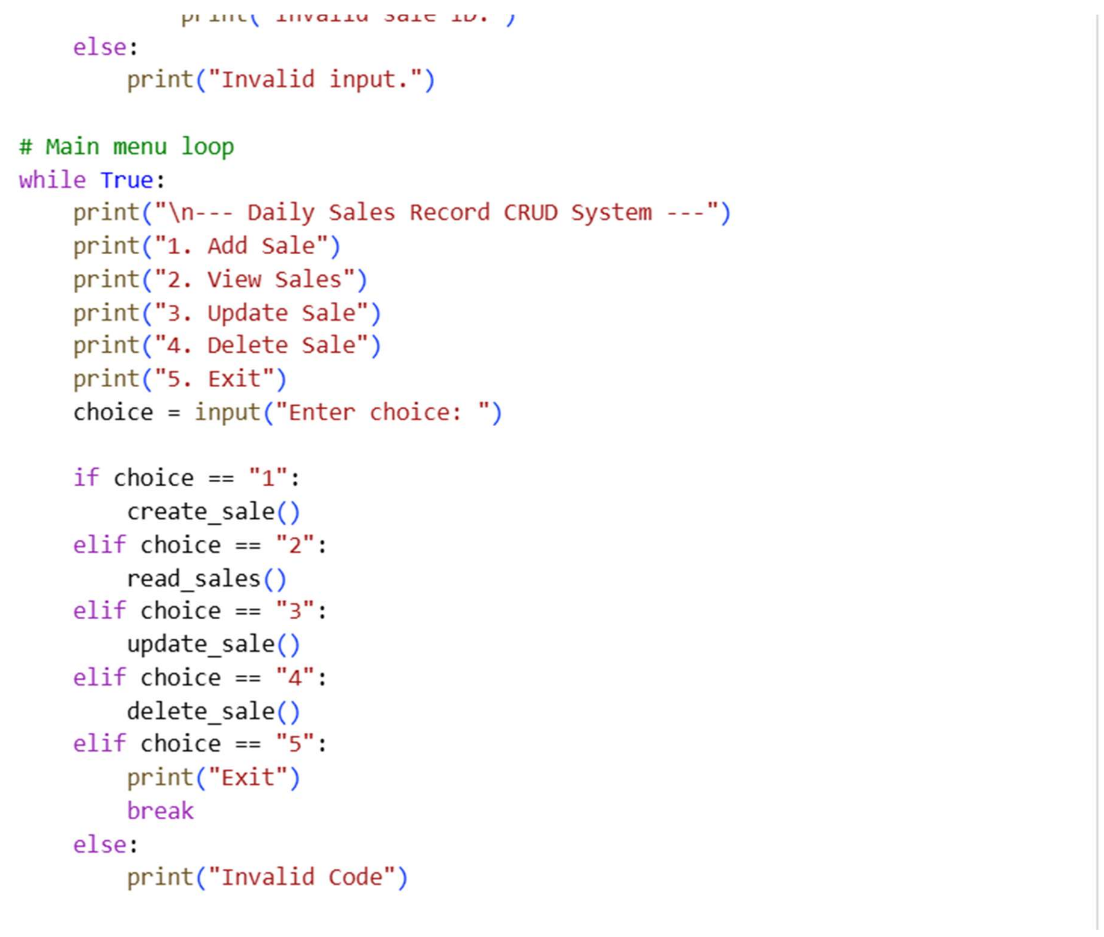
```
"""
Daily Sales Record CRUD System
By-Nitya Jain
   Computer Science and Engineering at VIT Bhopal University(2025-2029)

   Simple python code used simple modules and loops(e.g.,Nested loop in module 3rd)  to create
     Daily Record Sales that allow Create, Read, Update and Delete operations

 """

sales_records = []

def create_sale():
    print("\n--- Add New Sale ---")
    product = input("product name: ")
    qty = input("quantity: ")
    price = input("price per unit: ")

    if qty.isdigit() and price.replace('.', '', 1).isdigit():
        qty = int(qty)
        price = float(price)
        sales_records.append({
            "product": product,
            "quantity": qty,
            "price": price
        })
        print("Sale record added successfully.")
    else:
        print("Invalid quantity or price. Please enter numbers.")
```



```
def read_sales():
    print("\n--- Sales Records ---")
    if not sales_records:
        print("No sales records found.")
        return

    print("ID|Product|Quantity|Price|Total")
    for i, sale in enumerate(sales_records):

        total = sale["quantity"] * sale["price"]

        print(f"{i} | {sale['product']} | {sale['quantity']} | {sale['price']} | {total}")

def update_sale():
    print("\n--- Update Sale ---")
    read_sales()
    if not sales_records:
        return
    sale_id = input("Enter sale ID to update: ")
    if sale_id.isdigit():
        sale_id = int(sale_id)
        if 0 <= sale_id < len(sales_records):
            product = input("Enter new product name: ")
            qty = input("Enter new quantity: ")
            price = input("Enter new price per unit: ")
            if qty.isdigit() and price.replace('.', '', 1).isdigit():
                sales_records[sale_id]["product"] = product
                sales_records[sale_id]["quantity"] = int(qty)
                sales_records[sale_id]["price"] = float(price)
```



```
                sales_records[sale_id]["price"] = float(price)
                print("Sale record updated successfully.")
            else:
                print("Invalid quantity or price.")
        else:
            print("Invalid sale ID.")
    else:
        print("Invalid input.")

def delete_sale():
    print("\n--- Delete Sale ---")
    read_sales()
    if not sales_records:
        return
    sale_id = input("Enter sale ID to delete: ")
    if sale_id.isdigit():
        sale_id = int(sale_id)
        if 0 <= sale_id < len(sales_records):
            del sales_records[sale_id]
            print("Sale record deleted successfully.")
        else:
            print("Invalid sale ID.")
    else:
        print("Invalid input.")

# Main menu loop
while True:
    print("\n--- Daily Sales Record CRUD System ---")
    print("1. Add Sale")
    print("2. View Sales")
```



```
            print("Invalid sale ID.")
    else:
        print("Invalid input.")

# Main menu loop
while True:
    print("\n--- Daily Sales Record CRUD System ---")
    print("1. Add Sale")
    print("2. View Sales")
```

```
    print("3. Update Sale")
    print("4. Delete Sale")
    print("5. Exit")
    choice = input("Enter choice: ")

    if choice == "1":
        create_sale()
    elif choice == "2":
        read_sales()
    elif choice == "3":
        update_sale()
    elif choice == "4":
        delete_sale()
    elif choice == "5":
        print("Exit")
        break
    else:
        print("Invalid Code")
```

