

Universiti Tunku Abdul Rahman

Foundation in Science

**FHCT1024 Programming Concepts and Design**

Session 202201

FINAL REPORT

|  |  |
| --- | --- |
| **Practical Group: P6** |  |
| **Practical Lecturer’s Name:** | **Ling Kock Sheng** |
| **Group Name: Group 3** |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Student ID | Signature |
| 1. | Elisha Wong Peh Jie | 2104945 |  |
| 2. | Elisabeth Lee Mei Jin | 2104817 |  |
| 3. | Lee Shu En | 2104408 |  |
| 4. | Wong Siew Hang | 2104026 |  |

1. **Table of Content**

|  |  |
| --- | --- |
| Content | Page Number |
| *Flowchart* | *Page 3- 44* |
| *Screenshots* | *Page 45-50* |
| *Program Codes* | *Page 51-64* |

1. **Task Distribution**

|  |  |
| --- | --- |
| Name | Task |
| Elisha Wong Peh Jie | * Group leader, merging files, * Coding and flowchart for Membership\_name\_dup\_check * Coding for CRUD, displayRec, overwriteFile * Contributed in MembershipMaint |
| Elisabeth Lee Mei Jin | * Coding and Flowchart for Additm, Payment, readFile, addNewItm, receipt * Contributed in MembershipMaint * Flowchart for Final |
| Lee Shu En | * Coding and flowchart for main menu, item maintenance, receipt, welcome, delItem, saveFile, * Contributed in MembershipMaint * Flowchart for CRUD, overwriteFile, MembershipMaint |
| Wong Siew Hang | * Coding and flowchart for displayMembership, updItem, updateFile, * Contributed in MembershipMaint * Coding for Final |

1. **Flowcharts**

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: welcome()** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: main()** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: crud\_action\_menu** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: readFile(filename)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: displayRec(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function:displayMembership(Membership)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: displaySalesHist(SalesHist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: addNewItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: addNewItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: addNewItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: delItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: updItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: updItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: updItem(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: saveFile(AssignmentList)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: overwriteFile(arr, filename)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: updateFile(arr, filename)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: additem(AssignmentList, membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: Payment(AssignmentList, disc\_rate)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: Payment(AssignmentList, disc\_rate)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: receipt(AssignmentList,disc\_rate,is\_reprint)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: receipt(AssignmentList,disc\_rate,is\_reprint)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: receipt(AssignmentList,disc\_rate,is\_reprint)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: receipt(AssignmentList,disc\_rate,is\_reprint)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module: flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: MembershipMaint(membershiplist)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: membership\_name\_dup\_check(arr, name)** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: Final()** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: Final()** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: Final()** |
|  |

|  |
| --- |
| **Module:** **flowchart.drawio**  **Function: Final()** |
|  |

1. **Screenshots**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: main()** |
| *Text  Description automatically generated* |

**Item Maintenance**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: addNewitem(AssignmentList)** |
| Graphical user interface  Description automatically generated |

**Sales History**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: displaySalesHist(SalesHist)** |
| Graphical user interface  Description automatically generated |

**Sales**

**Membership**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: displayMembership(Membership)** |
|  |

**Adding Item for Payment**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: addItm(AssignmentList, membershiplist)** |
|  |

**Payment**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: Payment(AssignmentList, disc\_rate)** |
| Text  Description automatically generated with medium confidence |

**Receipt**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: receipt(AssignmentList,disc\_rate,is\_reprint)** |
| **Text  Description automatically generated** |

**Inventory Shortfall**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: addItm(AssignmentList, membershiplist)** |
| **Graphical user interface, text  Description automatically generated** |

**Data Validation**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: final()** |
| **Text  Description automatically generated** |

**Membership Maintenance**

|  |
| --- |
| **Module:** **FHCT1024 202201 Assignment.py**  **Function: final()** |
| **Text  Description automatically generated** |

**Reprint Last Receipt**

|  |
| --- |
| **Module: FHCT1024 202201 Assignment.py**  **Function: receipt(AssignmentList,disc\_rate,is\_reprint)** |
| **Text  Description automatically generated** |

