GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

AAPP010-4-2-PWP

PROGRAMMING WITH PYTHON

UCDF2111ICT(SE) / UCDF2111ICT(DI) / UCDF2111ICT / UCDF2111ICT(ITR) / UCDF2111BIT

HAND OUT DATE: 13th OCTOBER 2022

HAND IN DATE: 02nd DECEMBER 2022

WEIGHTAGE: 50%

	Name	TP Number
1.	LIM JIA SHENG	TP067045
2.	DENNIS CHONG KEE KIAN	TP065544

Table of Contents

1.0 INTRODUCTION

The structure of the system for the online shopping mall are broken down into four distinct sections. These sections are the functionalities of the admin, the functionalities of the delivery staff, the functionalities of all customers, including unregistered customers, and the full functionalities of registered customers.

The goal of this project is to design and create an efficient online order management system that will be referred to as (OOMS) for short. This system will be used for the APU online shopping mall (AOSM) system, which is comparable to other large E-commerce platforms like Lazada and Shopee. Using this technology, the administrator of the APU online shopping mall can easily establish new items and categories, as well as quickly handle all of the orders placed by customers and the payments they make. In the meanwhile, the architecture of the system was built in such a way that it enables customers who have already registered for an account to access the system through a login page, while customers who have not yet registered for an account may do so in order to use the system in the future. In addition to this, registered consumers of the APU online shopping mall system have the ability to purchase things from the system, while unregistered users of the system are only able to browse the various products that are for sale there. Other than that, the system also contains the delivery staff section which support the system to let our delivery staff to update the customer's order delivery status by just simply entering the customer email to update.

2.0 Assumptions of the system

1. People with access to the online order management system (OOMS) will be the admin users, delivery staff users and as well as the registered user.

2. Unregistered customer can only view the available items in our store.

3. Any user can only have access to the online order management system (OOMS) only if the login is successful.

4. Every user type has its own access to each function.

5. Once login is complete the appropriate menu will be displayed for the type of user.

6. Admin can add item, modify, and delete.

7. Admin can add delivery staff, modify, and delete

8. Only 20 ID's can be generated for admin user and delivery staff user.

9. Password of the user needed to be same in confirmation password to able register an account.

10. Delivery staff can update the delivery status of customer's order through the delivery staff system function

11. Delivery staff can only update the order status of 1 out of 3 choices which are shipped, delivered, or cancelled in an order.

12. Admin can assign order for delivery staff by simply entering the customer email that wanted to assign and select the delivery staff to deliver the order.

13. Registered customer can only add 10 items total in a cart.

14. If the items that added to cart are out of stock, the purchasing process will be automatically cancelled.

15. The payment method is only valid to Debit card and Credit card.

16. Before placing an order, the customer need to provide their email address and phone number.

17. All user can close the program by selecting exit from the main menu.

3.0 Design of the program

3.1 Pseudocode

admin_login()

```
#function of admin login page
DEFINE FUNCTION admin_login(adminList)
        DISPLAY("")
        DISPLAY(" --- ADMIN LOGIN PAGE ---")
        DEFINE username
        DISPLAY("Enter your Username: ")
        READ username
        DEFINE password
        DISPLAY("Enter your Password: ")
        READ password
        allRec = []
        ind = -1
        with open("adminlog.txt") in READ MODE as f
                LOOP rec in f
                        allRec.append(rec.strip().split(":"))
                        NEXT rec
                ENDLOOP
        endwith
        LOOP cnt in range(len(allRec))
                if (username == allRec[cnt][0]) and (password == allRec[cnt][1]) THEN
                        ind = cnt
                endif
                NEXT cnt
        ENDLOOP
        if (ind != -1) THEN
                DISPLAY("*************** ADMIN SYSTEM *************** \n")
                DISPLAY("Logged in Successfully!")
                DISPLAY("Welcome back",username)
                DISPLAY("")
                RUN admin_menu()
        else
                DISPLAY("Login failed, credential not valid")
                RUN main_menu()
        endif
```

admin_menu()

```
#function of admin menu
DEFINE FUNCTION admin_menu()
        DISPLAY(" --- ADMIN MENU ---")
        DISPLAY("1. Add item category")
        DISPLAY("2. Modify item")
        DISPLAY("3. Delete item")
        DISPLAY("4. View/Search customer order")
        DISPLAY("5. View/Search customer payment" )
        DISPLAY("6. Order Delivery Management")
        DISPLAY("7. Exit")
        DEFINE ch as INTEGER
        DISPLAY("Enter your choice: ")
        READ ch
        if ch == 1 THEN
                RUN add_item_category()
        elif ch == 2 THEN
                RUN modify_item()
        elif ch == 3 THEN
                RUN remove_item()
        elif ch == 4 THEN
                RUN view_search_cust_order()
        elif ch == 5 THEN
                RUN view_search_cust_payment()
        elif ch == 6 THEN
                RUN odm()
        elif ch == 7 THEN
                DISPLAY("End of program \n")
                RUN main_menu()
        else
                DISPLAY("Wrong input!")
                DISPLAY("Wrong choice! \n")
                RUN main_menu()
        endif
```

register_page()

```
#function of register page
DEFINE FUNCTION register()
        DISPLAY("")
        DISPLAY("--- REGISTER PAGE ---")
        DEFINE username
        DISPLAY("Enter Username: ")
        READ username
        DEFINE password
        DISPLAY("Enter password: ")
        READ password
        DEFINE password1
        DISPLAY("Confirm password: ")
        READ password1

        if password == password1 THEN

                with open("credentials.txt") in APPEND MODE as f
                        rec = username +":"+password
                        WRITE (rec +"\n") in f
                        print("You have registered successfully!")
                endwith
                RUN login()
        else
                DISPLAY("Password is not same as above! \n")
                RUN register()
        endif
```

login()

```
DEFINE FUNCTION login()
        DISPLAY("")
        DISPLAY("--- LOGIN PAGE ---")
        DEFINE username
        DISPLAY("Enter your Username: ")
        READ username
        DEFINE password
        DISPLAY("Enter your Password: ")
        READ password
        allRec = []
        ind = -1
        with open("credentials.txt") in READ MODE as f
                LOOP rec in f
                        allRec.append(rec.strip().split(":"))
                        NEXT rec
                ENDLOOP
        endwith
        LOOP cnt in range(len(allRec))
                if (username == allRec[cnt][0]) and (password == allRec[cnt][1]) THEN
                        ind = cnt
                endif
                NEXT cnt
        ENDLOOP
        if (ind != -1) THEN
                DISPLAY("")
                DISPLAY("Logged in Successfully!")
                DISPLAYt("Welcome Back," + username)
                DISPLAY("Happy Shopping!")
                DISPLAY("")
                DISPLAY(" --- Welcome to Jason and Dennis ALL IT APU Online Shopping Mall ---")
                DISPLAY("1. View all item details")
                DISPLAY("2. Place order")
                DISPLAY("3. Exit")
                DEFINE ch as INTEGER
                DISPLAY("Enter your choice: ")
                READ ch
                if ch == 1 THEN
                        with open("item_Details.txt") in READ MODE as f
                                DISPLAY(f.read())
                                DEFINE placeorder
                                DISPLAY("Do you want to place an order? (yes/no): ")
                                READ placeorder
                                if placeorder == "yes" THEN
                                        RUN place_order()
                                elif placeorder == "no" THEN
                                        DISPLAY("Back to main menu!")
                                        RUN main_menu()
                                else
                                        DISPLAY("Wrong Input!")
                                        RUN main_menu()
                                endif

                elif ch == 2 THEN
                        RUN place_order()
                elif ch == 3 THEN
                        DISPLAY("End of program \n")
                        RUN main_menu()
                else:
                        DISPLAY("Wrong input!")
                        RUN main_menu()
                endif
        else
                DISPLAY("Login failed \n")
                RUN main_menu()
        endif
```

view_all_items()

```
#function of view all items page and access to register page
DEFINE FUNCTION view_all_items()
        DISPLAY("")
        DISPLAY(" --- VIEW ALL ITEM PAGE ---")
        DISPLAY("")
        OPEN ("Category_Details.txt") in READ MODE as f
        DISPLAY(f.read())
        f.close()
        DISPLAY("")
        OPEN ("All_items.txt") in READ MODE as f
        DISPLAY(f.read())
        f.close()
        DISPLAY("")
        DISPLAY("1.Registered")
        DISPLAY("2.Unregistered")
        DISPLAY("3.Back")
        DISPLAY("")
        DEFINE ch as INTEGER
        DISPLAY("Please enter your choice:")
        READ ch
        if ch == 1 THEN
                RUN login()
        elif ch == 2 THEN
                DISPLAY("")
                DISPLAY("1. Register to access more details")
                DISPLAY("2. Exit")
                DEFINE ch as INTEGER
                DISPLAY("Please enter your choice:")
                READ ch
                if ch == 1 THEN
                        RUN register()
                else
                        DISPLAY("Exit to main menu")
                        RUN main_menu()

        elif ch == 3 THEN
                DISPLAY("Back to main menu")
                RUN main_menu()
        else
                DISPLAY("Wrong Input \n")
                RUN main_menu()
        endif
```

