**A white paper with text and a computer chip

Description automatically generated with medium confidence**

**Introduction**

Insightful financial reporting and effective inventory management are two features that this Python script offers to businesses. It also includes a comprehensive store sales reporting system. The script collects and analyses product information, computes earnings, produces comprehensible reports, keeps an Excel record, and uses a dynamic pie chart to show the distribution of profits. It does all of this by utilizing the concepts of object-oriented programming.

**Principal Elements:**

1. **Category of Product:**

* The ProductItem class, which represents specific shop products, is the foundation of the system. Important details including the product name, price, kind, and quantity of stock are contained in this class.

1. **How to Calculate Profit:**

* By calculating the cost of each product by the quantity of stock on hand, the script determines the profit for each one. Financial analysis is based on this basic computation on an item-by-item basis.

1. **Readable by Humans Sales Report:**

* The product data is converted into an extensive and legible sales report using the formatItemDetails function. It contains pertinent information such as the product name, price, kind, quantity, and profit margin. The report also includes a consolidation and presentation of the store's total earnings.

1. **Creating an Excel Report:**

* The storeSalesReport.xlsx Excel report creation and updating capabilities is provided by the script. As the product list changes, this report adjusts to act as a dynamic record. Headers like Product Name, Price, Stock, Product Type, Profit, and Entry Date and Time are among them.

1. **Handling User Input:**

* Using the getUserInput method, the system makes sure that user input is handled robustly. It asks consumers to contribute information about specifics of the product, ensuring that the data they supply is accurate.

1. **Collecting Interactive Product Information:**

* Users are guided through the process of entering product information via the collectProductItems function. Users first indicate how many goods they wish to report on, then supply specifics for each product, such as name, price, amount in stock, and type.

1. **Visualising Data with Matplotlib:**

* Beyond only textual reporting, the script uses the matplotlib package to produce a pie chart that shows how earnings are divided across various goods. This graphic depiction provides a brief and understandable synopsis of the store's financial situation.

**In summary:**

In conclusion, this store sales reporting system gives businesses a full arsenal that enables them to keep precise records, evaluate profit creation, and learn more about how each product affects overall profitability. The project covers topics such object-oriented design, file management, user input validation, and data visualisation and provides a useful example of Python programming. This screenplay provides an invaluable framework for the implementation and comprehension of inventory management and financial reporting systems, suitable for both small-scale retail businesses and educational settings.

Python Code :

import openpyxl

from datetime import datetime

import os.path

import matplotlib.pyplot as plt

class ProductItem:

def \_\_init\_\_(self, itemName, itemCost, itemStock, itemType):

self.itemName = itemName

self.itemCost = itemCost

self.itemStock = itemStock

self.itemType = itemType

def calculateTotalProfit(item):

return item.itemCost \* item.itemStock

def formatItemDetails(items):

salesReport = "Store Sales Report:\n\n"

totalRevenue = 0

for item in items:

profit = calculateTotalProfit(item)

salesReport += f"Product: {item.itemName}\n"

salesReport += f"Cost: ${item.itemCost:.2f}\n"

salesReport += f"Stock: {item.itemStock}\n"

salesReport += f"Type: {item.itemType}\n"

salesReport += f"Profit: ${profit:.2f}\n\n"

totalRevenue += profit

salesReport += f"Overall profit from the entire store is: ${totalRevenue:.2f}\n"

return salesReport

def createSalesReport(items):

reportFileName = "storeSalesReport.xlsx"

if not os.path.isfile(reportFileName):

salesWorkbook = openpyxl.Workbook()

salesWorksheet = salesWorkbook.active

headers = ["Product Name", "Cost", "Stock", "Product Type", "Profit", "Date and Time"]

for columnNumber, columnHeader in enumerate(headers, 1):

salesWorksheet.cell(row=1, column=columnNumber, value=columnHeader)

else:

salesWorkbook = openpyxl.load\_workbook(reportFileName)

salesWorksheet = salesWorkbook.active

for rowNumber, productItem in enumerate(items, salesWorksheet.max\_row + 1):

salesWorksheet.cell(row=rowNumber, column=1, value=productItem.itemName)

salesWorksheet.cell(row=rowNumber, column=2, value=productItem.itemCost)

salesWorksheet.cell(row=rowNumber, column=3, value=productItem.itemStock)

salesWorksheet.cell(row=rowNumber, column=4, value=productItem.itemType)

salesWorksheet.cell(row=rowNumber, column=5, value=calculateTotalProfit(productItem))

salesWorksheet.cell(row=rowNumber, column=6, value=datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

salesWorksheet.cell(row=salesWorksheet.max\_row + 1, column=1, value="Total Profit")

salesWorksheet.cell(row=salesWorksheet.max\_row, column=5, value=sum(calculateTotalProfit(productItem) for productItem in items))

try:

salesWorkbook.save(reportFileName)

print(f"Sales report updated successfully at {reportFileName}")

except PermissionError:

print(f"PermissionError: Unable to save the file. Check write permissions for the specified path.")

def getUserInput(prompt, dataType=float):

validInput = False

userInputData = None

while not validInput:

try:

userInputData = dataType(input(prompt))

validInput = True

except ValueError:

print("Invalid input. Please provide a valid value.")

return userInputData

def collectProductItems():

print("Welcome to the Store Sales Reporting System!")

numOfItems = int(getUserInput("Provide the number of items you want to create a report for: ", int))

productItems = []

for i in range(1, numOfItems + 1):

print(f"\nProvide details for item {i}:")

itemName = input("Provide the product name: ")

itemCost = getUserInput("Provide the cost of the item: ")

itemStock = getUserInput("Provide the current stock quantity: ", int)

itemType = input("Provide the type of the item: ")

productItem = ProductItem(itemName, itemCost, itemStock, itemType)

productItems.append(productItem)

print("\nThank you for providing the details!")

return productItems

productItems = collectProductItems()

salesReport = formatItemDetails(productItems)

print(salesReport)

createSalesReport(productItems)

totalRevenue = sum(calculateTotalProfit(productItem) for productItem in productItems)

itemProfits = [calculateTotalProfit(productItem) for productItem in productItems]

itemNames = [productItem.itemName for productItem in productItems]

percentages = [(profit / totalRevenue) \* 100 for profit in itemProfits]

plt.figure(figsize=(8, 8))

plt.pie(percentages, labels=itemNames, autopct='%1.1f%%', startangle=140)

plt.title("Profit Distribution Among Items")

plt.axis('equal')

plt.show()

Output screenshot:

A screenshot of a computer

Description automatically generated

Generated graph screenshot:

A pie chart with numbers and a number of items

Description automatically generated

Excel Sheet Screenshot:

A screenshot of a computer

Description automatically generated