

FINAL PROJECT	
Progress Report 5	
Course Code: CPE201L	Program: Computer Engineering
Course Title: Data Structure and Algorithm	Date Performed: 10/11/2025
Section: CPE 2-A	Date Submitted: 10/11/2025
Name(s): Disomnong, Jalilah Nerio, Hannah Grace Palmes, Lewis Clark	Instructor: Engr. Maria Rizette H. Sayo
1. Objectives:	
<p>The key objectives of this project are:</p> <ul style="list-style-type: none"> • To implement a system using Linked list and Stack • To develop a system that records and tracks inventory items • To record and manage daily transactions 	
2. Intended Learning Outcomes (ILOs):	
<p>By the end of this project, participants will:</p> <ul style="list-style-type: none"> • Apply data structure concepts • Design and implement a functional inventory management system 	
3. Discussion:	

This canteen inventory system is designed to efficiently manage stocks, products, and transaction history. The goal is to provide a solution that tracks available products, monitors stock levels, and records transaction histories for easy reference. The system will allow users to add new products, update stock quantities, and remove discontinued items, while maintaining a log of past transactions for future review or adjustments.

To handle these tasks efficiently, we will use a **stack** and a **linked list**. The **stack** will store the transaction history using the **Last-In-First-Out (LIFO)** principle. This means that the most recent transaction is always easily accessible, making it ideal for reviewing or correcting the latest sales. Whenever a product is sold, the transaction is pushed onto the stack. This structure allows quick access to the most recent transactions, which is helpful for managing returns or refunds.

The **linked list** will be used for managing the inventory of products. Each node in the linked list will store details about a product, including its name, price, unit and stock quantity. The linked list is ideal for this purpose because it allows for easy insertion and removal of products. As products are added or discontinued, the linked list can be updated dynamically without requiring significant restructuring, making it flexible and efficient for handling an ever-changing product catalog.

4. Materials:

Hardware:

- Computer
- Software:
- Python Programming Language
- PyCharm
- GitHub

5. Procedure:

In this project, we will enhance the existing canteen inventory system by integrating **stack** and **linked list** structures. First, we will implement the **linked list** to manage the product inventory. Each product will be stored in a node with details such as **product name**, **price**, **unit**, and **stock quantity**. We'll modify existing inventory management functions to support dynamic updates (e.g., adding, updating, or removing products).

Next, we'll add a **stack** to record transaction history. The stack will store transaction details (e.g., product name, quantity, and sale price) using the **Last-In-First-Out (LIFO)** principle. When a sale

occurs, we will push the transaction onto the stack, enabling quick access to recent transactions for review, corrections, or refunds.

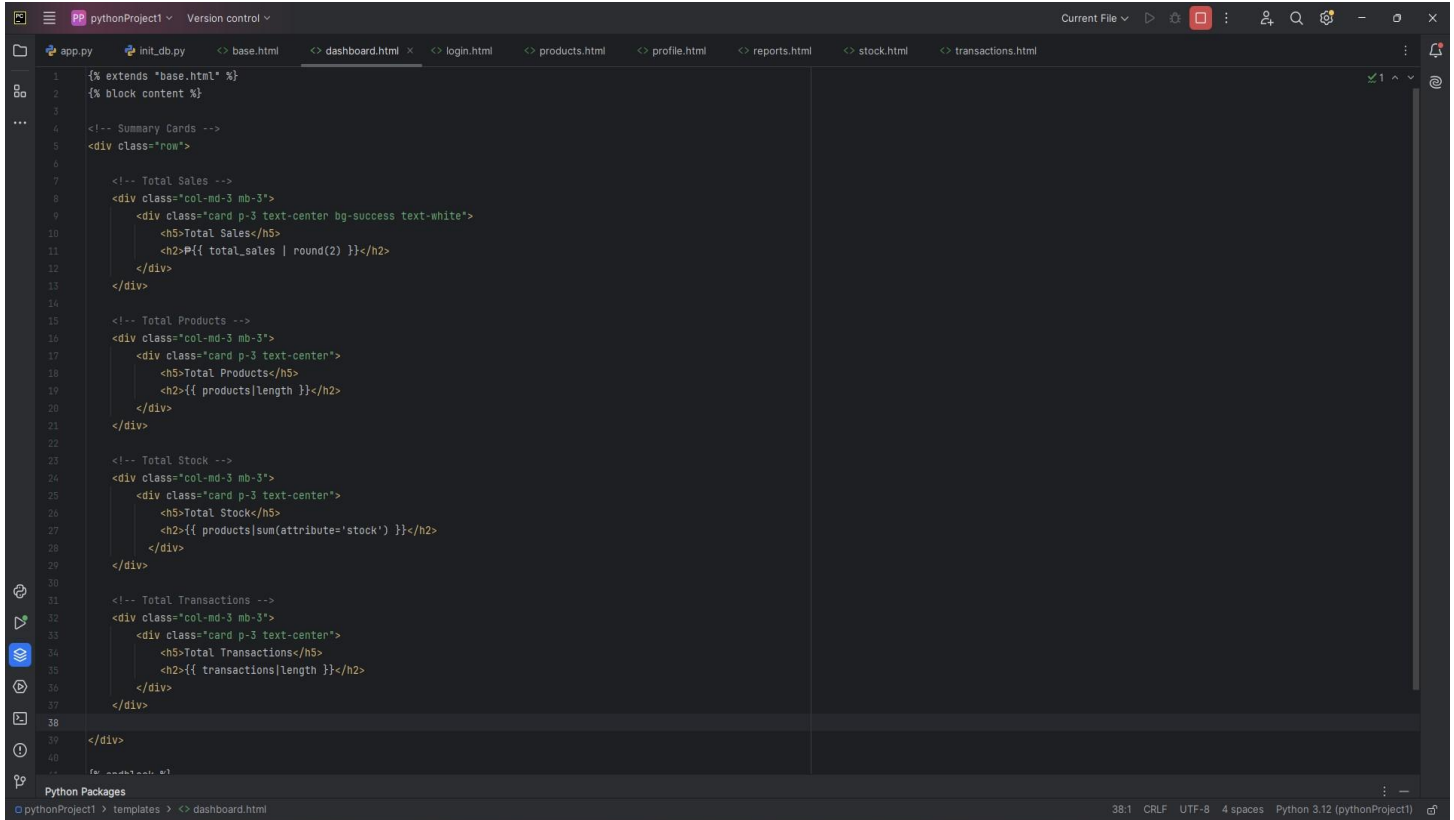
The integration will be straightforward: **transactions will update the stack**, while **product information will be managed via the linked list**. We will update relevant functions to ensure that both data structures work seamlessly with the existing system.