place_order()

```
#function of allow registered user to place order
DEFINE FUNCTION place_order()
        shoppingDict as DICTIONARY

        #read data from text file
        OPEN ("All_items.txt") as my_file
        file_line = my_file.readline()
        itemsAvailable = my_file.readlines()
        my_file.close()

        DEFINE email
        DISPLAY("Enter your email address:")
        READ email
        DEFINE phone_no as INTEGER
        DISPLAY("Enter your phone number:")
        READ phone_no
        DISPLAY("")
        DISPLAY("********** Items Available in Our Store ********** \n")
        DISPLAY("Item Code: Item Name: Item Type: Item Quantity: Item Price")
        LOOP item in itemsAvailable
                itmCode = item.split(":")[0]
                itmName = item.split(":")[1]
                itmType = item.split(":")[2]
                itmQuantity = item.split(":")[3]
                itmPrice = item.split(":")[4]
                NEXT item
        ENDLOOP

                DISPLAY(f"{itmCode}: {itmName}: {itmType}: {itmQuantity}: {itmPrice}")


        allRec = []
        with open("All_items.txt") in READ MODE as f
                LOOP rec in f
                        allRec.append(rec.strip().split(":"))
                        NEXT rec
                ENDLOOP
        endwith
        DEFINE proceedShopping
        DISPLAY("Do you wish to proceed (y/n): ").lower()
        READ proceedShopping
        DOWHILE (proceedShopping == "y") THEN
                DEFINE item_added as INTEGER
                DISPLAY("Enter item code to place order: ")
                READ item_added
                ind = -1
                LOOP cnt in range(len(allRec))
                        if (item_added in allRec[cnt][0]) THEN
                                ind = cnt
                        endif
                        NEXT cnt
                ENDLOOP

                if (ind != -1) THEN
                        DEFINE item_qty as INTEGER
                        DISPLAY(allRec[ind][0] +":"+ allRec[ind][1] +":"+ allRec[ind][2] +":"+ allRec[ind][3] +":"+ allRec[ind][4] +" >"+ " Enter quantity: ")
                        READ item_qty
                shoppingDict.update({"Item: "+item_added:{"quantity":item_qty,"subtotal(RM)":float(allRec[ind][4])*item_qty}})
                DISPLAY(shoppingDict)
                        ind1 = -1
                        LOOP cnt1 in range(len(allRec))
                                if (item_qty <= int(allRec[ind][3])) THEN
                                        ind1 = cnt1
                                endif
                                NEXT cnt1
                        ENDLOOP
```

```
        if (ind1 != -1) THEN
                DEFINE ori_qty as allRec[ind][3]
                DEFINE new_qty as INTEGER(ori_qty) - item_qty
                DEFINE allRec[ind][3] as STRING(new_qty)
                with open("All_items.txt") in WRITE MODE as f
                        LOOP recList in allRec
                                rec = ":".join(recList) + "\n"
                                WRITE (rec) in f
                                NEXT recList
                        ENDLOOP
                        DISPLAY("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ADD CART SUCCESSFUL~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
                endwith
        else
                DISPLAY("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~FAIL TO ADD CART~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
                DISPLAY("Item stock quantity not enough!!")
                DISPLAY("Please re-enter the quantity of product needed according to our stock quantity.")
                DISPLAY("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~PLEASE PLACE ORDER AGAIN~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
                RUN place_order()
        endif
else
        DISPLAY("Item not found. Please try again.")
        DISPLAY("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~GOING BACK ORDER PAGE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        RUN place_order()
endif
DEFINE proceed_Shopping
DISPLAY("Do you wish to add more items(y/n): ")
READ proceed_Shopping.lower()
if (proceed_Shopping == "y") THEN
        continue
else:
        DISPLAY("\n\n")
        DISPLAY("********************************Bill Summary********************************")
        DISPLAY("***************JASON AND DENNIS ALL IT APU ONLINE SHOPPING MALL***************")
        DISPLAY("Item\t\t\t\tQuantity\t\t\tSubtotal")
        DEFINE total as 0
        LOOP record in shoppingDict
                DISPLAY(f"{record}\t\t\t{shoppingDict[record]['quantity']}\t\t\t{shoppingDict[record]['subtotal(RM)']}")
                DEFINE total as shoppingDict[record]['subtotal(RM)']+total
                DISPLAY(f"Total: {total}")
                NEXT record
        ENDLOOP
        DEFINE proceedpayment
        DISPLAY("Do u wish to proceed payment (y/n):")
        READ proceedpayment
        if proceedpayment.lower() == "y" THEN
                DEFINE payment_method
                DISPLAY("Enter your payment method (Debit/Credit) card:")
                READ payment_method
                DEFINE card_no
                DISPLAY("Enter your " + payment_method + " card number:")
                READ card_no
                DEFINE cfm_payment
                DISPLAY("Comfirm payment? (yes/no) :")
                READ cfm_payment
                if cfm_payment.lower() == "yes" THEN
                        with open("cust_order.txt") in APPEND MODE as f
                                rec = email + "/" + phone_no + "/" + str(shoppingDict) + "/"
                                WRITE (rec + "\n") in f
                        endwith
                        cfm_payment = "paid" + ", " + str(datetime.datetime.now())
                        with open("cust_payment.txt", "a") as f
                                info = email+", "+phone_no+", "+ str(total)+", "+payment_method+", "+str(card_no)+", "+ cfm_payment
                                f.write(info + "\n")
                        endwith

                        DISPLAY("")
                        DISPLAY("*********** Thank You ***********")
                        DISPLAY("Hope to see you back soon!")
                        RUN main_menu()
                else
                        DISPLAY("Exit to main menu")
                        RUN main_menu()
                endif

        else:
                DISPLAY("Back to main menu")
                RUN main_menu()
        endif
        break
```

delivery_staff_login()

```
#function of delivery staff login page
DEFINE FUNCTION delivery_staff_login()
        DISPLAY("")
        DISPLAY(" --- LOGIN PAGE FOR DELIVERY STAFF ---")
        DEFINE username
        DISPLAY("Enter your username:")
        READ username
        DEFINE password
        DISPLAY("Enter your password:")
        READ password
        allRec = []
        ind = -1
        with open("stafflog.txt") in READ MODE as f
                LOOP rec in f
                        allRec.append(rec.strip().split(":"))
                        NEXT rec
                ENDLOOP
        endwith
        LOOP cnt in range(len(allRec))
                if (username == allRec[cnt][0]) and (password == allRec[cnt][1]) THEN
                        ind = cnt
                endif
                NEXT cnt
        ENDLOOP
        if (ind != -1) THEN
                DISPLAY("Logged in Successfully!")
                DISPLAY("Welcome back", username)
                RUN delivery_staff_system()
        else
                DISPLAY("Login failed, credential not valid")
                RUN main_menu()
        endif
```

add_item_category()

```
#function of allow admin to add items in inventory
DEFINE FUNCTION add_item_category()
        with open("All_items.txt") in READ MODE as f
                DISPLAY("")
                DISPLAY(" " + "--- ITEM LIST ---")
                DISPLAY("")
                DISPLAY("Code : Item : Type : Quantity : Price(RM) \n")
                DISPLAY(f.read())
        endwith
        with open("All_items.txt") in APPEND MODE as f
                DISPLAY(" --- ADD ITEM CATEGORY ---")
                DEFINE itmCode
                DISPLAY("Please enter new ITEM CODE:")
                READ itmCode
                DEFINE itmName
                DISPLAY("Please enter new ITEM NAME:")
                READ itmName
                DEFINE itmType
                DISPLAY("Please enter new ITEM TYPE (Electronic Gadget/Home Appliance):")
                READ itmType
                DEFINE itmQuantity
                DISPLAY("Please enter new ITEM QUANTITY:")
                READ itmQuantity
                DEFINE itmPrice
                DISPLAY("Please enter new ITEM PRICE:")
                READ itmPrice
                DEFINE rec as itmCode+":"+itmName+":"+itmType+":"+itmQuantity+":"+itmPrice
                WRITE (rec + "\n")in f
                DISPLAY(rec + " " + "has been successfully added")
        endwith
        RUN admin_menu()
```

modify_item()

```
#function of allow admin to modify items in inventory
DEFINE FUNCTION modify_item()
        with open("All_items.txt") in READ MODE as f
                DISPLAY("")
                DISPLAY(" " + "--- ITEM LIST ---")
                DISPLAY("")
                DISPLAY("Code : Item : Type : Quantity : Price(RM) \n")
                DISPLAY(f.read())
        endwith

        allRec = []
        with open("All_items.txt") in READ MODE as f
                LOOP rec in f
                        allRec.append(rec.strip().split(":"))
                        NEXT rec
                ENDLOOP
        endwith

        DISPLAY(" --- MODIFY ITEM ---")
        DEFINE itmCode
        DISPLAY("Please enter the ITEM CODE to modify: ")
        READ itmCode
        ind = -1
        LOOP cnt in range(len(allRec))
                if (itmCode == allRec[cnt][0]) THEN
                        ind = cnt
                endif
                NEXT cnt
        ENDLOOP
        if (ind != -1) THEN
                DEFINE newName
                DISPLAY(allRec[ind][1] + " Please enter new name: ")
                READ newName
                DEFINE newType
                DISPLAY(allRec[ind][2] + "Please enter new type(Electronic Gadget/Home Appliance: ")
                READ newType
                DEFINE newQuantity
                DISPLAY(allRec[ind][3] + "Please enter new quantity: ")
                READ newQuantity
                DEFINE newPrice
                DISPLAY(allRec[ind][4] + "Please enter new price: ")
                READ newPrice
                DEFINE allRec[ind][1] as newName
                DEFINE allRec[ind][2] as newType
                DEFINE allRec[ind][3] as newQuantity
                DEFINE allRec[ind][4] as newPrice
                DISPLAY("Modify Successfully")
        else:
                DISPLAY("Record Not Found")
                RUN admin_menu()
        endif

        with open("All_items.txt") in READ MODE as f
                LOOP recList in allRec
                        rec = ":".join(recList) + "\n"
                        WRITE (rec) in f
                ENDLOOP
        endwith
        RUN admin_menu()
```

