FIFO Compliant Inventory Management System

Chen Gu, Sikang Hu, Weihan Liu, Hao Yuan, Jiahui Zhang

Northeastern University, Boston, MA, USA

Abstract

This report discusses the result of the work done in the development of a "FIFO Compliant Inventory Management System." It aims to develop a system which simplifies the inventory management procedures of small retail business and provides analysis to support business decisions.

This system includes a user-friendly web interface that allows the user to input information about sales and orders, filter and view sales and orders using user-defined criteria, and tracks real-time inventory status. Furthermore, this system provides an analysis of turnover rate and inventory age, in order to send out a reminder at the most efficient reorder date for each item, as well as identifies the overstocked goods. Finally, in order to provide an analysis of profit and COGS (Cost of Goods Sold), the database design of this project builds on an SKU (Stock Keeping Unit) system to implement the FIFO (First-In-First-Out) assumption of cost flow.

Introduction

According to the National Small Business Profile [1], there are 30.7 million small businesses in the United States, and 59.9 million people are employed by small business Although each small company alone has far less impact than the big companies, together the entire small business sector takes up 47.3% of employment of the private workforce and becomes the largest employer in the United States. Also, the small business sector grows faster than other types of business. The small business created 1.8 million net jobs in 2016, which is about seven times the net jobs created by big companies [2]. With the trend of decentralization, small businesses will continually grow in the foreseeable future.

However, compared to big firms, small business owners have much less technical support. According to an online survey of 1467 self-identified small business owners from across the United States, more small businesses are being left behind when it comes to the digital world.

29% of the respondents said they would not use digital techniques. Also, 22% of the respondents mentioned the lack of time and resources is the biggest challenge [3].

Before using the relational model based inventory management systems, managing inventory involved a large number of manually conducted tasks in the file-based system. Even for small businesses, these tasks are labor-intensive and time-consuming. Small businesses usually relied heavily on manually logging the storage into spreadsheet files, or even more traditional way, into paper-based files. These traditional storage management methods do not have an easy and direct way to provide an aggregated status report of the inventory, and more importantly, the manual steps often led to errors in the system. Furthermore, without the inventory management methods, important business decisions, such as price adjustment and deciding the timing of purchases, largely relied on the business owner's expertise and experiences, but these decision methods are neither testable nor reproducible.

Therefore, the goal of this project is to develop an Inventory Management Application to support the small-scale retail business by providing reliable suggestions for efficient utilization of their capital and storage space. In order to be friendly to those who are not familiar with digital products, this application is designed to maintain a manageable size without redundant functionalities, but at the same time, provides core features and an easy-to-use interface that matter the most to small retail owners.

Features of this application can be categorized into two major parts. Firstly, this application implemented basic inventory control, including tracking all orders and sales, filtering orders or sales using user-defined criteria, viewing vendor information, handling returned items, as well as showing real-time inventory status for the entire store, or for a user-specified item/item category.

Secondly, this application implemented the computation and analysis of profit, cost of goods sold (COGS), turnover rate, and inventory age. The profit and COGS analysis is based on the FIFO (first in first out) cost flow assumptions, which follow the logic that a retailer sells the oldest stock available for each purchase. The turnover rate analysis helps the owner to measure the performance of a product and is able to remind the owner of the most efficient date to reorder a certain product based on its turnover ratio. The inventory age analysis contributes to identifying obsolescent goods. These analyses help the small business owner to minimize their capital and storage costs and to have more efficient usage of current resources they have.

The following report includes three sections: In the first section, we will illustrate the design of this application, including the system architecture, database design, and how we design our user interface. In the second section, we mentioned what data we used to test our functionalities. Last, we show how we use the data to demonstrate the functionalities, what are the results of these analyses, as well as the conclusion.

System Architecture

Our system consists of a separate frontend and backend. For the frontend, we constructed a web user interface using the BootStrap template, which will display the information about inventory status in the form of charts and tables. Users can interact with the website, and provide commands to the system. These commands will then be transformed into HTTP requests and sent to the server in the backend.

In the backend, the business logic implementation will process the commands and communicate with the database management system through API. After retrieving and processing the data, our server will make a response to the request and send back the desired JSON data. Once the callback function receives the data, the website will be updated, revealing the latest HTML and CSS data on users' browsers.