1. **Source-code (program code)**

#program module name : FHCT1024 202201 Assignment.py

#purpose : Group Assignment

#Date created : 7 March 2022

#programmers : Elisha Wong Peh Jie

# Elisabeth Lee Mei Jin

# Lee Shu En

# Wong Siew Hang

import datetime

import time

from os import system

t=[]

sls={}

itmdict={} #assign empty dictionary with variable name "itmdict"

membershiplist = []

disc\_rate = 0

#Welcome

def welcome():

x = datetime.datetime.now()

print(x.strftime("%d/%m/%Y"))

print(x.strftime("%H:%M %p"))

print("-"\*50)

print("\t~Welcome to Mask Face Sdn Bhd‍~")

print("-"\*50)

#Main menu

def main():

print("\n>>Main Page<<")

print("-"\*50)

print("Mask Face Sdn Bhd")

print("-"\*50)

print("<1> Stock/inventory Management")

print("<2> Sales History")

print("<3> Sales")

print("<4> Membership Maintenance")

print("<5> Reprint Last Receipt")

print("<Q> Quit")

print("-"\*50)

#CRUD Action menu

def crud\_action\_menu():

print("-" \* 50)

print("Membership Maintenance Menu")

print("-" \* 50)

print("<1> Add new")

print("<2> View")

print("<3> Update")

print("<4> Delete")

print("<Q> Quit")

print("-" \* 50)

#read from txt file

def readFile(filename):

fileobj = open(filename, "r")

lines = fileobj.readlines() #convert to a list (each line= 1 element)

tmp = []

for line in lines:

tLst = line.strip("\n").split("|")

#strip -> strip "\n" at end of the line

#split -> convert the string to a list, based on "|"

tmp.append(tLst)

fileobj.close()

return tmp

#display product

def displayRec(AssignmentList):

print("-"\*60)

print("ItemCode ItemDesc Amount UOM Price ")

print("-"\*60)

for i in range(len(AssignmentList)): #using reference

#print(itmLst[idx])

print(" %s %-25s %4d %-4.4s %6.2f"%(AssignmentList[i][0],

AssignmentList[i][1], int(AssignmentList[i][2]), AssignmentList[i][3], float(AssignmentList[i][4])))

print("-"\*60)

#display membership details

def displayMembership(Membership):

print("-"\*60)

print("No Tier Discount Rate (%)")

print("-"\*60)

for i in range(len(Membership)):

print("%s %-20.20s %4d" % (i+1, Membership[i][0], float(Membership[i][1])))

print("-"\*60)

#display sales history

def displaySalesHist(SalesHist):

print("-"\*65)

print("Date Time SubTotal Discount Paid Amount")

print("-"\*65)

for i in range(len(SalesHist)):

print("%s %s %7.2f %8s %7.2f"% (SalesHist[i][0], SalesHist[i][1], (float(SalesHist[i][2])), SalesHist[i][3], (float(SalesHist[i][4]))))

print("-"\*65)

def addNewitem(AssignmentList): #mainItem(itmLst, opt):

#testcode

print("inside addNewItem")

loop = True

step = 1

while loop:

if step == 1:

itm = input("Enter item code <Q>uit >>")

if itm == "Q" or itm == "q":

step = 99

elif len(itm) != 5:

print("Invalid item code added")

elif itm in [x[0] for x in AssignmentList]:

print("Item code already exist")

else:

step = step+1

if step == 2:

itmDesc = input("Enter item Description <Q>uit >>")

if itmDesc == "Q" or itmDesc == "q":

step = 99

else:

step += 1

if step == 3:

amt = input("Enter item Amount <Q>uit >>")

if amt == "Q" or amt == "q":

step = 99

elif not amt.isdigit():

print("Invalid item Amount entered")

else:

step += 1

if step == 4:

UOM = input("Enter item UOM <Q>uit >>")

if UOM == "Q" or UOM == "q":

step = 99

else:

step = step+1

if step == 5:

itmPr = input("Enter item price <Q>uit >>")

if itmPr == "Q" or itmPr == "q":

step = 99

else: #need to check for floating point number

step += 1

if step==6: #complete input

tLst=[itm, itmDesc, amt, UOM, itmPr]

