DATA STRUCTURES

PROJECT REPORT

SHOPPING CART

TEAM MEMBERS:

1.Kamalika G – 2023115081

2.Praveena R – 2023115071

3. Babi – 2023115067

INTRODUCTION:

The Electronic Shopping Cart System is designed to simulate the management and operation of an online shopping platform. The system categorizes products into predefined categories such as Electronics, Groceries, Books, Beauty Products, and Clothing. Using data structures like linked lists, stacks, queues, binary search trees (BST), and graphs, the program handles product addition, searching, updating, deletion, and tracking low stock items. This system demonstrates how data structures can efficiently manage and organize large datasets.

OVERVIEW:

The system categorizes products into five distinct groups, each with unique product IDs and operations. Key functionalities include adding products, searching by ID, updating stock, deleting products with tracking, and analyzing stock levels. Deleted products are maintained in a stack, ensuring retrieval of deletion history. The program is menu-driven, offering user-friendly interaction.

DATA STRUCTURE USED:

**LINKED LIST**:

Maintains a dynamic list of products in each category. Products are linked sequentially, enabling efficient traversal and management.

**STACK**:

Tracks deleted products for retrieval and review in Last-In-First-Out (LIFO) order.

**QUEUE**:

Reserved for future functionalities like customer order management.

**BINARY SEARCH TREE (BST)**:

Can be integrated for quick product searches by ID.

**GRAPH**:

Represents category relationships, providing a foundation for advanced analytics.

#### **FUNCTION OPERATIONS IN DETAIL:**

##### **1.ADDING A PRODUCT**

**Process**:

* 1. Accepts user input for product name, price, and quantity.
  2. Generates a unique product ID using category-specific counters.
  3. Creates a new linked list node for the product and appends it to the category list.

**Data Structure**: Linked List.

##### **2.DELETING A PRODUCT**

**Process**:

* 1. Locates the product by ID in the linked list.
  2. Removes the node from the list.
  3. Pushes the removed product onto a stack for tracking.

**Data Structure**: Linked List and Stack.

##### **3.DISPLAYING PRODUCTS**

**Process**:

* 1. Iterates through the linked list of a specific category.
  2. Prints product details such as ID, name, price, and quantity.

**Data Structure**: Linked List.

##### **4.UPDATING STOCK**

**Process**:

* 1. Searches the linked list for the product by ID.
  2. Updates the quantity of the product.

**Data Structure**: Linked List.

##### **5.FINDING LOWEST STOCK**

**Process**:

* 1. Traverses the linked list of a category.
  2. Tracks the product with the lowest stock.

**Data Structure**: Linked List.

##### **6.DISPLAYING DELETED PRODUCTS**

**Process**:

* 1. Traverses the stack to display products in LIFO order.

**Data Structure**: Stack.

**FLOW OF EXECUTION**

1. The program begins with a category selection menu.
2. Based on user input, the relevant category menu is displayed.
3. Users can perform operations such as adding, deleting, updating, or searching for products.
4. Each operation interacts with the corresponding data structure (e.g., linked list for storage, stack for deletion history).
5. After completing the operation, the program returns to the category menu.

**CHALLENGES AND CONSIDERATIONS**

* **Efficient Memory Management**: Dynamic allocation for linked list nodes and stack elements required careful handling to avoid memory leaks.
* **Data Integrity**: Ensuring product IDs are unique and consistent across operations.
* **Category-Specific Operations**: Handling operations specific to categories without hardcoding repeated logic.
* **Edge Cases**:
  + Empty linked lists or stacks during traversal.
  + Invalid product IDs or quantities during updates and searches.

#### **CONCLUSION**

This project demonstrates the integration of various data structures to efficiently manage and operate an electronic shopping cart system. The program is robust, menu-driven, and showcases practical applications of linked lists, stacks, and category management in real-world scenarios.

**Future Enhancements**

* **Customer Order Management**:
  + Use queues to manage customer orders in First-In-First-Out (FIFO) order.
* **Search Optimization**:
  + Implement a binary search tree (BST) for faster product searches by ID.
* **Graph-Based Recommendations**:
  + Use a graph to recommend products based on category relationships.
* **Persistent Storage**:
  + Integrate file handling or a database for persistent product storage.
* **Enhanced User Interface**:
  + Develop a graphical user interface (GUI) for improved usability.

Top of Form

Bottom of Form