The overall architecture can be sketched in figure 1:

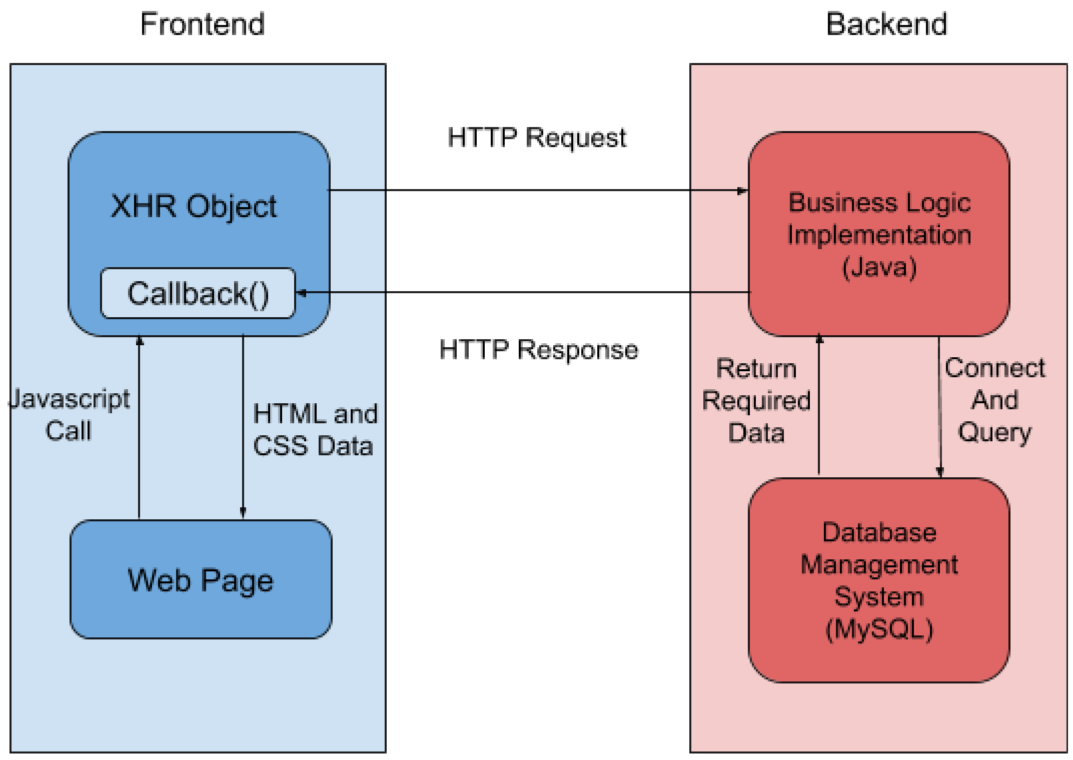


Figure 1. System Architecture Diagram

In the model of this application, we utilized the Data Access Object (DAO) and Data Transfer Object (DTO) design pattern to transfer data between the database and business logic implementation. The DTO carries data between processes and can transfer more data within each call and reduces the number of calls to the database. In our application, each entity of the database is mapped as a corresponding DTO, which allows us the transfer of the entity as a whole between different processes using only one single method call. It is worth mentioning that some DTOs do not have the corresponding table in our database, but are still created for convenience purposes (e.g., Inventory Status).
The DAO is used to directly access and manipulate the data from the database using JDBC API. This design is good for separating the lower-level data access API (JDBC) from high-level business services. The DAOs majorly include the CRUD(Create, Get, Update and Delete) methods of the database, as well as the calls of the stored procedures in the database. This design successfully decouples the data storage from the rest of the application.

Database Design

There have been several overhauls on the logical design during the implementation of a conceptual model. As more features came into the picture, additional procedures and tables were added to the database. One major issue that has changed the entirety of the database design is the First-In-First-Out (FIFO) inventory valuation in cost accounting.