remove_item()

```
#function of allow admin to remove items in inventory
DEFINE FUNCTION remove_item()
        with open("All_items.txt") in READ MODE as f
                DISPLAY("")
                DISPLAY(" " + "--- ITEM LIST ---")
                DISPLAY("")
                DISPLAY(f.read())
        endwith

        allRec = []
        with open("All_items.txt") in READ MODE as f
                LOOP rec in f
                        allRec.append(rec.strip().split(":"))
                        NEXT rec
                ENDLOOP
        endwith
        DISPLAY(" --- REMOVE ITEM ---")
        DEFINE itmCode
        DISPLAY("Please enter item code to delete: ")
        READ itmCode
        ind = -1
        LOOP cnt in range(len(allRec)):
                if (itmCode == allRec[cnt][0]) THEN
                        ind = cnt
                endif
                NEXT cnt
        ENDLOOP
        if (ind != -1) THEN
                DISPLAY("Item Code: "+ allRec[ind][0])
                DISPLAY("Item Name: "+ allRec[ind][1])
                DISPLAY("Item Type: "+ allRec[ind][2])
                DISPLAY("Item Quantity: "+ allRec[ind][3])
                DISPLAY("Item Price: "+ allRec[ind][4])
                DEFINE ch
                DISPLAY("Are you sure to delete this item? (y/n): ")
                READ ch
                if (ch == "y") THEN
                        DISPLAY(rec + "has been removed successfully")
                        with open("All_items.txt") in WRITE MODE as f
                                LOOP cnt in range(len(allRec))
                                        if (cnt != ind) THEN
                                                rec = ":".join(allRec[cnt]) + "\n"
                                                WRITE (rec) in f
                                        endif
                                ENDLOOP
                        endwith
                        RUN admin_menu()
                else:
                        DISPLAY("Back to admin menu")
                        RUN admin_menu()
                endif

        else:
                DISPLAY("Record Not Found")
                RUN admin_menu()
        endif
```

view_search_cust_order()

```
#function of menu for view and search customer order
  DEFINE FUNCTION view_search_cust_order()
          DISPLAY("")
          DISPLAY("1. View customer order history")
          DISPLAY("2. Search customer order history")
          DISPLAY("3. Back")
          DEFINE ch
          DISPLAY("Enter your choice:")
          READ ch
          if ch == 1 THEN
               with open("cust_order.txt") in READ MODE as f
                    DISPLAY("")
                    DISPLAY(f.read())
                    DISPLAY("1. Back to admin menu")
                    DISPLAY("2. Exit to main menu")
                    DEFINE ch
                    DISPLAY("Enter your choice: ")
                    READ ch
                    if ch == 1 THEN
                        RUN admin_menu()
                    elif ch == 2 THEN
                        RUN main_menu()
                    else
                        DISPLAY("Wrong Input")
                        RUN main_menu()
                    endif
               endwith

          elif ch == 2 THEN
               RUN search_cust_order()
          elif ch == 3 THEN
               DISPLAY("")
               DISPLAY("Back to admin menu")
               RUN admin_menu()
          else
               DISPLAY("Wrong choice!")
               RUN admin_menu()
          endif
```

search_cust_order()

```
#function of search customer order
DEFINE FUNCTION search_cust_order()
        with open("cust_order.txt") in READ MODE as f
             DISPLAY("")
             DISPLAY(f.read())
        endwith

        allRec = []
        with open("cust_order.txt") in READ MODE as f
             LOOP rec in f
                 allRec.append(rec.strip().split("/"))
                 NEXT rec
             ENDLOOP
        endwith
        DEFINE search
        DISPLAY("Enter email to search customer order: ")
        READ search
        ind = -1
        LOOP cnt in range(len(allRec))
            if (search == allRec[cnt][0]) THEN
                ind = cnt
            endif
            NEXT cnt
        ENDLOOP
        if (ind != -1) THEN
            DISPLAY("Email: " + allRec[ind][o])
            DISPLAY("Phone No: " + allRec[ind][1])
            DISPLAY("Order: " + allRec[ind][2])
            RUN admin_menu()
        else
            DISPLAY("Record not found!")
            RUN admin_menu()
        endif
```

view_search_cust_payment()

```
DEFINE FUNCTION view_search_cust_payment()
       DISPLAY("")
       DISPLAY("1. View customer payment history")
       DISPLAY("2. Search customer payment history")
       DISPLAY("3. Back")
       DEFINE ch
       DISPLAY("Enter your choice:"))
       READ ch
       if ch == 1 THEN
          with open("cust_payment.txt") in READ MODE as f
               DISPLAY("")
               DISPLAY(f.read())
          endwith
       elif ch == 2 THEN
          with open("cust_payment.txt") in READ MODE as f
               DISPLAY("")
               DISPLAY(f.read())
          endwith

          allRec = []
          with open ("cust_payment.txt") in READ MODE as f
               LOOP rec in f
                   allRec.append(rec.strip().split(","))
                   NEXT rec
               ENDLOOP
          endwith
          DEFINE search
          DISPLAY("Enter email to search customer payment: ")
          READ search
          ind = -1
          LOOP cnt in range(len(allRec))
              if (search == allRec[cnt][0]) THEN
                  ind = cnt
              endif
              NEXT cnt
          ENDLOOP
          if (ind != -1) THEN
               DISPLAY("Email: " + allRec[ind][0])
               DISPLAY("Phone No: " + allRec[ind][1])
               DISPLAY("Total Amount(RM): " + allRec[ind][2])
               DISPLAY("Payment Method: " + allRec[ind][3])
               DISPLAY("Card No: " + allRec[ind][4])
               DISPLAY("Payment Status: " + allRec[ind][5])
               DISPLAY("Payment Record Date and Time: " + allRec[ind][6])
          else
               DISPLAY("Record not found!")
               RUN view_search_cust_payment()
          endif
       elif ch == 3 THEN
            DISPLAY("Back to admin menu")
            RUN admin_menu()
       else
            DISPLAY("Wrong choice!")
            RUN admin_menu()
       endif
       RUN view_search_cust_payment()
```

delivery_staff_system()

```
#function of delivery staff system
DEFINE FUNCTION delivery_staff_system()
        DISPLAY("")
        DISPLAY(" --- DELIVERY STAFF SYSTEM ---")
        with open("cust_order.txt") in READ MODE as f
             DISPLAY("")
             DISPLAY(f.read())
        endwith

        allRec = []
        with open("cust_order.txt") in READ MODE as f
             LOOP rec in f
                 allRec.append(rec.strip().split("/"))
                 NEXT rec
             ENDLOOP
        endwith
        DISPLAY(" --- SELECT CUSTOMER ORDER TO UPDATE DELIVERY STATUS ---")
        DEFINE cust_email
        DISPLAY("Enter customer email to select: ")
        READ cust_email
        ind = -1
        LOOP cnt in range(len(allRec))
             if (cust_email == allRec[cnt][0]) THEN
                 ind = cnt
             endif
             NEXT cnt
        ENDLOOP

        if (ind != -1) THEN
            DEFINE delivery_status
            DISPLAY("Do you sure you want to update the delivery status? (yes/no): ")
            READ delivery_status
            if delivery_status.lower() == "yes" THEN
                DEFINE ch
                DISPLAY(allRec[ind][3] + "Enter number to update the following status (1.Shipped, 2.Delivered 3.Cancelled): ")
                READ ch
                if ch == 1 THEN
                    ch = "Shipped"
                    allRec[ind][3] = ch
                    DISPLAY("Updated Successfully")
                elif ch == 2 THEN
                    ch = "Delivered"
                    allRec[ind][3] = ch
                    DISPLAY("Updated Successfully")
                elif ch == 3 THEN
                    ch = "Cancelled"
                    allRec[ind][3] = ch
                    DISPLAY("Updated Successfully")
                else
                    DISPLAY("Wrong Input! Please enter number 1 up to 3 to update following status")
                    RUN delivery_staff_system()
                endif
            elif delivery_status.lower() == "no" THEN
                RUN main_menu()
            else
                DISPLAY("Wrong Input!")
                RUN main_menu()
            endif


        else
            DISPLAY("Record Not Found")
        endif

        with open("cust_order.txt") in WRITE MODE as f
                LOOP recList in allRec
                    rec = "/".join(recList) + "\n"
                    WRITE rec in f
                ENDLOOP
        endwith
```

