The following project is to demonstrate my skills in synthesizing computing, project management and logistic.

Inventory Management & Billing System Report

Prepared by

Keagan Suah

20 November 2021

Other Information

The project has been posted online into Github, where developers contribute their ideas to the open-source community. The Python code file can be found inside. You may click on this link to view my project:

https://github.com/KeaganSuah/Inventory-management-and-billing-system/tree/Keagan-Suah/Inventory%20management%20and%20billing%20system

Contents

1. Introduction

1.1 Problem Statement

Many retail outlets keep their inventory in the back of the store, where it is mostly organized in alphabetical order. The majority of businesses use a system where they can only filter a particular model name or number to locate the inventory in the storeroom. This creates a problem where there is a lack of categorization in the system. This means that if a customer asks an employee if they have a specific shoe model in their preferred color and size, the employee will have to physically go into the storeroom and list out all the information available for the customer. The selling process is also time-consuming. Employees can only proceed with payment once the physical inventory has been located and the bar code has been scanned. This procedure consumes even more time than necessary, which results in a poor consumer experience.

1.2 Project objectives

The project I designed has a string of codes that can be applied as a phone app where employees can easily access it. The system also includes more categorization features, which allows employees to navigate and filter through the inventories to collate necessary information for customers without needing to physically enter the storeroom. Employees can view all inventories associated with their group by selecting the category of inventory they want to view. For example, if a customer requested a red shoe, it would return a list of all shoes with the color red. This allows employees to identify all the red shoes available and promote them to customers. This can help employees avoid disruption and save time when looking for inventory.

The billing system act as a cash registry that calculates the total amount purchased by the consumer. When the consumer has finished selecting their products, the calculated amount will be displayed. Additionally, this system allows employees to process payments for customers even if they do not have the physical product on hand. Meanwhile, another employee can assist in consolidating physical inventory and passing it on to customers which saves time and creates a more efficient experience for the customers.

2. Functions

Integrated Development Environment

```
142  while True:
143      # The main operation of this program, it is the start and the end of the program.
144      activity = str(input('Would you like to view the stock or make a bill (view/bill): '))
145      if activity == 'end':
146          print('Thank you and have a nice day')
147          f.close()
148          break
149      else:
150          if activity == 'view':
151              inventory_display(inventory_dict)
152          elif activity == 'bill':
153              billing(inventory_dict)
154          else:
155              print(activity + ' is a invalid command!')
```

In this project, I will be using the context of Salvatore Ferragamo, a luxury retail store outlet that is famous for its ladies' shoe. The employees of the retail outlet would be the users in this project. Viewing and billing would be the two primary functions. When the user wishes to view the inventory stock sheet, this is referred to as viewing. When the user wants to calculate the consumer's total purchase amount for payment, this is referred to as billing. The first question the project will ask is whether they want to perform an action, view something, or bill someone. Following your selection, many smaller functions will assist the main function. If the user input something that cannot be found, it will prompt the user that it is an invalid command. Entering ‘end’ means the user would like close the program, and it will output a ‘Thank you and have a nice day’ to the user.

Integrated Development Environment

```
22  def show_features(item, feature):
23      # This will be the main function for other function that requires a display of items
24      return('Model Number : ' + str(item) + ' | name : ' + str(feature[0])+ ' | colour : ' + str(feature[3]) + '| size : ' +
    str(feature[4]) + '| Qty : ' + str(feature[2]) + '| location : ' + str(feature[5]))
```

This show feature’s function will be the main display item for all the inventory management functions. It will list out all the relevant features of an inventory such as the model number, model name, colour, size, quantity, and location of the inventory. Smaller functions have this show features function to least out relevant features. However, some of these smaller functions will remove irrelevant features to display lesser values which will visually more appealing.