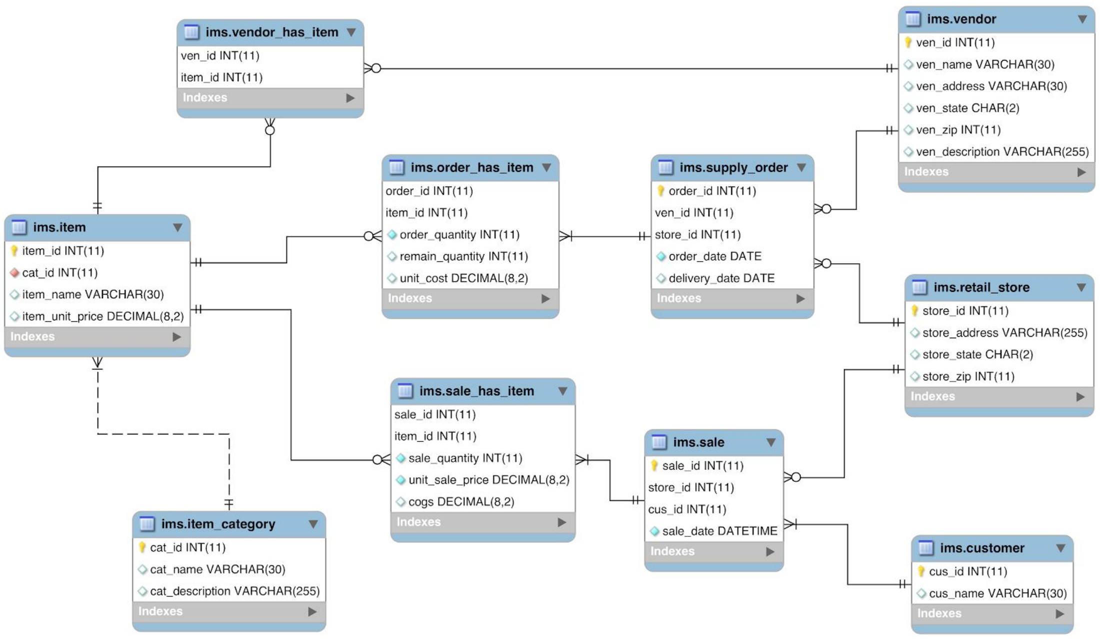


Figure 2. Old Logical Design without SKU

While not mentioned in the project status, inventory valuation is actually a critical problem for database design and implementation as all potential users would have eventually face the problem of cost accounting when conducting every business operation. Although all three prevalent valuation methods (First-In-First-Out, Last-In-First-Out, and Weighted Average) are allowed by the U.S. Generally Accepted Accounting Principles for financial reporting purposes, only FIFO is allowed by the International Financial Reporting Standards. Thus the logistics of this database will be FIFO compliant and would sell the oldest inventory items first and restock returned items would be assigned with the most recent cost of goods when the sale was made, no matter the actual sale/return sequences at the store.

To realize key functionalities (new sales and returned sales) in FIFO order, the old design (Figure 2) was insufficient and flawed as the database must be able to track not only the sequence the items entered the inventory but also the sequence the cost of goods sold assigned to sales. Therefore splitting Sale and Supply_Order from the item table is inevitable and one plausible way to trace down the sequences is to add a Stock Keeping Unit (SKU) in between (Figure 3).