AssignmentList.append(tLst)

loop=False

if step==99:

loop=False

return AssignmentList

def delItem(AssignmentList):

loop = True

step = 1

while loop:

if step == 1:

itm = input("Enter item <Q>uit >>")

if itm == "Q":

step = 99

elif itm not in [x[0] for x in AssignmentList]:

print("Item code not found")

else: #need to check if item exists

step=step+1

if step==2:

idx=[x[0] for x in AssignmentList].index(itm)

AssignmentList.pop(idx)

#need to pop it from itmLst

print("Remove successful")

loop=False

if step==99:

loop=False

return AssignmentList

def updItem(AssignmentList):

loop=True

step=1

while loop:

if step==1:

itm= input("Enter item <Q>uit >>")

if itm=="Q":

step=99

elif itm not in [x[0] for x in AssignmentList]:

print("Item code Found")

else: #need to check if item exists

step=step+1

if step==2:

itmDesc=input("Enter item Description <Q>uit >>")

if itmDesc=="Q":

step=99

else:

step+=1

if step==3:

amt= input("Enter item Amount <Q>uit >>")

if amt=="Q":

step=99

elif not amt.isdigit():

print("Invalid item Amount entered")

else:

step +=1

if step==4:

UOM= input("Enter item UOM <Q>uit >>")

if UOM=="Q":

step=99

else:

step=step+1

if step==5:

itmPr= input("Enter item price <Q>uit >>")

if itmPr=="Q":

step=99

else: #need to check for floating point number

step +=1

if step==6: #complete input for update

tLst=[itm, itmDesc, amt, UOM, itmPr]

#list comphresion to locate pos/record

idx=[x[0] for x in AssignmentList].index(itm)

AssignmentList[idx]=tLst

loop=False

if step==99:

loop=False

return AssignmentList

def saveFile(AssignmentList):

wStr = ""

for rec in AssignmentList:

wStr += "|".join(rec)+"\n"

print(wStr)

f = open("AssignmentList.txt", "w")

f.write(wStr)

f.close()

def overwriteFile(arr, filename):

wStr = ""

for rec in arr:

wStr += "|".join(rec) + "\n"

f = open(filename, "w")

f.write(wStr)

f.close()

def updateFile(arr, filename):

wStr = ""

for rec in arr:

wStr += "|".join(rec) + "\n"

f = open(filename, "a")

f.write(wStr)

f.close()

#addItmPayment

def addItm(AssignmentList, membershiplist):

global t

global sls

global itmdict

global disc\_rate

fileobj = open("AssignmentList.txt", "r")

lines = fileobj.readlines() # convert a list (each list=1 element)

for line in lines:

tLst = line.strip("\n").split("|")

t.append(tLst)

fileobj.close()

for i in range(len(t)):

itmdict[t[i][0]] = t[i][1:] # convert t which was a list into a data dictionary

x = datetime.datetime.now()

date = x.strftime("%Y-%m-%d")

time = x.strftime("%H:%M:%S")

print("-"\*65)

print("Sales Transaction Menu --> Date:", date, " Time:", time)

print("-"\*65)

loop = True

step = 0

disc\_rate = 0

while loop:

if step == 0:

#select membership

displayMembership(membershiplist)

sel\_membership = input("Please select Membership <Q>uit >> ")

if sel\_membership == "Q" or sel\_membership == "q":

loop = False

elif int(sel\_membership) < 0 or int(sel\_membership) > len(membershiplist):

print("#############################")

print("# Invalid Response #")

print("#############################")

else:

#get disc rate

disc\_rate = 1 - (int(membershiplist[int(sel\_membership)-1][1])/100)

step += 1

if step == 1:

#display item list for selection

displayRec(AssignmentList)

itmc = input("Enter item Code <P>ayment <Q>uit >> ")

if itmc == "Q" or itmc == "q":

loop = False

if itmc == "P" or itmc == "p":

Payment(AssignmentList, disc\_rate)

elif itmc not in itmdict.keys():

print("Invalid item code")

else:

Desc = itmdict[itmc][0]

price = itmdict[itmc][3]

print("%s(RM%s)" % (Desc, price))

step += 1

if step == 2:

print("%s left" % itmdict[itmc][1])

qty = input("Enter quantity <B>ack <Q>uit >> ")

if qty == "Q" or qty == "q":

loop = False

elif qty == "B" or qty == "b":

step -= 1

elif qty.isdigit() == False: #Check whether qty is integer

print("Please enter an integer")

elif 0 < int(qty) <= int(itmdict[itmc][1]):

amt = int(itmdict[itmc][1]) - int(qty)

itmdict[itmc][1] = amt

print("%s left" % amt)