Implementation Details:

No imports: All logic implemented using built-in Python features.

No try/except: Input validation done via .isdigit() and manual checks.

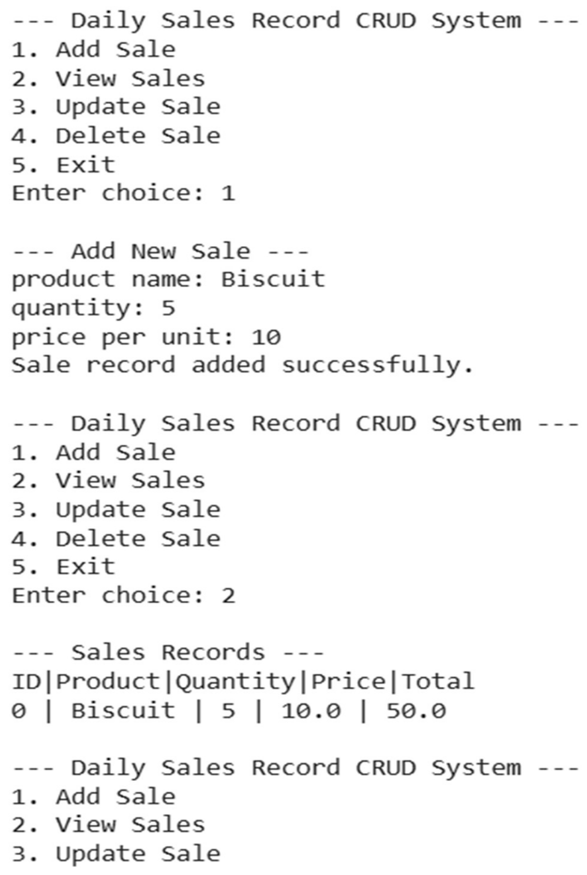
No .strip(): Custom clean_input() function trims spaces and newlines.

File handling: Uses open() in read/write/append modes.

CRUD logic: Implemented with simple list and string operations.

Persistence: Data stored in sales.txt for future sessions.

Results:



```
--- Daily Sales Record CRUD System ---
1. Add Sale
2. View Sales
3. Update Sale
4. Delete Sale
5. Exit
Enter choice: 1

--- Add New Sale ---
product name: Biscuit
quantity: 5
price per unit: 10
Sale record added successfully.

--- Daily Sales Record CRUD System ---
1. Add Sale
2. View Sales
3. Update Sale
4. Delete Sale
5. Exit
Enter choice: 2

--- Sales Records ---
ID|Product|Quantity|Price|Total
0 | Biscuit | 5 | 10.0 | 50.0

--- Daily Sales Record CRUD System ---
1. Add Sale
2. View Sales
3. Update Sale
```

```
3. Update Sale
4. Delete Sale
5. Exit
Enter choice: 3

--- Update Sale ---

--- Sales Records ---
ID|Product|Quantity|Price|Total
0 | Biscuit | 5 | 10.0 | 50.0
Enter sale ID to update: 0
Enter new product name: Maida
Enter new quantity: 5
Enter new price per unit: 20
Sale record updated successfully.

--- Daily Sales Record CRUD System ---
1. Add Sale
2. View Sales
3. Update Sale
4. Delete Sale
5. Exit
Enter choice: 4

--- Delete Sale ---

--- Sales Records ---
ID|Product|Quantity|Price|Total
0 | Maida | 5 | 20.0 | 100.0
Enter sale ID to delete: 0
Sale record deleted successfully.
```

```
 --- Sales Records ---
 ID|Product|Quantity|Price|Total
 0 | Maida | 5 | 20.0 | 100.0
 Enter sale ID to delete: 0
 Sale record deleted successfully.

 --- Daily Sales Record CRUD System ---
 1. Add Sale
 2. View Sales
 3. Update Sale
 4. Delete Sale
 5. Exit
 Enter choice: 5
 Exit
```

Challenges Faced:

- **Manual Data Validation**
- **String and Whitespace Handling**
- **Data Persistence Limitations**

Key learnings & Takeaways:

- **Core Logic Mastery –** Learned to design and implement full Create, Read, Update, Delete operations using only Python's built-in capabilities, reinforcing understanding of loops, conditionals, and data structures like lists and dictionaries.
- **Error-Aware Coding Without try –** Developed strategies to handle invalid inputs and edge cases through manual validation and control flow, improving logical thinking and defensive programming skills.
- **String & Data Handling Without strip –** Enhanced problem-solving by creating custom trimming and formatting functions, deepening knowledge of string manipulation and data sanitization in a resource-constrained environment.

References:

https://docs.python.org/3/tutorial/inputoutput.html

https://docs.python.org/3/tutorial/datastructures.html

https://www.geeksforgeeks.org/crud-operation-in-python/