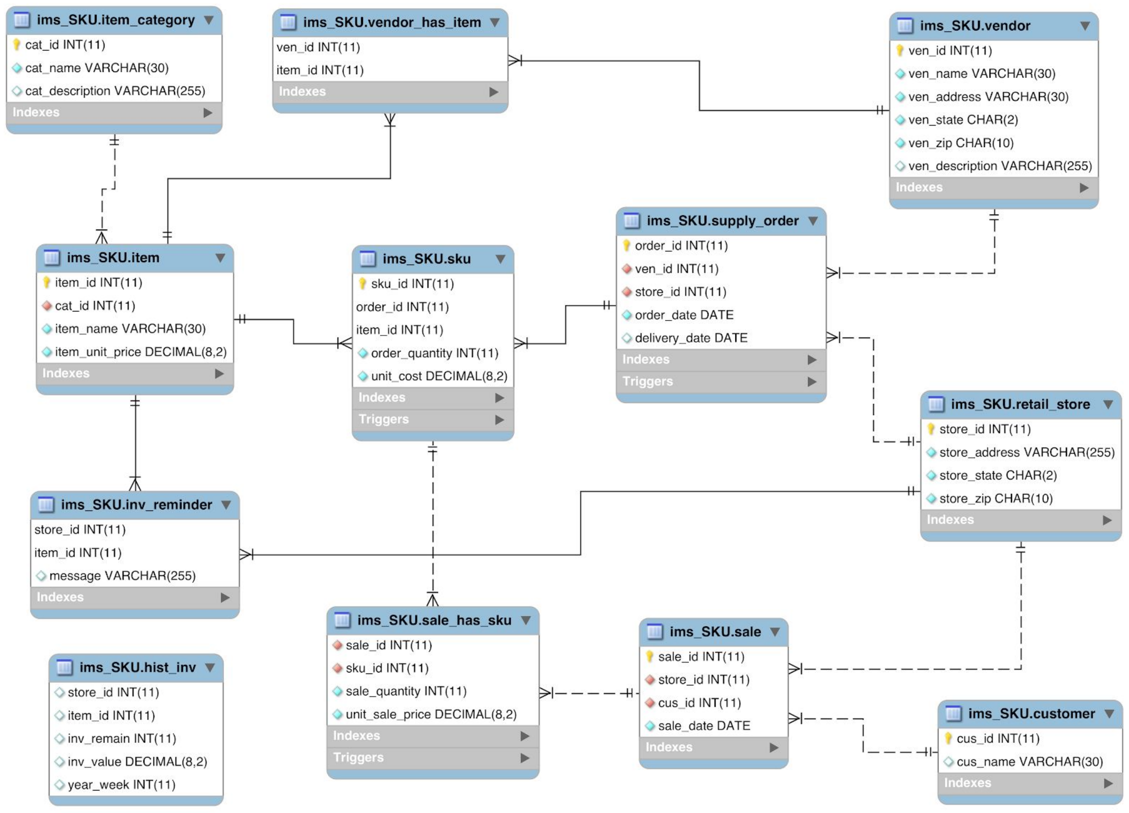


Figure 3. Final Logical Design with SKU

The SKU table acts as an intermediary agent between Supply_Order and Sale, allowing Sale table to backtrack the cost of goods sold based on order delivery time. Once again providing independence to attributes and tables, the new design not only avoided the potential chasm trap between Order_Has_Item and Sale_Has_Item presented in the old design but also removed the remain_quantity and cogged attributes whose calculations were extremely convoluted by different procedures and triggers on other tables. All inventory and sales status can thus be dynamically selected by joining different tables with SKU in stored procedures.

With the new design, when inserting a new sale and its corresponding items, the database and its stored procedures will automatically find out all SKU by item_id in FIFO order based on delivery_date and produce a list of SKU with the quantity remaining by joining with Sale_Has_SKU. The procedure will then decide which SKU stack in FIFO order the sale should take based on sale_quantity and assign sale_id with sku_id accordingly.

When dealing with returned items, the old sale entries will not be modified. Instead, a new Sale_Has_SKU will be added with the same sale_id and sku_id (looked up through item_id) but with negative sale_quantity representing the quantity of the item(s) returned based on reversed FIFO stack order as long as the returned item actually is a part of that sale and the returned quantity does not exceed the sale_quantity. The insertion procedure will strictly follow the FIFO rule of sales where the cost of goods returned must be the most recent costs assigned to inventory when the sale was made. And since all inventory and sale status are calculated dynamically by procedures, no other attributes or tables would need to be modified in order to reflect this sale return, providing extra flexibilities for both retail users and database administrators.

The new design also provides additional tables such as Hist_Inv ('historical inventory'), which is separated from the database for isolation control, and Inv_Reminder ('Inventory Reminder') to support both stock data analysis and replenishment reminders for different items and different stores. The inventory status will be copied to the Hist_Inv table every Sunday by SQL event and item replenishment reminders will be automatically created/updated/deleted by SQL triggers on Supply_Order and Sale.

Data Acquisition

All data are manually made up and entered into the database. There is no strict rule regarding data generation as long as the input follows simple business logic. For example, sales quantity must be smaller or equal to inventory remains, and return quantity cannot be greater than what's actually sold.