odm()

```
#function of order delivery management system
DEFINE FUNCTION odm()
        DISPLAY("")
        DISPLAY(" --- ORDER DELIVERY MANAGEMENT SYSTEM --- \n")
        DISPLAY("1. Add delivery staff")
        DISPLAY("2. Modify delivery staff")
        DISPLAY("3. Delete delivery staff")
        DISPLAY("4. Assign orders to delivery staff")
        DISPLAY("5. Exit")
        DEFINE ch
        DISPLAY("Enter your choice: ")
        READ ch
        if ch == 1 THEN
            with open("stafflog.txt") in READ MODE as f
                 DISPLAY("")
                 DISPLAY(" --- DELIVERY STAFF LIST ---")
                 DISPLAY("")
                 DISPLAY("ID-PASSWORD-NAME \n")
                 DISPLAY(f.read())
            endwith
            with open("stafflog.txt") in APPEND MODE as f
                 DISPLAY(" --- ADD DELIVERY STAFF ---")
                 DEFINE staffid
                 DISPLAY("Please enter new delivery staff ID:")
                 READ staffid
                 DEFINE staffpass
                 DISPLAY("Please enter new delivery staff PASSWORD:")
                 READ staffpass
                 DEFINE staffname
                 DISPLAY("Please enter new delivery staff NAME:")
                 READ staffname
                 rec = staffid + ":" + staffpass + ":" + staffname
                 WRITE (rec + "\n") in f
                 DISPLAY(rec + " " + "has been successfully added")
            endwith
            RUN odm()

        elif ch == 2 THEN
            with open("stafflog.txt") in READ MODE as f
                 DISPLAY("")
                 DISPLAY(" --- DELIVERY STAFF LIST ---")
                 DISPLAY("")
                 DISPLAY("ID-PASSWORD-NAME \n")
                 DISPLAY(f.read())
            endwith
            allRec = []
            with open("stafflog.txt") in READ MODE as f
                 LOOP rec in f
                     allRec.append(rec.strip().split(":"))
                     NEXT rec
                 ENDLOOP
            endwith
            DISPLAY(" --- MODIFY DELIVERY STAFF ---")
            DEFINE staffid
            DISPLAY("Please enter the staff ID to modify: ")
            READ staffid
            ind = -1
            LOOP cnt in range(len(allRec))
                 if (staffid == allRec[cnt][0]) then
                     ind = cnt
                 endif
                 NEXT cnt
            ENDLOOP
```

```
    if (ind != -1) THEN
         DEFINE newId
         DISPLAY(allRec[ind][0] + " Please enter new ID: ")
         READ newId
         DEFINE newPass
         DISPLAY(allRec[ind][1] + "Please enter new PASSWORD: ")
         READ newPass
         DEFINE newName
         DISPLAY(allRec[ind][2] + "Please enter new NAME: ")
         READ newName
         allRec[ind][0] = newId
         allRec[ind][1] = newPass
         allRec[ind][2] = newName + ":"
         DISPLAY("Modify Successfully")
    else
         DISPLAY("Record Not Found")
         RUN odm()
    endif

    with open("stafflog.txt") in WRITE MODE as f
         LOOP recList in allRec:
             rec = ":".join(recList) + "\n"
             WRITE rec in f
         ENDLOOP
    endwith
    RUN odm()


elif ch == 3 THEN
    with open("stafflog.txt") in READ MODE as f
         DISPLAY("")
         DISPLAY(" --- DELIVERY STAFF LIST ---")
         DISPLAY("")
         DISPLAY("ID-PASSWORD-NAME \n")
         DISPLAY(f.read())
    endwith

    allRec = []
    with open("stafflog.txt") in READ MODE as f
         LOOP rec in f
             allRec.append(rec.strip().split(":"))
         ENDLOOP
    endwith
    DISPLAY(" --- REMOVE DELIVERY STAFF ---")
    DEFINE staffid
    DISPLAY("Please enter staff ID to delete: ")
    READ staffid
    ind = -1
    LOOP cnt in range(len(allRec))
         if (staffid == allRec[cnt][0]) THEN
             ind = cnt
         endif
         NEXT cnt
    ENDLOOP
```

```
    if (ind != -1) THEN
         DISPLAY("Staff ID: "+ allRec[ind][0])
         DISPLAY("Staff Password: "+ allRec[ind][1])
         DISPLAY("Staff Name: "+ allRec[ind][2])
         DEFINE ch
         DISPLAY("Are you sure to delete this item? (y/n): ")
         READ ch
         if (ch == "y") THEN
             DISPLAY(rec + "has been removed successfully")
             with open("stafflog.txt") in WRITE MODE as f
                  LOOP cnt in range(len(allRec)):
                      if (cnt != ind) THEN
                          rec = ":".join(allRec[cnt]) + "\n"
                          WRITE rec in f
                      endif
                  ENDLOOP
             endwith
    else
         DISPLAY("Record Not Found")
         RUN odm()
    endif
    RUN odm()


elif ch == 4 THEN
         DISPLAY("")
         DISPLAY(" ********** ASSIGN ORDER TO DELIVERY STAFF SYSTEM **********")
         with open("stafflog.txt") in READ MODE as f
             DISPLAY("")
             DISPLAY(" --- DELIVERY STAFF LIST ---")
             DISPLAY("ID-PASSWORD-NAME-CUSTOMER EMAIL")
             DISPLAY(f.read())
         endwith

         allRec = []
         with open("stafflog.txt") in READ MODE as f
              LOOP rec in f
                  allRec.append(rec.strip().split(":"))
              ENDLOOP
         endwith
         DISPLAY(" --- SELECT STAFF ID TO ASSIGN ORDER FOR DELIVER ---")
         DEFINE staffid
         DISPLAY("Enter staff ID to select: ")
         READ staffid
         ind = -1
         LOOP cnt in range(len(allRec))
             if (staffid == allRec[cnt][0]) THEN
                 ind = cnt
             endif
             NEXT cnt
         ENDLOOP
```

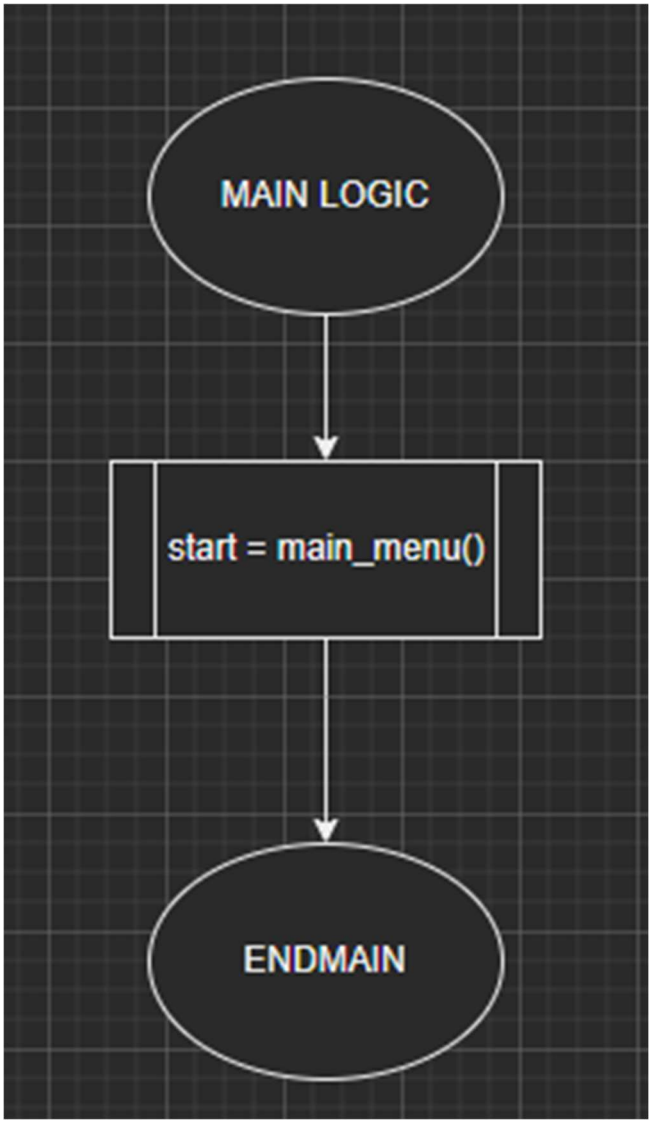
```
            if (ind != -1) THEN
                with open("cust_order.txt") in READ MODE as f
                      DISPLAY("")
                      DISPLAY(" === CUSTOMER ORDER LIST ---")
                      DISPLAY(f.read())
                      DEFINE cust_email
                      DISPLAY(allRec[ind][3] + "Enter customer email to assign deliver order: ")
                      READ cust_email
                      allRec[ind][3] = cust_email
                      DISPLAY("Updated Successfully")
                endwith
            else
                DISPLAY("Record Not Found")
                RUN odm()
            endif

            with open("stafflog.txt") in WRITE MODE as f
                      LOOP recList in allRec:
                          rec = ":".join(recList) + "\n"
                          WRITE rec in f
                      ENDLOOP
            endwith
            RUN odm()

elif ch ==5 THEN
            DISPLAY("Back to admin menu")
            RUN admin_menu()
else
    DISPLAY("Wrong Input!")
    RUN admin_menu()
endif
```

main_menu()

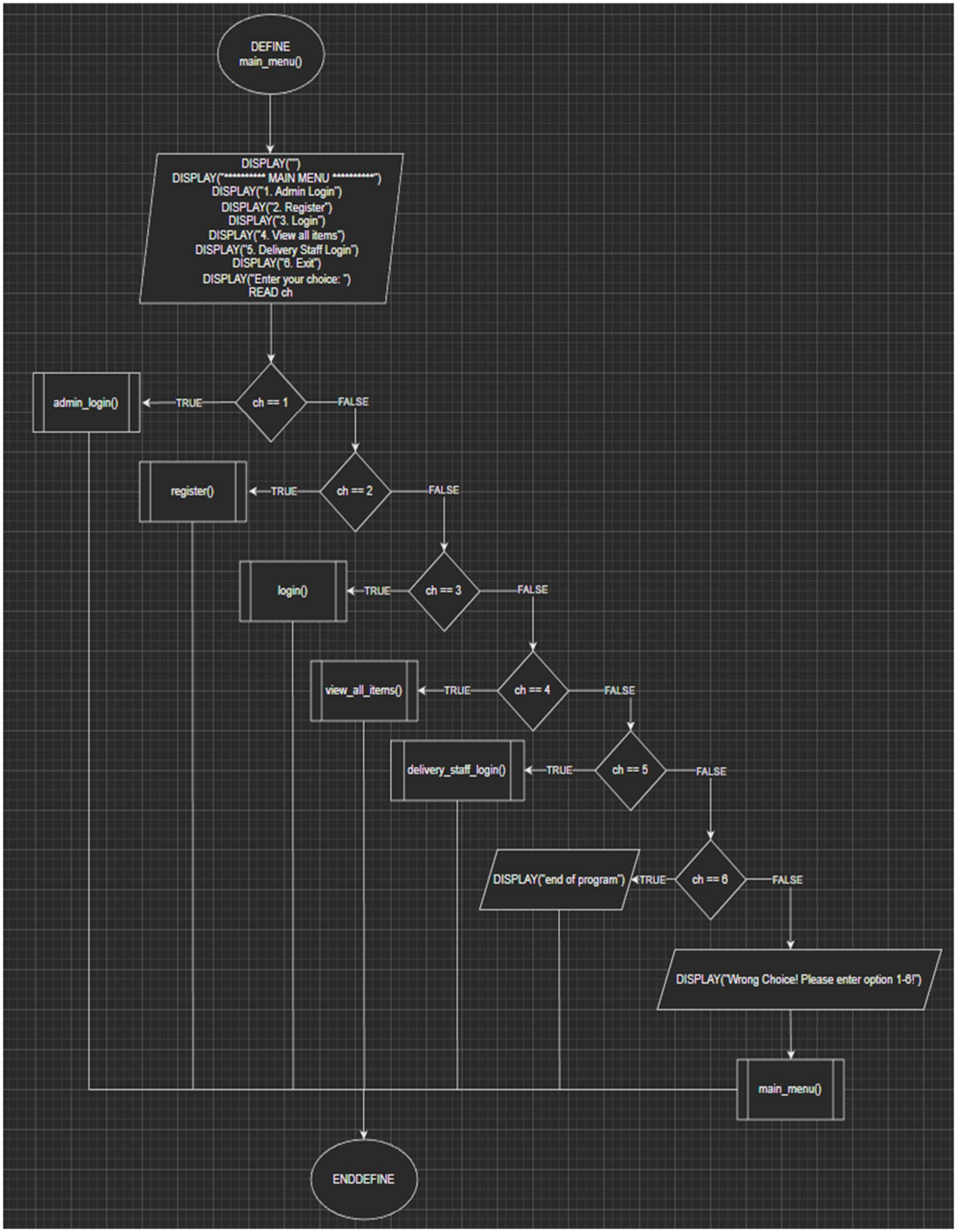
```
#function of main menu
DEFINE FUNCTION main_menu()
        DISPLAY("")
        DISPLAY("********** MAIN MENU **********")
        DISPLAY("1. Admin Login")
        DISPLAY("2. Register")
        DISPLAY("3. Login")
        DISPLAY("4. View all items")
        DISPLAY("5. Delivery Staff Login")
        DISPLAY("6. Exit")
        DEFINE ch
        DISPLAY("Enter your choice: "))
        READ ch
        if ch == 1 THEN
            RUN admin_login(adminList)
        elif ch == 2 THEN
            RUN register()
        elif ch == 3 THEN
            RUN login()
        elif ch == 4 THEN
            RUN view_all_items()
        elif ch == 5 THEN
            RUN delivery_staff_login()
        elif ch == 6 THEN
            DISPLAY("end of program")
        else
            DISPLAY("Wrong Choice! Please enter option 1-6!")
            RUN main_menu()
        endif


DOWHILE (1)
        RUN main_menu()
        break
ENDWHILE
```

3.2 Flowchart

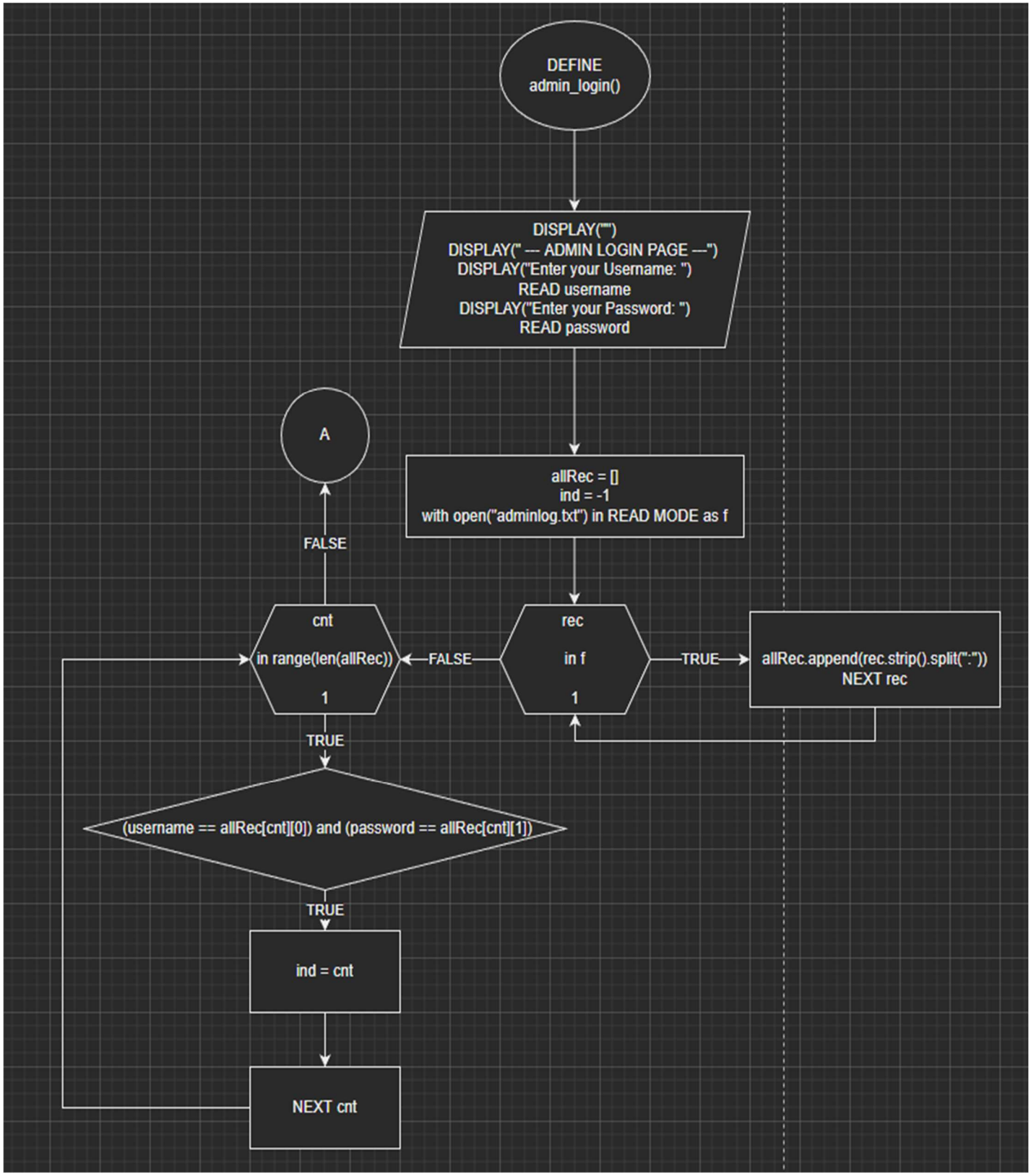
Main logic

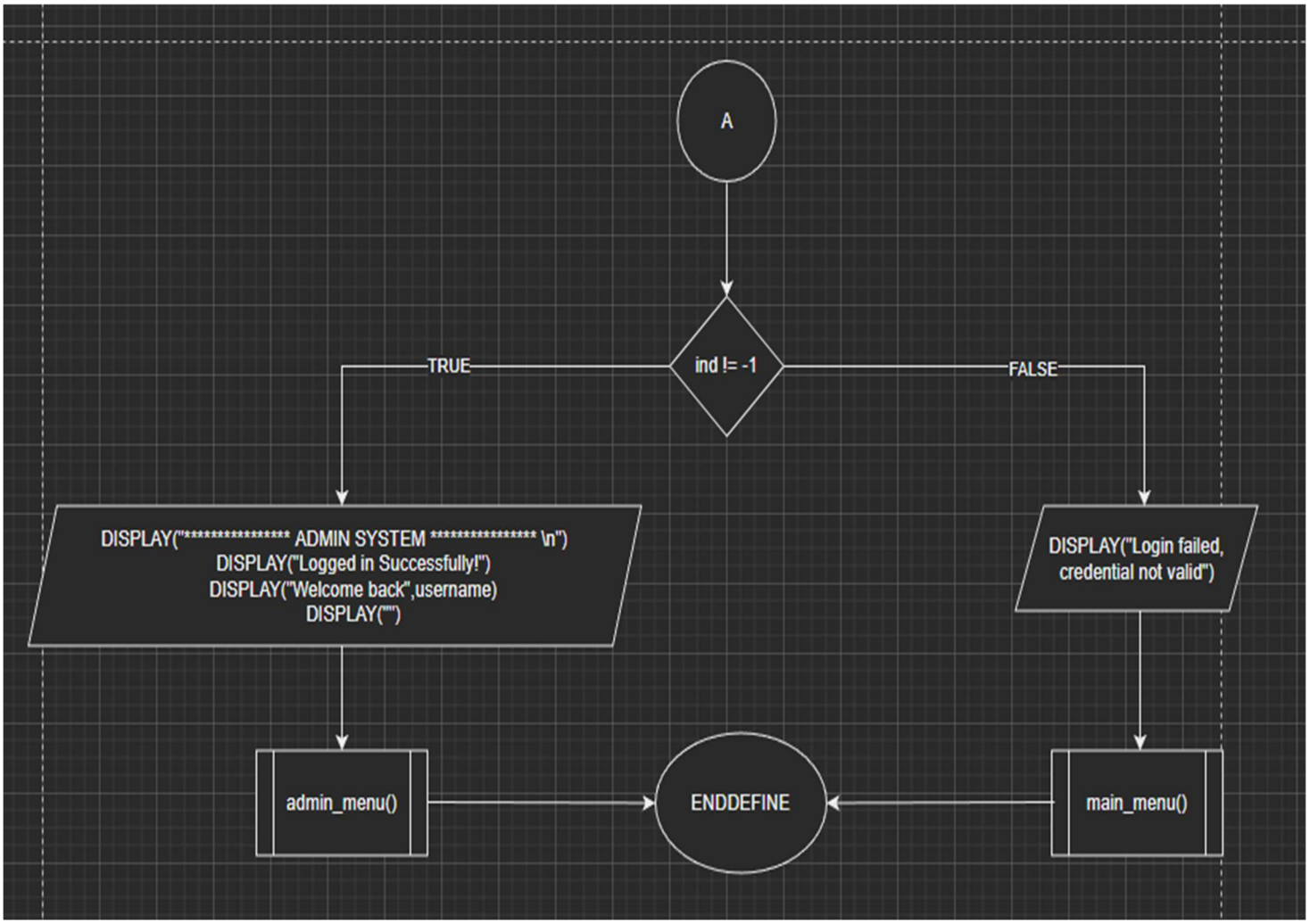


Main_menu()

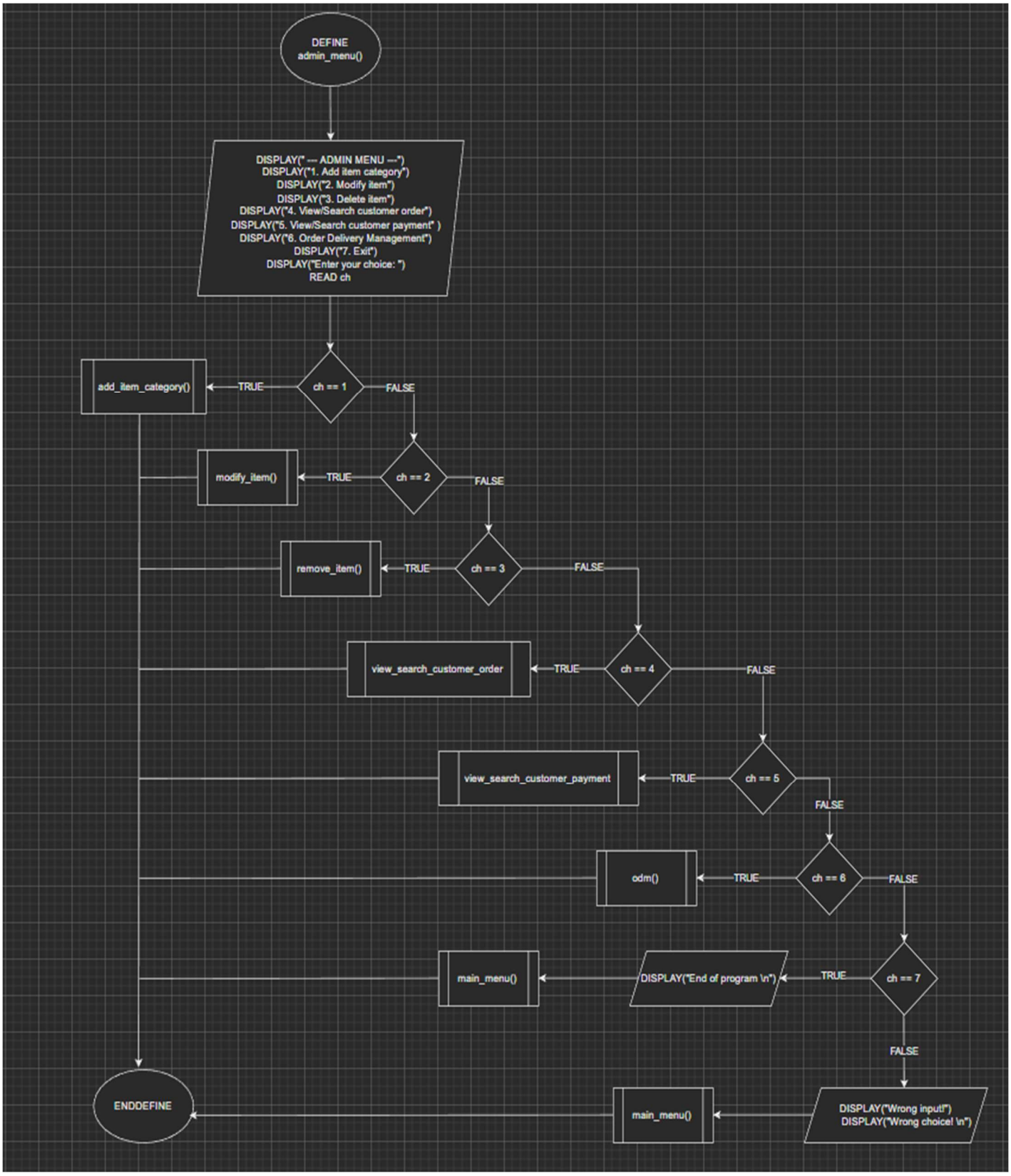

DEFINE
main_menu()
DISPLAY("")
DISPLAY("********** MAIN MENU **********")
DISPLAY("1. Admin Login")
DISPLAY("2. Register")
DISPLAY("3. Login")
DISPLAY("4. View all items")
DISPLAY("5. Delivery Staff Login")
DISPLAY("6. Exit")
DISPLAY("Enter your choice: ")
READ ch
ch == 1
TRUE
admin_login()
FALSE
ch == 2
TRUE
register()
FALSE
ch == 3
TRUE
login()
FALSE
ch == 4
TRUE
view_all_items()
FALSE
ch == 5
TRUE
delivery_staff_login()
FALSE
ch == 6
TRUE
DISPLAY("end of program")
FALSE
DISPLAY("Wrong Choice! Please enter option 1-6!")
main_menu()
ENDDEFINE

Admin_login()

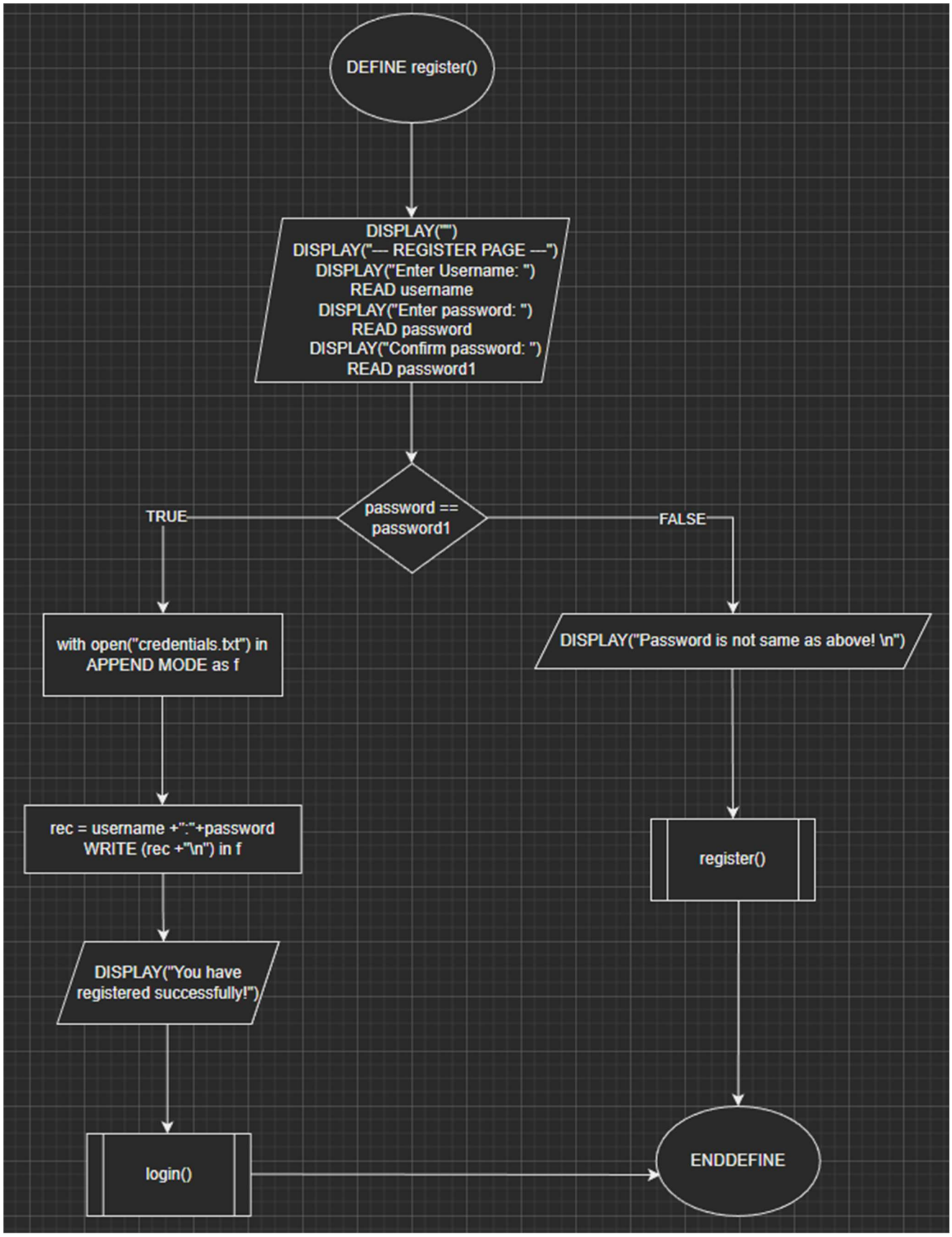

DEFINE
admin_login()
DISPLAY("")
DISPLAY(" — ADMIN LOGIN PAGE —")
DISPLAY("Enter your Username: ")
READ username
DISPLAY("Enter your Password: ")
READ password
A
allRec = []
ind = -1
with open("adminlog.txt") in READ MODE as f
FALSE
cnt
in range(len(allRec))
1
FALSE
rec
in f
1
TRUE
allRec.append(rec.strip().split(":"))
NEXT rec
TRUE
(username == allRec[cnt][0]) and (password == allRec[cnt][1])
TRUE
ind = cnt
NEXT cnt


A
ind != -1
TRUE
FALSE
DISPLAY("********** ADMIN SYSTEM ********** \n")
DISPLAY("Logged in Successfully!")
DISPLAY("Welcome back",username)
DISPLAY("")
DISPLAY("Login failed, credential not valid")
admin_menu()
ENDDEFINE
main_menu()

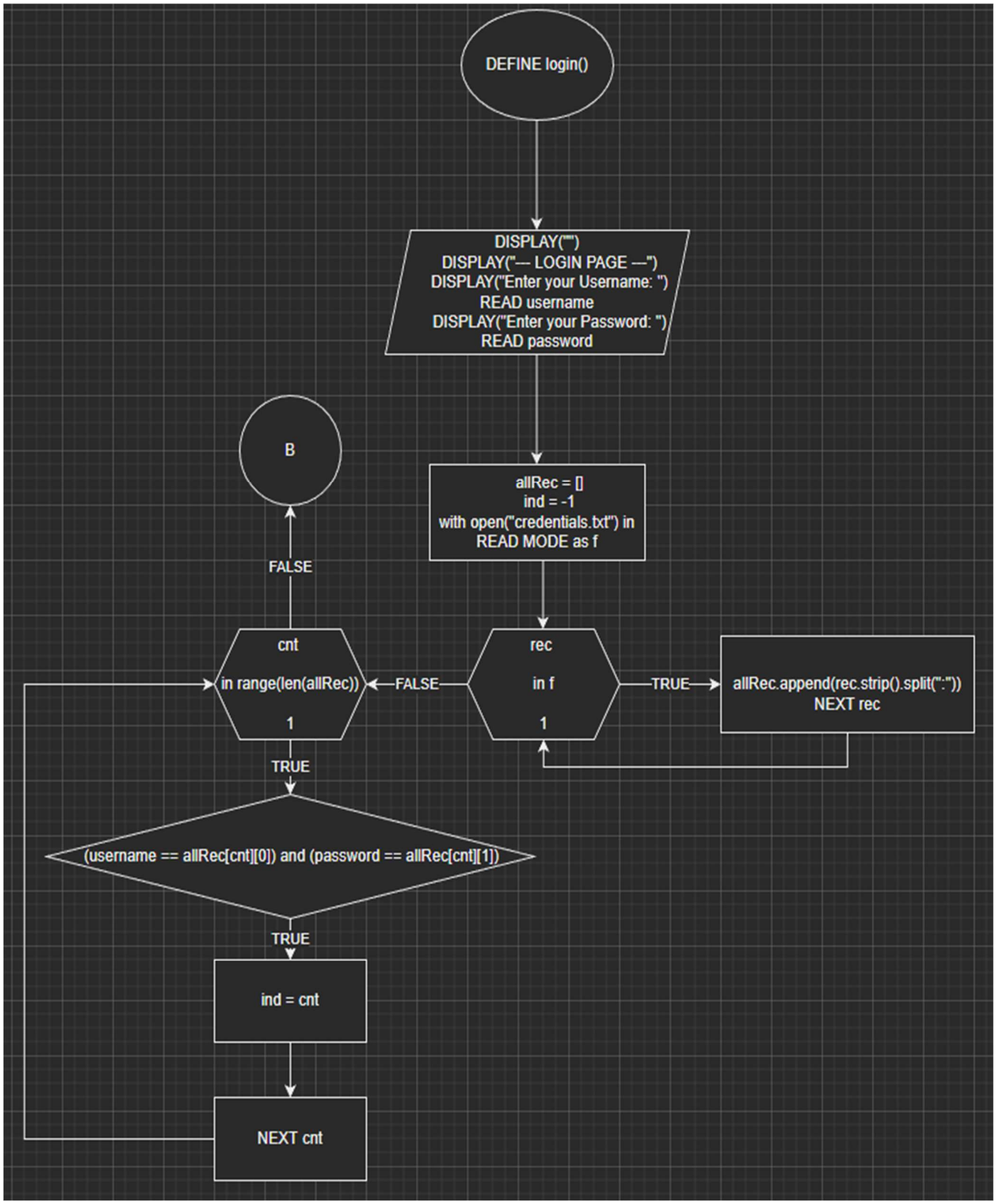
admin_menu()

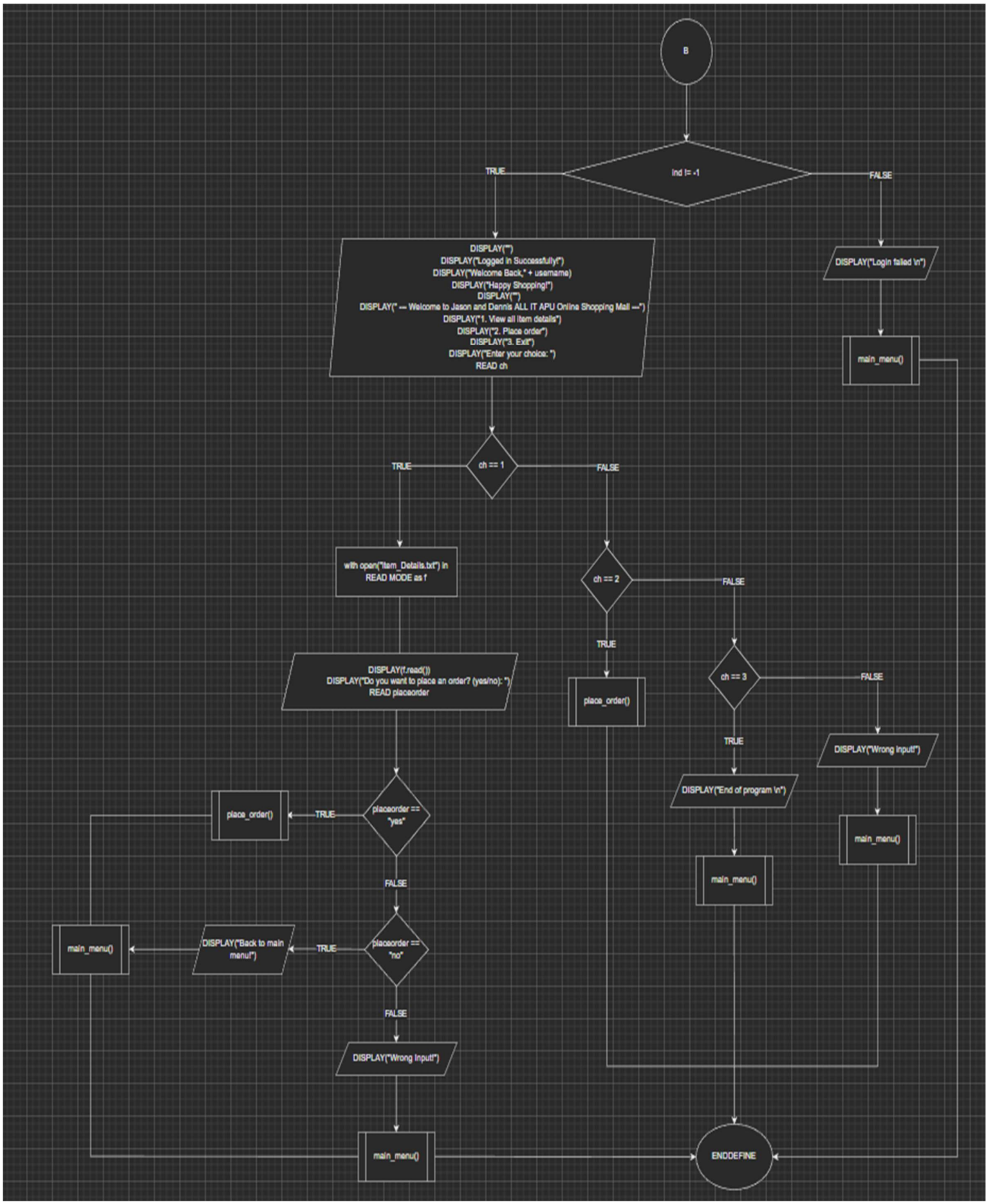

DEFINE
admin_menu()
DISPLAY(" --- ADMIN MENU ---")
DISPLAY("1. Add item category")
DISPLAY("2. Modify item")
DISPLAY("3. Delete item")
DISPLAY("4. View/Search customer order")
DISPLAY("5. View/Search customer payment")
DISPLAY("6. Order Delivery Management")
DISPLAY("7. Exit")
DISPLAY("Enter your choice: ")
READ ch
ch == 1
TRUE
add_item_category()
FALSE
ch == 2
TRUE
modify_item()
FALSE
ch == 3
TRUE
remove_item()
FALSE
ch == 4
TRUE
view_search_customer_order
FALSE
ch == 5
TRUE
view_search_customer_payment
FALSE
ch == 6
TRUE
odm()
FALSE
ch == 7
TRUE
DISPLAY("End of program \n")
main_menu()
FALSE
DISPLAY("Wrong input!")
DISPLAY("Wrong choice! \n")
main_menu()
ENDDEFINE

register()