Below is the code that use the data structures (linked list and stack) to manage products and transactions in an inventory system.

```
app.py x <> base.html <> login.html <> products.html <> reports.html <> pro
8
9 class ProductNode: 1 usage
10     def __init__(self, product):
11         self.product = product
12         self.next = None
13
14 class ProductLinkedList: 1 usage
15     def __init__(self):
16         self.head = None
17
18     def append(self, product): 1 usage
19         node = ProductNode(product)
20         if not self.head:
21             self.head = node
22         else:
23             current = self.head
24             while current.next:
25                 current = current.next
26             current.next = node
27
28     def find(self, product_id): 2 usages
29         current = self.head
30         while current:
31             if current.product["id"] == product_id:
32                 return current.product
33             current = current.next
34         return None
35
36     def find_by_name(self, name): 1 usage
37         current = self.head
38         while current:
39             if current.product["name"] == name:
40                 return current.product
41             current = current.next
42         return None
43
44     def delete(self, product_id): 1 usage
45         current = self.head
46         prev = None
47         while current:
48             if current.product["id"] == product_id:
49                 if prev:
```

```
app.py x <> base.html <> login.html <> products.html <> reports.html <> profile
14 class ProductLinkedList: 1 usage
36     def find_by_name(self, name): 1 usage
40         return current.product
41         current = current.next
42     return None
43
44     def delete(self, product_id): 1 usage
45         current = self.head
46         prev = None
47         while current:
48             if current.product["id"] == product_id:
49                 if prev:
50                     prev.next = current.next
51                 else:
52                     self.head = current.next
53                 return True
54             prev = current
55             current = current.next
56         return False
57
58     def to_list(self): 5 usages
59         result = []
60         current = self.head
61         while current:
62             result.append(current.product)
63             current = current.next
64         return result
65
66
67 class TransactionStack: 1 usage
68     def __init__(self):
69         self.stack = []
70
71     def push(self, transaction): 1 usage
72         self.stack.append(transaction)
73
74     def get_all(self): 3 usages
75         return list(reversed(self.stack)) # newest first
76
77     def total_sales(self): 1 usage
78         return sum(t["total"] for t in self.stack)
79
```

HTML (not fully finished)



The screenshot shows a code editor window for a project named 'pythonProject1'. The file explorer on the left lists several files: 'app.py', 'init_db.py', 'base.html', 'dashboard.html' (selected), 'login.html', 'products.html', 'profile.html', 'reports.html', 'stock.html', and 'transactions.html'. The main editor area displays the content of 'dashboard.html', which is a Jinja2 template. The code includes a block for 'Summary Cards' containing four summary cards: 'Total Sales', 'Total Products', 'Total Stock', and 'Total Transactions'. Each card displays a total value and a count. The status bar at the bottom indicates the file is at line 38, column 1, with CRLF line endings, UTF-8 encoding, 4 spaces for indentation, and Python 3.12 interpreter.

```
1 {% extends "base.html" %}
2 {% block content %}
3
4 <!-- Summary Cards -->
5 <div class="row">
6
7 <!-- Total Sales -->
8 <div class="col-md-3 mb-3">
9     <div class="card p-3 text-center bg-success text-white">
10         <h5>Total Sales</h5>
11         <h2>{{ total_sales | round(2) }}</h2>
12     </div>
13 </div>
14
15 <!-- Total Products -->
16 <div class="col-md-3 mb-3">
17     <div class="card p-3 text-center">
18         <h5>Total Products</h5>
19         <h2>{{ products|length }}</h2>
20     </div>
21 </div>
22
23 <!-- Total Stock -->
24 <div class="col-md-3 mb-3">
25     <div class="card p-3 text-center">
26         <h5>Total Stock</h5>
27         <h2>{{ products|sum(attribute='stock') }}</h2>
28     </div>
29 </div>
30
31 <!-- Total Transactions -->
32 <div class="col-md-3 mb-3">
33     <div class="card p-3 text-center">
34         <h5>Total Transactions</h5>
35         <h2>{{ transactions|length }}</h2>
36     </div>
37 </div>
38
39 </div>
40
41 <!-- Footer -->
42 <div class="row">
43     <div class="col">
44         <small>© 2024 All rights reserved.</small>
45     </div>
46 </div>
```