2.1 Viewing Function

Integrated Development Environment

```
82  def inventory_display(list):
83      #This is the main function for displaying all the stocks, if user were to choose a specific method of listing the
    ↳   inentories, other function will operate.
84      viewing_type = str(input('Choose what would you like to view (model number / name / size / colour / or all): '))
85      if viewing_type == 'all':
86          print('Stock Count list \n ------------------')
87          for item, feature in list.items():
88              print(show_features(item, feature))
89          print('------------------------')
90      elif viewing_type == 'model number':
91          display_model_number(list)
92      elif viewing_type == 'name':
93          display_model_name(list)
94      elif viewing_type == 'size':
95          display_by_size(list)
96      elif viewing_type == 'colour':
97          display_by_colour(list)
98      else:
99          print(viewing_type + ' is a invalid command!')
```

The inventory display function will prompt the user what action they would like to do in the inventory management system. After inputting the user's intention, other functions will take over to display the inventory category, and the user will proceed to the next function. If the user input something that cannot be found, it will prompt the user that it is an invalid command. Entering 'end' will break the loop and bring the user back to the first question.

Viewing entire stock sheet

Output

```
Would you like to view the stock or make a bill (view/bill): view
Choose what would you like to view (model number / name / size / colour / or all): all
Stock Count list
 ------------------
Model Number : 01R252 749540 | name : viva ballet flat | colour : pink| size : 4| Qty : 7| location : Shelf A, Row 1
Model Number : 01R252 749541 | name : viva ballet flat | colour : pink| size : 4.5| Qty : 8| location : Shelf A, Row 1
```

The first viewing option is to view everything. This function code can be found between lines 85 and 89. This function displays all existing inventories, along with their pertinent features, quantity amount and location. Each row in the stock count list represents a different model type, and it will include the model number, model name, color, size, quantity amount and location. This function comes in handy when taking inventory.

Model number

Integrated Development Environment

```
36  def display_model_number(list):
37      #function display items by its model number, this list will only have 1 item showing
38      while True:
39          product_type = str(input('enter the model number you like to view: '))
40          if product_type == 'end':
41              break
42          else:
43              for item, feature in list.items():
44                  if product_type == item:
45                      print(show_features(item,feature).replace('Model Number : ' + str(item), ''))
```

Output

```
Would you like to view the stock or make a bill (view/bill): view
Choose what would you like to view (model number / name / size / colour / or all): model number
enter the model number you like to view: 01R252 749541
 | name : viva ballet flat | colour : pink| size : 4.5| Qty : 8| location : Shelf A, Row 1
enter the model number you like to view:
```

This is the model number category function, and it will display the relevant features of the item that the user has entered. The function will prompt the user to enter the inventory's model number before displaying the inventory's remaining features. This function comes in handy when only the model number can be found, and the employees have no idea what the product is. Entering ‘end’ will end the function and bring the user to the previous function.

Model name

Integrated Development Environment

```
48  def display_model_name(list):
49      #This display item by its name, usually, this will be the most common way of navigating through the inventories
50      while True:
51          product_type = str(input('enter the name you like to view: '))
52          if product_type == 'end':
53              break
54          else:
55              for item, feature in list.items():
56                  if product_type == feature[0]:
57                      print(show_features(item, feature).replace(' | name : ' + str(feature[0]), ''))
```

Output

```
Would you like to view the stock or make a bill (view/bill): view
Choose what would you like to view (model number / name / size / colour / or all): name
enter the name you like to view: varina ballet flat
Model Number : 01A181 574556 | colour : black| size : 4| Qty : 12| location : Shelf D, Row 3
Model Number : 01A181 574557 | colour : black| size : 4.5| Qty : 2| location : Shelf D, Row 3
```

In a typical retail store outlet, the model's name category function will be used frequently. This is because when customers browse the store, the most common questions they ask are "is there another color for this shoe?" or "can I get this in my size?" This function assists employees by displaying all possible color and size options for the shoe model after entering its name. Entering ‘end’ will end the function and bring the user to the previous function.