#to update latest inventory amount

for i in range(len(AssignmentList)):

if AssignmentList[i][0] == itmc:

AssignmentList[i][2] = str(amt)

if itmc in sls.keys():

sls[itmc] += int(qty)

else:

sls[itmc] = int(qty)

step -= 1

else:

print("Insufficient stock")

#Payment

def Payment(AssignmentList, disc\_rate):

x = datetime.datetime.now()

date = x.strftime("%Y-%m-%d")

time = x.strftime("%H:%M:%S")

print()

print("-"\*65)

print("Sales Transaction Menu --> Date:", date, " Time:", time)

print("-"\*65)

i = 0

#print header

print("No Product Desc Price Qty Disc Amount")

for item in sls.keys():

i += 1

print("%2.d. %s %-23s %6.2f %4d %6.2f %6.2f" % (

i, item, itmdict[item][0], float(itmdict[item][3]), int(sls[item]), float(itmdict[item][3])\*int(sls[item])\*(1-disc\_rate), float(itmdict[item][3])\*int(sls[item])\*disc\_rate))

print()

loop = True

while loop:

confirm = input("Confirm payment? <Y>es <N>o ")

if confirm in ["N", "n"]:

loop = False

elif confirm in ["Y", "y"]:

receipt(AssignmentList, disc\_rate, False)

loop = False

else:

print("Please enter a valid response")

#receipt

def receipt(AssignmentList,disc\_rate,is\_reprint):

x = datetime.datetime.now()

date = x.strftime("%Y-%m-%d")

time = x.strftime("%H:%M:%S")

loop = True

while loop:

confirm = input("Do you want a receipt? <Y>es <N>o ")

if confirm in ["N", "n"]:

tot = 0

for item in sls.keys():

tot += float(itmdict[item][3])\*sls[item]

disc\_amt = float(tot) \* (1 - disc\_rate)

after\_disc = float(tot) - disc\_amt

# rounding

sub\_ttl = str("{:.2f}".format(float(after\_disc)))

ad = sub\_ttl[-1]

ad = int(ad)

num = [0.00, -0.01, -0.02, -0.03, -0.04, 0.00, 0.04, 0.03, 0.02, 0.01]

adj = float(num[ad])

# if its not from reprint last receipt, save into txt file to update the quantity

if is\_reprint is False:

saveFile(AssignmentList)

# update sales amount to file

sales\_hist = []

sales\_hist.append([str(date), str(time), "{:.2f}".format(float(tot)), "{:.2f}".format(float(disc\_amt)),

"{:.2f}".format(float(float(tot) + adj - disc\_amt))])

updateFile(sales\_hist, "SalesHistory.txt")

loop = False

elif confirm in ["Y", "y"]:

tot = 0

print()

print("-"\*65)

print("Payment --> Date:", date, " Time:", time)

print("-"\*65)

i = 0

#print header

print("No Product Desc Price Qty Disc Amount")

for item in sls.keys():

i += 1

print("%2.d. %s %-23s %4.2f %4d %6.2f %5.2f"%(

i, item, itmdict[item][0], float(itmdict[item][3]), int(sls[item]), float(itmdict[item][3])\*int(sls[item])\*(1-disc\_rate), float(itmdict[item][3])\*int(sls[item])\*disc\_rate))

print("-"\*65)

tot += float(itmdict[item][3])\*sls[item]

disc\_amt = float(tot) \* (1 - disc\_rate)

after\_disc = float(tot) - disc\_amt

#rounding

sub\_ttl = str("{:.2f}".format(float(after\_disc)))

ad = sub\_ttl[-1]

ad = int(ad)

num = [0.00, -0.01, -0.02, -0.03, -0.04, 0.00, 0.04, 0.03, 0.02, 0.01]

adj = float(num[ad])

print("Sub-Total(RM) %39.2f" % tot)

print("Discount Amount(RM) %39.2f" % disc\_amt)

print("Adjusted Amount(RM) %39.2f" % adj)

print("Total(RM) %39.2f" % (float(sub\_ttl) + float(adj)))

print("Payment paid(RM) %39.2f" % (float(sub\_ttl) + float(adj)))

print("="\*65)

loop = False

pay = input("Press enter to proceed to pay <Q>uit >>")

print("Thank you")

print()

#if its not from reprint last receipt, save into txt file to update the quantity

if is\_reprint is False:

saveFile(AssignmentList)

#update sales amount to file

sales\_hist = []

sales\_hist.append([str(date), str(time), "{:.2f}".format(float(tot)), "{:.2f}".format(float(disc\_amt)), "{:.2f}".format(float(float(tot)+adj-disc\_amt))])

updateFile(sales\_hist, "SalesHistory.txt")

final()

else:

print("Please enter a valid response")

#membership maintenance

def MembershipMaint(membershiplist):

loop = True

step = 0

isEdit = False

while loop:

#step 0 select action

if step == 0:

crud\_action\_menu()

opt = input("Please select an action >> ").upper()

if opt == "Q":

loop = False

elif not opt.isdigit():

print("Please enter a valid integer")

elif int(opt) < 0 or int(opt) > 4:

print("Please enter only 1 - 4 or 'Q' to exit")

elif opt == "1":

step = 3

elif opt == "2":

displayMembership(membershiplist)

input("Press enter to continue...")

continue

elif opt == "3":

step = 1

isEdit = True

elif opt == "4":

step = 1

isEdit = False

#step 1 select membership (for edit, delete only)

if step == 1:

displayMembership(membershiplist)

sel\_membership = input("Please select Membership <Q>uit >> ")

if sel\_membership == "Q" or sel\_membership == "q":

loop = False

elif not sel\_membership.isdigit():

print("#############################")

print("# Please enter an integer #")

print("#############################")

elif int(sel\_membership) > len(membershiplist) or int(sel\_membership) < 0:

print("#############################")

print("# Invalid selection #")

print("#############################")

else:

#if pass all checking above, proceed to next step

if isEdit is True:

step = 2

else:

step = 4

#step 2 : update membership

elif step == 2:

print("-" \* 60)

print("Tier Discount Rate")

print("-" \* 60)

print("%-20.20s %4f" % (membershiplist[int(sel\_membership)-1][0], float(membershiplist[int(sel\_membership)-1][1])))

print("-" \* 60)

print("####################")

print("# Edit Menu #")

print("####################")

print("<1> Edit Tier ")

print("<2> Edit Edit Discount Rate ")

sel\_item = input("Please select item to edit <Q>uit >> ")

if sel\_item == "Q" or sel\_item == "q":

loop = False

elif not sel\_item.isdigit():

print("#############################")

print("# Please enter an integer #")

print("#############################")

elif int(sel\_item) == 1:

#editting tier name

new\_tier\_name = input("Please enter new tier name for <" + membershiplist[int(sel\_membership)-1][0] + "> or enter <Q> to exit :")

if new\_tier\_name == "Q" or new\_tier\_name == "q":

loop = False

#validation for name

if not membership\_name\_dup\_check(membershiplist, new\_tier\_name):

print("########################################")

print("# Duplicate membership tier detected #")

print("########################################")

input("Press enter to continue.....")

else:

#overwrite old data with new data

membershiplist[int(sel\_membership)-1][0] = new\_tier\_name

#update txt file

overwriteFile(membershiplist, "Membership.txt")

print("Membership updated successfully")

loop = False

elif int(sel\_item) == 2:

#editing discount rate

new\_disc\_rate = input("Please enter new discount rate for <" + membershiplist[int(sel\_membership)-1][0] + "> :")

#overwrite old data with new data

membershiplist[int(sel\_membership)-1][1] = new\_disc\_rate

#update txt file

overwriteFile(membershiplist, "membership.txt")

print("Membership updated successfully")

loop = False

else:

print("#############################")

print("# Invalid selection #")

print("#############################")

#step 3 : create new membership

elif step == 3:

add\_new\_tier = input("Please enter new Tier Name <Q>uit >>")

if add\_new\_tier == "Q" or add\_new\_tier == "q":

loop = False

elif not membership\_name\_dup\_check(membershiplist, add\_new\_tier):

print("########################################")

print("# Duplicate membership tier detected #")

print("########################################")

input("Press enter to continue.....")

else:

#proceed to enter discount rate

add\_new\_disc = input("Please enter new Discount Rate for <" + add\_new\_tier + "> or enter Q to Quit >>")

if add\_new\_disc == "Q" or add\_new\_disc == "q":

loop = False

continue

elif not add\_new\_disc.isdigit():

print("#############################")

print("# Please enter an integer #")

print("#############################")

input("Press enter to continue.....")

elif int(add\_new\_disc) <= 0 or int(add\_new\_disc) > 100:

print("#############################")

print("# Invalid Discount Rate #")

print("#############################")

input("Press enter to continue.....")

else:

#pass all checking above, proceed to add new data

membershiplist.append([add\_new\_tier, add\_new\_disc])

overwriteFile(membershiplist, "Membership.txt")

print("New membership added successfully.")

continue

#step 4 : delete membership

elif step == 4:

print("#############################")

print("# Delete Membership #")

print("#############################")

print("-" \* 60)

print("Tier Discount Rate")

print("-" \* 60)

print("%-20.20s %4f" % (

membershiplist[int(sel\_membership) - 1][0], float(membershiplist[int(sel\_membership) - 1][1])))

print("-" \* 60)

confirm\_del = input("Are you sure to delete this Membership? <Y>es / <N>o / <Q>uit >> ")

confirm\_del = confirm\_del.upper()

if confirm\_del == "Q" or confirm\_del == "N":

loop = False

elif confirm\_del != "Y" and confirm\_del != "N" and confirm\_del != "Q":

print("#############################")

print("# Invalid Response #")

print("#############################")

elif confirm\_del == "Y":

#Proceed deletion

membershiplist.remove(membershiplist[int(sel\_membership) - 1])

overwriteFile(membershiplist, "Membership.txt")

print("Membership deleted successfully")

loop = False

#new membership name checking (for duplication)

def membership\_name\_dup\_check(arr, name):

for i in range(len(arr)):

if arr[i][0] == name:

return False

return True

def final():

welcome()

while True:

main()

opt = input("Option >>")

#Stock/Inventory Management

if opt == "1":

print("\n>>Item Maintenance<<")

AssignmentList = readFile("AssignmentList.txt")

loop = True

while loop:

print()

displayRec(AssignmentList)

print("<A>dd new item <U>pdate <D>elete <S>ync File <Q>uit")

opt = input("Option >> ").upper()

if opt == "Q":

loop = False

continue

elif opt == "A" or opt == "a": #need more validation in addNewitem()

#elif opt in ["A","D","U"]:

AssignmentList = addNewitem(AssignmentList)#itmLst=maintItem(itmLst,opt)

elif opt == "D" or opt == "d":

AssignmentList = delItem(AssignmentList)

elif opt == "U" or opt == "u":

AssignmentList = updItem(AssignmentList)

elif opt == "S" or opt == "s":

saveFile(AssignmentList)

print("~AssignmentList updated~")

else:

print("Invalid option entered")

#Daily Sales History

elif opt == "2":

SalesHist = readFile("SalesHistory.txt")

displaySalesHist(SalesHist)

input("Press enter to continue...")

#Sales

elif opt == "3":

AssignmentList = readFile("AssignmentList.txt")

#for i in range(len(AssignmentList)):

# for j in range(5):

# print(AssignmentList[i][j])

membershiplist = readFile("Membership.txt")

#for i in range(len(Membership)):

# for j in range(3):

# print(Membership[i][j])

addItm(AssignmentList, membershiplist)

#Membership Maintenance

elif opt == "4":

membershiplist = readFile("Membership.txt")

MembershipMaint(membershiplist)

#Reprint Last Receipt

elif opt == "5":

if disc\_rate == 0 or disc\_rate is None:

print("###############################")

print("# No last receipt available #")

print("###############################")

else:

AssignmentList = []

receipt(AssignmentList, disc\_rate, True)

elif opt == "Q" or opt == "q":

question = input("Are you sure?(<Y>es/<N>o):")

if question == "Y" or question == "y":

print("Thank you for visiting our shop :) Have a nice day")

time.sleep(2)

exit()

if question == "N" or question == "n":

main()

continue

else:

print("Invalid option entered")

continue

else:

print("Please enter a valid option.")

final()