6. Output:

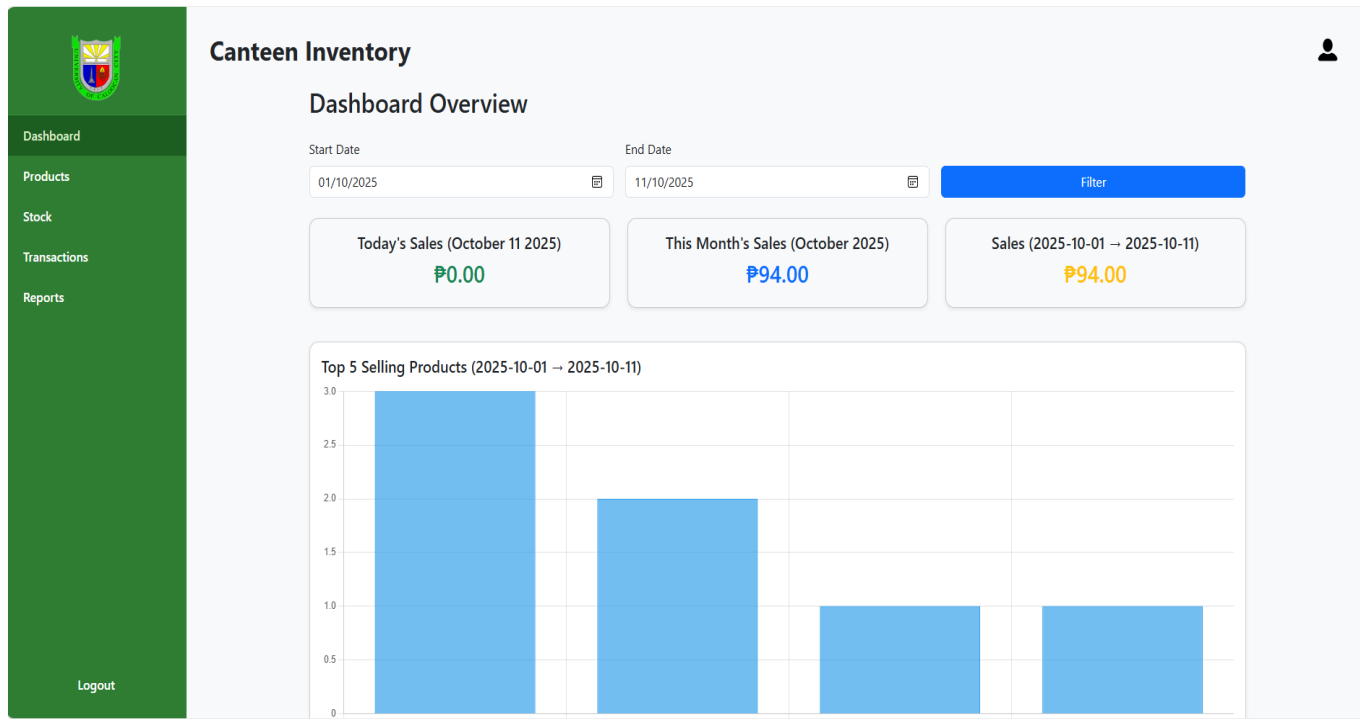
Canteen Inventory						admin
Products						Add Product
ID	Name	Stock	Unit	Price	Actions	
1	hotdog	18	1kilo	50.0	Edit Delete	
2	pencil	100	1 piece	10.0	Edit Delete	

This is the look of our product section without implementing yet the stack and linked list structure

Canteen Inventory						admin
Transactions						Add Transaction
Search by product or payment mode...						
Date	Product	Quantity	Price	Total	Payment Mode	Reference #
2025-09-13 09:04:05	hotdog	2	50.0	100.0	Cash	-
2025-09-13 09:04:18	pencil	10	10.0	100.0	Cash	-

This is the look of our transactions section without implementing yet the stack and linked list structure

Update:



In the recent update of our canteen inventory system, we added a calendar and a graph to the dashboard to improve its functionality and make it easier to manage. The calendar shows the total sales for each specific date, allowing us to quickly see how much was sold on any given day. Meanwhile, the graph provides a visual presentation of the most sold products, helping us identify top-selling items and manage inventory more effectively.

7. Conclusion:

8. References: