**NAME; NANA KWAME OWUSU ANTWI**

**ID; 10908853**

**MAIL;** [**nkoantwi@st.ug.edu.gh**](mailto:nkoantwi@st.ug.edu.gh)

**Github profile link;**

***INTRODUCTION:***

In the process of monitoring inventory, vendors, bills, as well supplied goods, store owners need to use a simple inventory management system. This management system should be very easy to use. Thus, it should have a good UI.

This document simply describes the simple inventory control system for a shop that can perform several tasks including;

* adding items to inventory
* viewing suppliers
* items in inventory
* viewing bills
* issuing items to customers
* viewing items that have been issued

This system is implemented by the Java programming language, as well as a Java FX which for the UI.

*FUNCTIONALITY:*

Using information such the product name, product ID, description, price(cost), quantity, the inventory management system enables the store owner to add new items to the inventory.

To individually identify each product, a unique ID is given to it. That is the PRODUCT ID.

The system makes all these visible to the shop owner. This will enable him perform his management tasks easily.

The list can be sorted according to several factors, including the product name, price, and quantity. The system also provides alerts when the stock of a specific product goes below a predetermined threshold.

*System design*

The main components of this system are;

* *User interface*

This user interface was developed using Java Fx. Thus, it has a very friendly interface to

Make navigation and browsing through the system very easy. The store managers would be able to perform their management tasks with ease.

* Data structures and algorithms

In a way to efficiently manage the inventory, the system implements data structures based on the given categories.

* Stacks - category 1 to 4
* Queues - Category 5 to 7
* Searching and sorting algorithms - 6 to 11
* For viewing goods and bills, iterators, recursion,
* Stack and queue implementations of the list, or other techniques based on the requirements of the application.
* Maps to simply keep track of product sales, where the product code is the key and
* the number of sales is the value. This makes retrieval of product sales easy.
* *Database*

*Additional Requirements*

* HashMap – This is to store information about the vendors, where the vendor’s name

is the key and the vendor details are the values

* To maintain the balance between too high and too low stock, the store owner needs to regularly monitor inventory levels and adjust orders accordingly. This can be done by setting minimum and maximum stock levels for each item, and placing orders to replenish stock when it falls below the minimum level.
* Stacks: Stacks are used when adding and removing items in categories 1 to 4. In this system, stacks are used to keep track of the inventory levels of items in the
* Beverages, Bread/Bakery, Canned/Jarred Goods, and Dairy categories. When a new item is added to the inventory, it is pushed onto the top of the stack. Whenan item is sold or removed from the inventory, it is popped off the top of the
* Queues: Queues are used when adding and removing items in categories 5 to 7.

In this system, queues are used to keep track of the inventory levels of items in the

* Dry/Baking Goods, Frozen Foods, and Meat categories. When a new item is added to the inventory, it is added to the back of the queue. When an item is sold or removed from the inventory, it is removed from the front of the queue.
* Lists: Lists are used when adding and removing items in categories 8 to 11. In this system, lists are used to keep track of the inventory levels of items in the Produce, Cleaners, Paper Goods, and Personal Care categories. When a new item is added to the inventory, it is added to the end of the list. When an item is sold or removed from the inventory, it is removed from the list.
* Maps: Maps are used to keep track of product sales. Each time a product is sold, its product code is entered into a sales file. In this system, a map is used to storethe product code and the number of times it has been sold. This allows the store owner to generate reports on product sales and track which products are most popular.

Generally, the system uses different data structures to work efficiently. Thus, it allows for efficient and organized management of inventory, sales, and vendor information. It enables the store owner to easily add and remove items from the inventory, track sales.

Based on the codes, the performance analysis of the algorithm using Big O Notation is below:

* It also gets the top index and fills the queue with null values if it is empty. Time Complexity: O(n), where is the number of items in the table.
* If the queue is full, it expands the queue size by doubling it and inserts the item.
* If the queue is empty, it displays an error message.
* Time Complexity: O(n), where n is the number of items that need to be inserted.
* Time Complexity: O(n), where n is the number of rows that need to be shifted.
* Enqueue: The enqueue method inserts an item into the queue by updating the row with the next available index.
* Time Complexity: O(1) for most cases. However, it can be O(n) in the worst case when the queue is full, and the size has to be doubled.
* Dequeue: The dequeue method removes an item from the front of the queue by updating the row with null values and shifting the rows to fill the gap.
* Time Complexity: O(n), where n is the number of items in the queue because it has to
* shift all the rows to fill the gap.
* Expand Queue Size: The expand “QueueSize” method doubles the queue's size by inserting null values into the table.
* Shift Rows: The shift Rows method shifts all the rows in the table to fill the gap leftby the removed item.

The overall time complexity of this algorithm is O(n) in the worst-case scenario when the queue is full, and the size must be doubled or when dequeuing, and all the rows need to be shifted. In most cases, the time complexity is O(1) for enqueue and O(n) for

dequeue. However, the algorithm's performance may vary depending on the number of

items in the queue and the size of the table.

CONCLUSION:

Based on the given requirements, a provision store inventory management system has been developed. The system provides features for adding goods, viewing vendors, viewing bills, issued goods, and viewing issued goods. In addition, other functionality has been included, such as maintaining the balance between too high and too low stock, generating reports, implementing searching and sorting algorithms, and analyzing the algorithm's performance using Big O Notation. The data structure implementation has been done using stacks, queues, and lists, depending on the categories of items. Maps and HashMaps have been used to keep track of product sales and vendor information, respectively. The system's interface has been designed to be user-friendly and intuitive, enabling store owners to manage their inventory efficiently. Overall, the provision store -inventory management system is an effective tool for store owners to manage their inventory and track sales. The system's various functionalities and data structure implementations enable efficient stock management and sales tracking. The system's analysis using Big O Notation has helped identify any performance bottlenecks and optimize the system's performance.