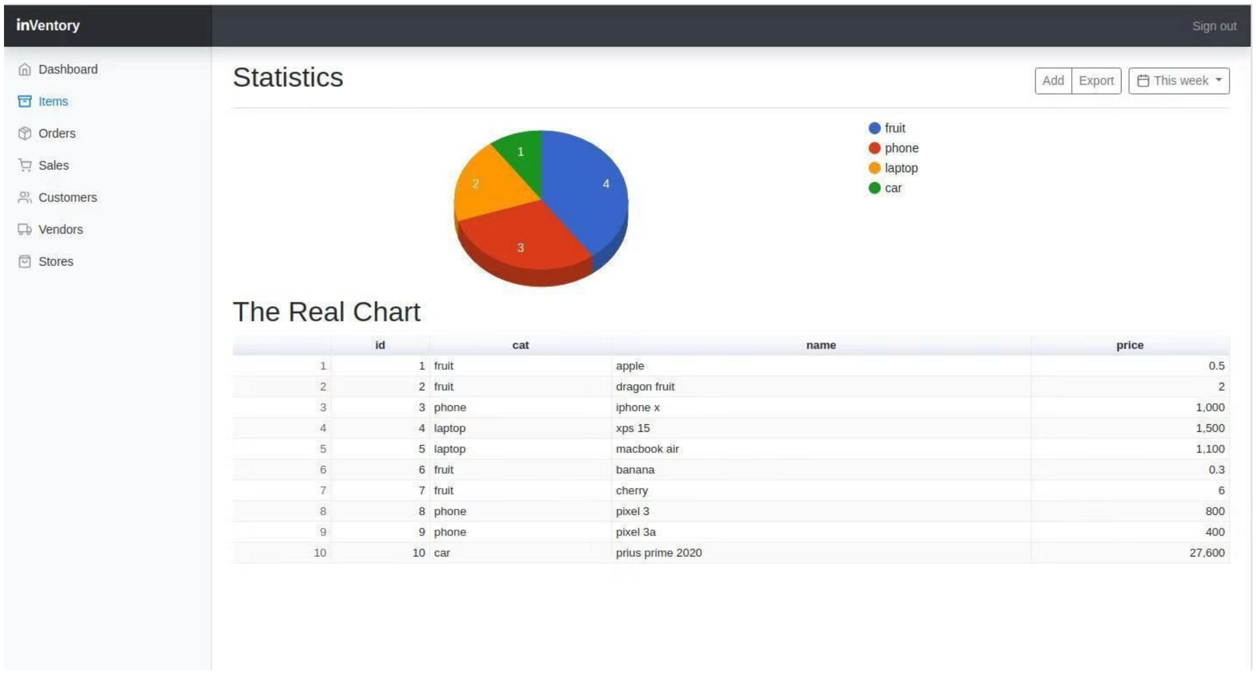
There are, however, a few assumptions for business logistics:

- All Supply_Order created by users will have a NULL delivery_date at first.
- All Items in the same Supply_Order will arrive together.
- Items will not suffer from degradation once become part of the inventory.

User Interface

We aim to create a web application that allows the users to use our application iteratively. The web interface includes features that allow the users to insert information of stores, vendors, item categories, and items, as well as view the information of these entities in table form. The user can customize the filters that applied to the table, which provides a more organized view of the information.

The web interface also has features to show inventory status, profits, inventory age, and turnover rate in the form of charts. These features require the user to enter the information on all sales and orders. The interface also provides iterative ways to insert this information.



	id	cat	name	price
1	1	fruit	apple	0.5
2	2	fruit	dragon fruit	2
3	3	phone	iphone x	1,000
4	4	laptop	xps 15	1,500
5	5	laptop	macbook air	1,100
6	6	fruit	banana	0.3
7	7	fruit	cherry	6
8	8	phone	pixel 3	800
9	9	phone	pixel 3a	400
10	10	car	prius prime 2020	27,600

Figure 4. Demo of User Interface

While our application supports all the functions and features mentioned above, we did not add all of them to the UI. Due to our limited time to create a web interface, the current user interface only support basic viewing functions.

Analysis and Results

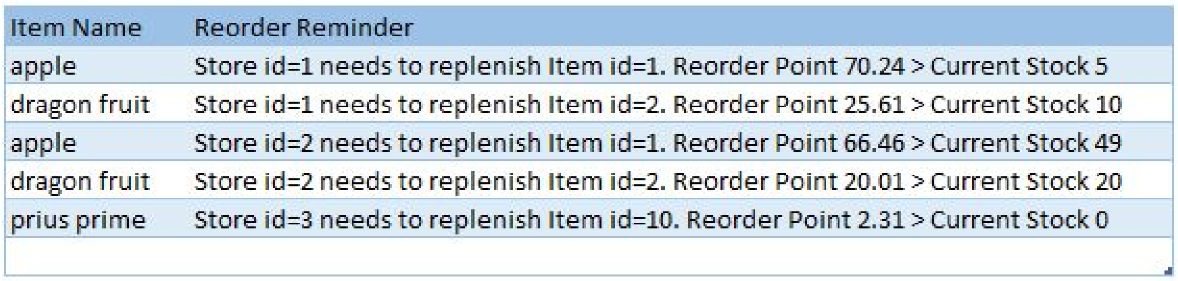
Our application will offer four features on top of the general inventory operation controls and order trackings. Those four features will help small businesses effectively measure their capital investment on each product, estimate the profitability of product lines, trace the detailed inventory situation for each product as well as analyze the operational efficiency in a given period. Specifically, our four uses cases can be categorized as:

Reorder Reminder:

Every purchase from a customer increases our revenue but also lowers our stock. Reordering too early makes a company spend more money needed in stock while reordering too late will disappoint our customers. Reorder reminders will help to reorder our products at the right time.