Sizes

Integrated Development Environment

```
60  def display_by_size(list):
61      # This display all the models by the size the user type in. This function is useful for consumers that only have a ↵
        ↳   special size that they can wear. Hence searching by size, allow employees to pull out all the models of the consumers
        ↳   size for the consumers
62      while True:
63          product_type = str(input('enter the size you like to view: '))
64          if product_type == 'end':
65              break
66          else:
67              for item, feature in list.items():
68                  if product_type == feature[4]:
69                      print(show_features(item, feature).replace('| size : ' +str(feature[4]), ''))
```

Output

```
Would you like to view the stock or make a bill (view/bill): view
Choose what would you like to view (model number / name / size / colour / or all): size
enter the size you like to view: 6.5
Model Number : 01R252 749545 | name : viva ballet flat | colour : pink| Qty : 14| location : Shelf A, Row 1
Model Number : 01R252 730600 | name : viva ballet flat | colour : black| Qty : 17| location : Shelf A, Row 1
Model Number : 01R252 730606 | name : viva ballet flat | colour : new blush| Qty : 4| location : Shelf A, Row 2
```

Some consumers enjoy browsing around the store, and only ask for their size when they see something they like. On the other hand, others would like to only see shoe models that are of their size. To address all customers, sale-people have to put these people into consideration to provide a pleasant experience for the customer. This function helps to categorize the list and only shows shoe models that have the size the user keyed in. Entering ‘end’ will end the function and bring the user to the previous function.

Colour

Integrated Development Environment

```
71  def display_by_colour(list):
72      # Consumers have different taste and prefernce, some would only like to buy shoes of their favourite colour. With this
    ↳   function, it allows user to list all the inventories that's the same colour as what the user has inputted
73      while True:
74          product_type = str(input('enter the colour you like to view: '))
75          if product_type == 'end':
76              break
77          else:
78              for item, feature in list.items():
79                  if product_type == feature[3]:
80                      print(show_features(item, feature).replace(' | colour : ' + str(feature[3]), ''))
```

Output

```
Would you like to view the stock or make a bill (view/bill): view
Choose what would you like to view (model number / name / size / colour / or all): colour
enter the colour you like to view: black
Model Number : 01R252 730595 | name : viva ballet flat| size : 4| Qty : 5| location : Shelf A, Row 1
Model Number : 01R252 730596 | name : viva ballet flat| size : 4.5| Qty : 18| location : Shelf A, Row 1
Model Number : 01R252 730597 | name : viva ballet flat| size : 5| Qty : 9| location : Shelf A, Row 1
```

Similar to the size function, this function helps to categorize the list and only shows shoe models that have the colour the user keyed in. Consumers have different tastes and preferences and may only like their shoes to be a specific colour. This function addresses these types of consumers, allowing them to choose the model type and size they desire. Entering ‘end’ will end the function and bring the user to the previous function.

2.2 Billing Function

Purchasing Item List

Integrated Development Environment

```
def billing_list(list):
    # This function is used to display item in a format of model name, colour and size only. This remove other features that customer are not interested in
    product_type = '6.5'
    # For this function to work, I've decided to print all the models but only the size 4 items, as all the shoe has a size 6.5 in common, this will only show 1 size for each model, preventing any duplicates in the list
    print('Item listing \n ------------------')
    for item, feature in list.items():
        if product_type == feature[4]:
            print(show_features(item, feature).replace('Model Number : ' + str(item),'').replace('| size : ' +str(feature[4]) + '| Qty : ' + str(feature[2]) + '| location : ' + str(feature[5]),'') + ' | price : $' + str(feature[1]))
    print('------------------')
```

This function displays the item list for the billing system, as it shows relevant features that are more readable and for selling. The features that will be shown are the name, colour and size.

