# app

August 7, 2024

## 1 A supermarket billing system is designed to manage transactions and inventory efficiently. Here are some key features typically found in such systems:

### 1.1 1. Point of Sale (POS)

#### 1.1.1 Barcode Scanning: Scans barcodes to quickly retrieve product information and prices.

[ ]:
```
import os
from pyzbar.pyzbar import decode
from PIL import Image
def scanBarCode(name_of_bar_code):
    pass
import barcode
from barcode.writer import ImageWriter


class BarCodeTools:
    def createBar(name,data,path = os.getcwd()):
        barcode_type = 'code128'
        code = barcode.get_barcode_class(barcode_type)
        barcode_data = data
        barcode_instance = code(barcode_data, writer=ImageWriter())
        barcode_instance.save(f'{path}/barcode/{name}')
        print('saved')

    def readingBar(name,path=os.getcwd()):
        for item in decode(Image.open(f'{path}/barcode/{name}.png')):
            return item.data.decode('utf-8')
```

#### 1.1.2 Manual Entry: Allows manual entry of product codes or descriptions if barcodes are not available.

### 1.2 2. Inventory Management

[ ]:
```
import pandas as pd
inventoryData = pd.read_csv(f'{os.getcwd()}/inventory/inventory.csv')
def intializing():
    for i in range(0,len(inventoryData['barnumber'])):
        BarCodeTools.
 ↪createBar(inventoryData['name'][i],str(inventoryData['barnumber'][i]))
```

#### 1.2.1 Stock Tracking: Monitors inventory levels in real-time to prevent stockouts and overstocking.

[ ]:
```
class StockManager:
    def reduceStock(name):
        for index in range(0,len(inventoryData['name'])):
            if inventoryData['name'][index]==name :
                itemIndex = index
                break
        inventoryData.loc[itemIndex,'available'] =␣
 ↪inventoryData['available'][itemIndex] -1
        inventoryData.to_csv(f"{os.getcwd()}/inventory/inventory.
 ↪csv",index=False)

    def AddToStock(name,number):
        for index in range(0,len(inventoryData['name'])):
            if inventoryData['name'][index] == name:
                itemIndex = index
                break
        inventoryData.loc[itemIndex,'available'] =␣
 ↪inventoryData['available'][itemIndex]+int(number)
        inventoryData.to_csv(f"{os.getcwd()}/inventory/inventory.
 ↪csv",index=False)
```

#### 1.2.2 Reorder Alerts: Notifies when stock levels fall below a predefined threshold.

[ ]:
```
def checkStock():
    for item in inventoryData['available']:
        try:
            if int(item)<5:
                print("stocks are less")
                return False
            else:
                return True
        except Exception as e:
```

```
            print("some empty values are neglected")
```

#### 1.2.3 Supplier Management: Keeps records of suppliers and purchase orders.

[ ]:
```
supplyData = pd.read_csv(os.path.join(os.getcwd(),'supplier/pending.csv'))

def calculatePendingAmount():
    try:
        for index in range(0,len(inventoryData['name'])):
            supplyData.loc[index,'name'] = inventoryData['name'][index]
            supplyData.loc[index,'supplier'] =␣
 ↪inventoryData['supply_manager'][index]
            supplyData.loc[index,'amount'] =␣
 ↪int(inventoryData['available'][index])*int(inventoryData['price'][index])
            supplyData.to_csv(os.path.join(os.getcwd(),'supplier/pending.
 ↪csv'),index=False)
        return True
    except Exception as e:
        return False
```

### 1.3 3. Pricing and Discounts

#### 1.3.1 Price Management: Updates and manages pricing for products.

[ ]:
```
def getPrice(name):
    for i in range(0,len(inventoryData['name'])):
        if inventoryData['name'][i] == name :
            return inventoryData['price'][i]
        else:
            return None
```

#### 1.3.2 Discounts and Promotions: Applies discounts, promotional offers, and loyalty rewards.

[ ]:
```
from random import randint
def discount(item):
    for index in range(0,len(inventoryData['name'])):
        if inventoryData['name'][index]==item:
            randomDiscount = float(randint(0,10)/100)
            newPrice =␣
 ↪inventoryData['price'][index]-float(inventoryData['price'][index]*randomDiscount)
            return f'{newPrice}'
```

## 1.4 4. Transaction Processing

### 1.4.1 Sales Recording: Logs sales transactions and generates receipts.

[ ]:
```
salesData = pd.read_csv(os.path.join(os.getcwd(),'sales/sales.csv'))

from datetime import datetime
def sellItem(name,customer_name,GW):
    StockManager.reduceStock(name)
    initialLength = len(salesData['item'])
    price = discount(name)
    for index in range(0,len(inventoryData['name'])):
        if inventoryData['name'][index]==name:
            salesData.loc[initialLength,'s_no'] = initialLength+1
            salesData.loc[initialLength,'item'] = name
            salesData.loc[initialLength,'sold_at'] = price
            salesData.loc[initialLength,'sold_to'] = customer_name
            salesData.loc[initialLength,'sold_on'] = datetime.now().date()
            salesData.loc[initialLength,'guarranty/warranty'] = GW
            break
    salesData.to_csv(os.path.join(os.getcwd(),'sales/sales.csv'),index=False)
    receipt(name)

def receipt(item):
    for index in range(0,len(salesData['item'])):
        if salesData['item'][index]==item:
            recieptItem = salesData['item'][index]
            price_on_reciept = salesData['sold_at'][index]
            customer_name = salesData['sold_to'][index]
            gw = salesData['guarranty/warranty'][index]
            sold_on = salesData['sold_on'][index]
    print(f'''      SHOP NAME   ''')
    print('''*'''*27)
    print(f'item\t: {recieptItem}')
    print(f'price\t: {price_on_reciept}')
    print(f'customer: {customer_name}')
    print(f'guarrantee/warranty:{gw}')
    print('''*'''*27)
    print(f"{sold_on}\tsignature")
```

## 1.5 6. Reporting and Analytics

### 1.5.1 Sales Reports: Provides detailed sales reports including daily, weekly, and monthly summaries.

[ ]:
```
def getTotalSales():
    total = 0
    for item in salesData['sold_at']:
        total+=float(item)
    print(f"your total sale at {datetime.now().date()} is {total}")
```

**Assembling the main app**

[ ]:
```
intializing()
print(checkStock())
calculatePendingAmount()
print(inventoryData)
sellItem('roter commercial cooler','rhydham','G1.5YRS')
# discount('roter commercial cooler')
getTotalSales()
```

```
saved
saved
saved
saved
True
   s_no                    name  available  barnumber supply_manager  price
0     1             luminousfan         30    6868636    shop_name_1   5132
1     2               orientfan          7    8783636    shop_name_2   6764
2     3                 hhkfhfh          8    7868646    shop_name_3   7837
3     4  roter commercial cooler          9     876798     desrajshop  13500
      SHOP NAME
**************************
item    : roter commercial cooler
price   : 12960.0
customer: rhydham
guarrantee/warranty:G1.5YRS
**************************
2024-08-07      signature
your total sale at 2024-08-07 is 12960.0
```