A reorder reminder will appear in the reminder table to alert the inventory manager what products in certain stores need to be replenished, i.e. when the inventory level of certain products is below the corresponding reorder point. Reorder point consists of two parts, which are lead time demand and safety stock. Lead time demand is supposed to satisfy the forecasted daily sales during lead time, and the daily sale is calculated dynamically in our database, which equals the actual daily sales in the past week. Safety stock takes into account the maximum daily sale and maximum lead time in history. Again, this helps the company to fulfill the orders from customers.

As shown in figure 5, after some purchases from customers, the stock of apples at store 1 is below the reorder point and a new order with the supplier needs to be replaced. After the orders with the supplier are placed, those reminders will be deleted. Keeping a record of a single store might be easy, but when there are multiple stores within the company, the possibility of missing orders from customers become likely. As we can see, apples in store two also need to be replenished. Our application monitors the inventory in real-time and reorders point is updated dynamically on a weekly basis, this will cut down the cost of excess stock and ensure there is enough stock for our customer even unexpected situation occurs.



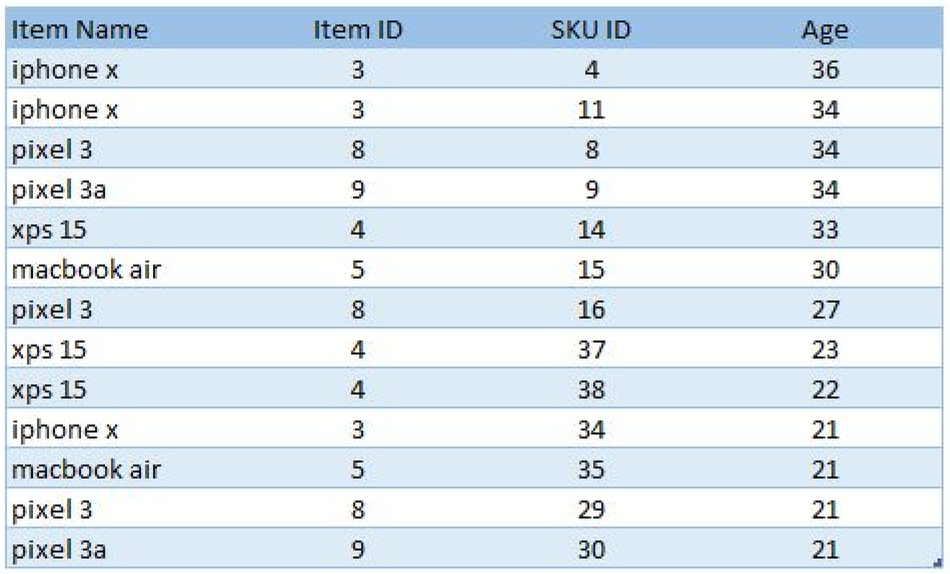
Item Name	Reorder Reminder
apple	Store id=1 needs to replenish Item id=1. Reorder Point 70.24 > Current Stock 5
dragon fruit	Store id=1 needs to replenish Item id=2. Reorder Point 25.61 > Current Stock 10
apple	Store id=2 needs to replenish Item id=1. Reorder Point 66.46 > Current Stock 49
dragon fruit	Store id=2 needs to replenish Item id=2. Reorder Point 20.01 > Current Stock 20
prius prime	Store id=3 needs to replenish Item id=10. Reorder Point 2.31 > Current Stock 0

Figure 5. Reorder Reminders Table

Inventory Age:

Inventories that stayed in stock for too long have a lot of disadvantages. These inventories are likely to become obsolete, increase holding cost, and take up money that could be invested in some other products that are more profitable. It also may lead to great write-off in a single financial period if these risks were not identified periodically. Therefore it is essential to have a process to identify the risks and reduce those losses to a company.

Our application can identify those stocks with high age and then the inventory manager can take further action to reduce or avoid the loss, for example, lowering the price. The item with the biggest age in our example is an iPhone x, which has stayed in stock for 36 days. Those electrical products tend to deteriorate as they continue to stay in the warehouse, and at the same time incurring more holding costs. Our application can help to spot such items to provide faster turns and reduce unnecessary costs.