Billing Process

Integrated Development Environment

```
def billing(list):
    # This is the billing function that charges the consumers for what they have purchased
    total_bill = 0
    while True:
        billing_list(inventory_dict)
        model_type = str(input('model name: '))
        colour_type = str(input('colour: '))
        if model_type == 'end':
            break
        else:
            for item, feature in inventory_dict.items():
                if model_type == feature[0] and colour_type == feature[3]:
                    print(show_features(item, feature).replace(' | name : ' + str(feature[0])+ ' | colour : ' + str(feature[3]),'') + ' | price : $' + str(feature[1]))
            user_keyed_item = str(input('Please key in the model number: '))
            # Most stores will usually scan the item model when processing a transaction, as the model number includes the model type, the size and the colour
            if user_keyed_item in list:
                try:
                    number_item = int(input('Enter the number of item purchased: '))
                except ValueError:
                    print('Please enter a number')
                else:
                    quantity = list[user_keyed_item][2]
                    quantity = int(quantity)
                    # This will remove update the quantity left automatically, so when a consumer purhcase X amount of item, it will minus X amount from the inventory list
                    if number_item > int(quantity):
                        print('Sorry, insufficient amount')
                    else:
                        quantity -= number_item
                        chosen_item = list.get(user_keyed_item)
                        total_bill += float(chosen_item[1])*number_item
                        # It calcualates and total up the inventory purchased and it will be shown at the end
                        list[user_keyed_item][2] = str(quantity)
                        proceed = str(input('Would you like to process another transaction? (yes/no) : '))
                        if proceed == 'no':
                            print("--------------- \nTotal bill is : ${}\n---------------".format(round(total_bill,3)))
                            break
                        else:
                            print('Item not found')
            else:
                print('The model number ' + user_keyed_item + ' cannot be found in the list')
```

Output

```
Would you like to view the stock or make a bill (view/bill): bill
Item listing
 ------------------
 | name : viva ballet flat | colour : pink | price : $595
 | name : viva ballet flat | colour : black | price : $595
 | name : viva ballet flat | colour : new blush | price : $595
 | name : viva ballet flat | colour : cumin | price : $595
 | name : viva ballet flat | colour : flamingo | price : $595
```

The billing function process will begin after the user enters 'bill' into the project's first question prompt. The item list is then displayed for the user to see, similar to the view entire stock sheet function. The only distinction is that it only displays the model's name, color, and price. This is done to ensure that the items displayed are easy to read, and other unnecessary features will not be displayed. Entering ‘end’ will end the function and bring the user to the previous function.

Output

```
 | name : vara bow pump | colour : black | price : $560
 ------------------
model name: vara bow pump
colour: black
Model Number : 01B221 749576| size : 4| Qty : 7| location : Shelf F, Row 3 | price : $560
Model Number : 01B221 749577| size : 4.5| Qty : 6| location : Shelf F, Row 3 | price : $560
```

At the bottom of the item list, the user will be prompted with two questions, one for the model’s name and one for the color. When you enter the model’s name and colour, it will display a list of the other relevant features, which includes the model number, size, quantity, location of the product, and price of the desired product. In this list, the user will also select the consumer shoe size and enter the model number from the same row as the desired shoe size. If the user enters a model number that cannot be found in the list, the program will tell the user that the model cannot be found and will prompt the user to re-enter the product.

Output

```
Model Number : 01B221 749591| size : 11.5| Qty : 5| location : Shelf F, Row 3 | price : $560
Please key in the model number: 01B221 749591
Enter the number of item purchased: 7
Sorry, insufficient amount
Item listing
 -----------------
 | name : viva ballet flat | colour : pink | price : $595
```

By entering the model number, the consumer confirms that he or she wishes to purchase the selected item. Entering ‘end’ will end the function and bring the user to the previous function. The project will compute the price multiplied by the amount purchased by the consumer. However, if the available stock count is less than what the customer desires, it will prompt that there is insufficient stock, and the item list will be displayed again. If the user enters something other than a number, it will raise a value-error. The program will prompt the user to enter a number instead and request the user to re-enter.

Output

```
Model Number : 01B221 749591| size : 11.5| Qty : 5| location : Shelf F, Row 3 | price : $560
Please key in the model number: 01B221 749591
Enter the number of item purchased: 5
Would you like to process another transaction? (yes/no) : no
--------------
Total bill is : $2800.0
--------------
```

When the customer demand is less than the stock count, it means that there is enough in the stock count. The total calculated amount will be added to the bill, and the user will be prompted with another question. The user will be asked if they want to continue purchasing. If the user selected "yes," the item list will be displayed, along with the two questions about the model's name and color again. If the user selected "no," the project will calculate all of the purchased models and display the total cost.