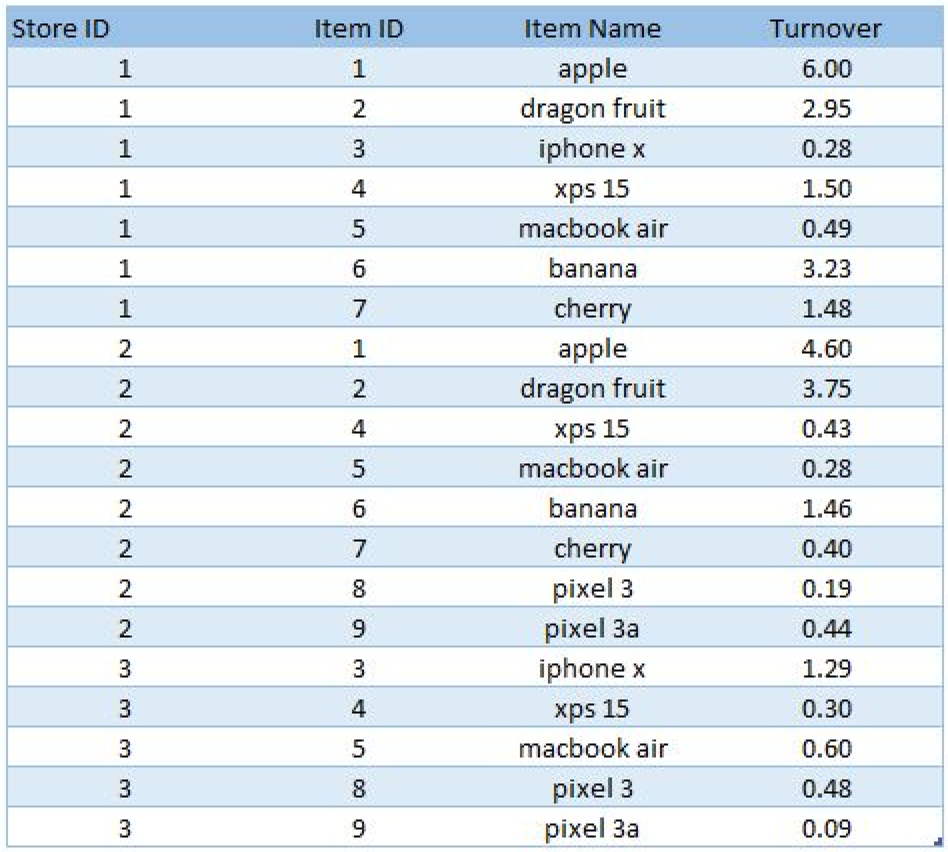
Item Name	Item ID	SKU ID	Age
iphone x	3	4	36
iphone x	3	11	34
pixel 3	8	8	34
pixel 3a	9	9	34
xps 15	4	14	33
macbook air	5	15	30
pixel 3	8	16	27
xps 15	4	37	23
xps 15	4	38	22
iphone x	3	34	21
macbook air	5	35	21
pixel 3	8	29	21
pixel 3a	9	30	21

Figure 6. Inventory Age Table

Turnover Analysis:

This analysis helps the inventory manager to measure the performance of a product line or a store. The index tells us the number of times inventory is sold in a given time period, and this can be benchmarked between store, historical data, or industry standards. The result can help us take further action. If the turnover ratio is lower than historic data, maybe the product is not popular anymore, the purchase of the product might need to be reduced to invest capital in other products. If the turnover ratio is lower than the industry standard, maybe some promotions are needed or some more competitive products could be introduced.

In figure 7, we can see that apple in store 1 has a higher turnover ratio than store 2, it might be that store 1 has a better management process to increase their sales or reduce their stocks, so it would be beneficial for the company as a whole since different stores can learn great management techniques from each other.