2.3 Reading CSV File Function

Comma-Spaced-Value File

	A	B	C	D	E	F	G
1	01R252 749540	viva ballet flat	595	7	pink	4	Shelf A, Row 1
2	01R252 749541	viva ballet flat	595	8	pink	4.5	Shelf A, Row 1
3	01R252 749542	viva ballet flat	595	12	pink	5	Shelf A, Row 1
4	01R252 749543	viva ballet flat	595	7	pink	5.5	Shelf A, Row 1

Integrated Development Environment

```
import os
import csv

dir_path = os.path.dirname(os.path.realpath(__file__))
filepath = ''
#To open the CSV file and read it
for root, dirs, files in os.walk(dir_path):
    for file in files:
        if file == "inventories.csv":
            filepath += root +'\\' + str(file)

f = open(filepath,'r')
readr = csv.reader(f)
#The data in the csv file will be transfered into python as a dictionary item
inventory_dict = {}

for row in readr:
    inventory_dict[row[0]]= [row[1],row[2],row[3],row[4],row[5],row[6]]
    # for this, the main key of the dictionary will be the model number, and he values will be stored in a list of model
    name, price, quantity, colour, and size
```

Output

```
[evaluate Inventory Management and Billing System.py]
{'01R252 749540': ['viva ballet flat', '595', '7', 'pink', '4', 'Shelf A, Row 1'], '01R252 749541': ['viva b
```

The whole project works well with data structures such as dictionaries and lists, so the best way to add value to the data structure is by importing a comma-spaced-value (CSV) file into the project. The CSV file act as data storage that can be amended. The series of code helps the project to read the CSV file from an excel sheet and converts it into a dictionary. In the excel sheet, each column represents the different features of the model, while each row represents different shoe models. In the dictionary, the key represents the model number, while the other features are placed in a list in the value of the dictionary.

3. Conclusion

The inventory management and billing systems saves time and allows employees to address consumer needs and preferences. The inventory management system provides a platform for the employees to easily access and navigate around the inventory on the go. The billing system lets employees choose and pick out the inventories that the consumers are interested to purchase. Both the systems are important to have as it can bridge the gap and enhance the efficacy of businesses. However, there are some areas where the system can be improved to provide a better quality of service to the consumer while also giving the business a competitive advantage.

3.1 Future Enhancements

Machine learning tools can assist in accurately forecasting future supply and demand. Data mining can reveal consumer tastes and preferences and find the most popular product amongst all based on the previous overall sales. After cleaning the data, data visualization tools such as Tableau or Power BI can be used to provide insights and make prediction on consumer demands using graphs.

Using the insights obtained, the feature can identify more desirable inventories and help with stock replenishment by requesting a larger amount of restocking for popular products. This ensures that there will be sufficient supply to meet the anticipated increase in demand. It can also eliminate product that are not popular and reduce excess stock to decrease stock wastage.

The same insights can be applied to the billing system as well. By incorporating a recommender feature into the system to inform employees of products that consumers may be interested in. As a result, these system enhancements can help the company's competitiveness even more.

Data collected can also help to improve the system by incorporating the internet of things. Track stock movement in real-time across all channels to ensure that the business got the right number of products in the right place at the right time. This will greatly improve the business quality of service, as consumers are able to receive their product on time.

4. Appendices

1. Salvatore Ferragamo. (n.d.). *Shoes*. Italian Leather Shoes for Women | Salvatore Ferragamo EU. Retrieved March 7, 2022, from https://www.ferragamo.com/shop/eu/en/women/shoes
2. Rafia . (2019, December 20). *Blog*. LogicAI. Retrieved March 7, 2022, from https://logicai.io/blog/predicting-customer-behaviour-artificial-intelligence/#4.-Reshaping-Customer-Experience-Through-Enhanced-Communication