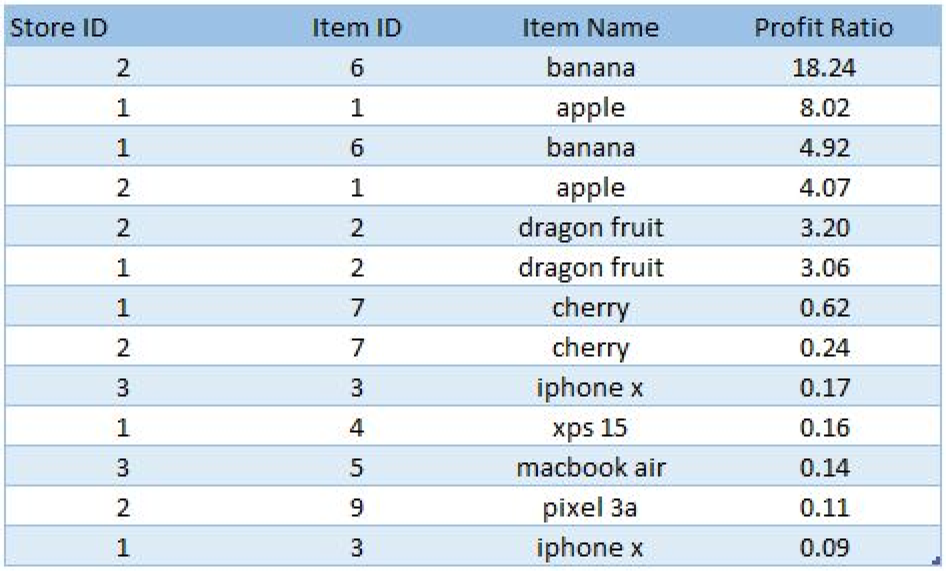
Store ID	Item ID	Item Name	Turnover
1	1	apple	6.00
1	2	dragon fruit	2.95
1	3	iphone x	0.28
1	4	xps 15	1.50
1	5	macbook air	0.49
1	6	banana	3.23
1	7	cherry	1.48
2	1	apple	4.60
2	2	dragon fruit	3.75
2	4	xps 15	0.43
2	5	macbook air	0.28
2	6	banana	1.46
2	7	cherry	0.40
2	8	pixel 3	0.19
2	9	pixel 3a	0.44
3	3	iphone x	1.29
3	4	xps 15	0.30
3	5	macbook air	0.60
3	8	pixel 3	0.48
3	9	pixel 3a	0.09

Figure 7. Turnover Table

Profit Analysis:

This return on inventory (ROI) index gives insights into the profitability of a product line or a store, and it reveals the profit earned from the investment in a unit of inventory. If the ROI of a product is high, then we need to build a strong relationship with the corresponding supplier, because that is the key to the profitability of our company. On the contrary, if the ROI is low, then the products need to be further investigated. If that product used to attract customers to come to the store, if not, then that product might need to be removed from our purchase list.

Take figure 8 as an example, apple in store 1 is more profitable than a store, and it is already known that a higher turnover ratio helps the store make more profit during the period. Again, it would be essential for store 2 to learn from store 1 how they achieve better efficiency in selling apples.
Different categories can also be compared, and it is easy to conclude that fruits are more profitable than electrical products in this company, so the company might consider spending more money in the fruit area.



Store ID	Item ID	Item Name	Profit Ratio
2	6	banana	18.24
1	1	apple	8.02
1	6	banana	4.92
2	1	apple	4.07
2	2	dragon fruit	3.20
1	2	dragon fruit	3.06
1	7	cherry	0.62
2	7	cherry	0.24
3	3	iphone x	0.17
1	4	xps 15	0.16
3	5	macbook air	0.14
2	9	pixel 3a	0.11
1	3	iphone x	0.09

Figure 8. Profit Ratio Table

Conclusions

Many small businesses manage their stock through spreadsheets or even manual account books, which could lead to missing out on orders, keeping too much inventory, and reduced profit.
Our final product is able to handle day-to-day inventory management operations necessary for small size retail stores while remains strictly compliant to FIFO inventory valuation for cost accounting purposes. Also, some inventory management techniques are applied in our system to reduce cost, improve efficiency and maximize profit for the company. The reorder point function helps lower unnecessary spending on inventory while at the same fulfilling orders from customers among multiple stores, which is not quite possible if just manual account books or

spreadsheets are applied. Some graphs are also provided in our application, making it easier to identify the potential risk and opportunity in the inventory management process. Despite many useful functions, our application is pretty easy to use by going to the URL we provide. After some simple setup, inventory can surely be managed smarter.

Author Contributions

Sikang Hu: Web development and coding(model implementation), presentation, report writing

Weihan Liu: Coding(model implementation), presentation, report writing

Jiahui Zhang: Database design and implementation, presentation, report writing

Chen Gu: Coding(model implementation), presentation, report writing

Hao Yuan: Coding(model implementation), presentation, report writing

References

1. U.S Small Business Administration, “2019 Small Business Profile”, *Small Business Administration Office Of Advocacy*, 2019. [Online]. Available: http://advocacy.sba.gov. [Accessed: July. 17, 2019].

2. U.S Census Bureau, *Statistics of U.S Businesses*, 2016. [Online]. Available: https://www.census.gov/programs-surveys/susb/data.html. [Accessed: July. 17, 2019].

3. B. Bihnam, “Small Business Marketing Trends Reports”, *Keap*, Nov. 2018. [Online]. Available: https://keap.com/resources/small-business-marketing-trends-report. [Accessed: July. 17, 2019].