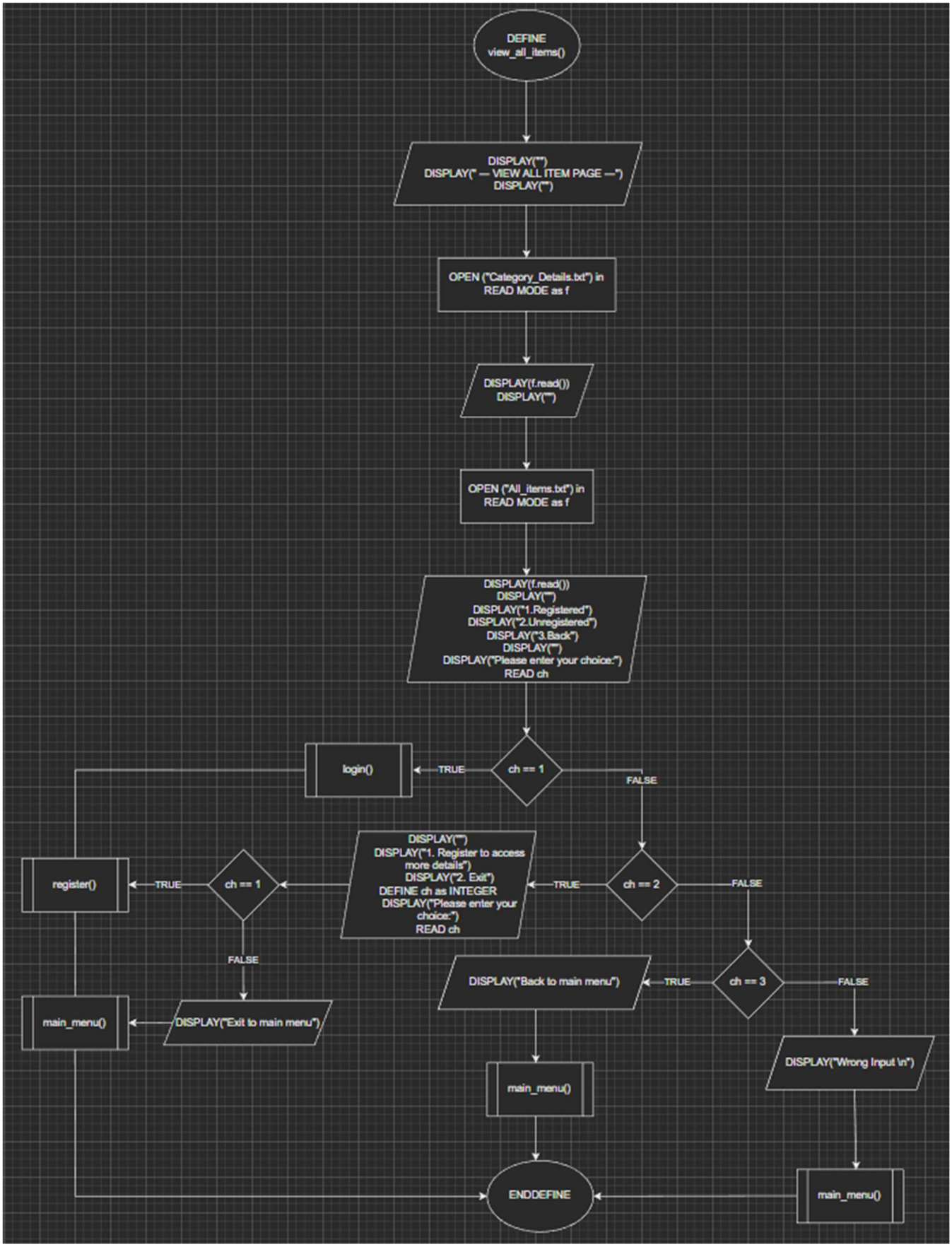

DEFINE register()
DISPLAY("")
DISPLAY("— REGISTER PAGE —")
DISPLAY("Enter Username: ")
READ username
DISPLAY("Enter password: ")
READ password
DISPLAY("Confirm password: ")
READ password1
password == password1
TRUE
FALSE
with open("credentials.txt") in APPEND MODE as f
rec = username +":"+password
WRITE (rec +"\n") in f
DISPLAY("You have registered successfully!")
login()
DISPLAY("Password is not same as above! \n")
register()
ENDDEFINE

login()

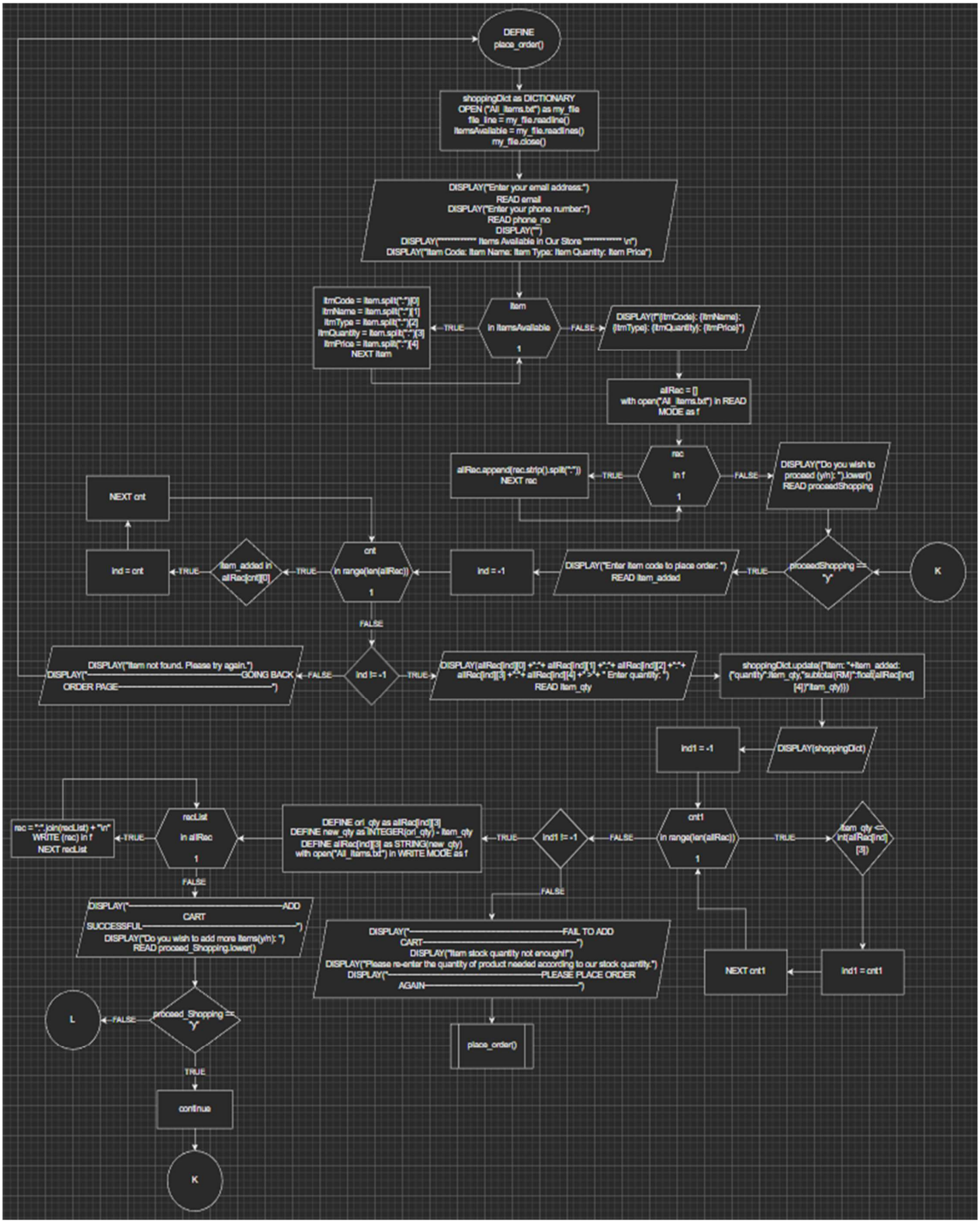

DEFINE login()
DISPLAY("")
DISPLAY("— LOGIN PAGE —")
DISPLAY("Enter your Username: ")
READ username
DISPLAY("Enter your Password: ")
READ password
B
allRec = []
ind = -1
with open("credentials.txt") in
READ MODE as f
FALSE
cnt
in range(len(allRec))
1
FALSE
rec
in f
1
TRUE
allRec.append(rec.strip().split(":"))
NEXT rec
TRUE
(username == allRec[cnt][0]) and (password == allRec[cnt][1])
TRUE
ind = cnt
NEXT cnt

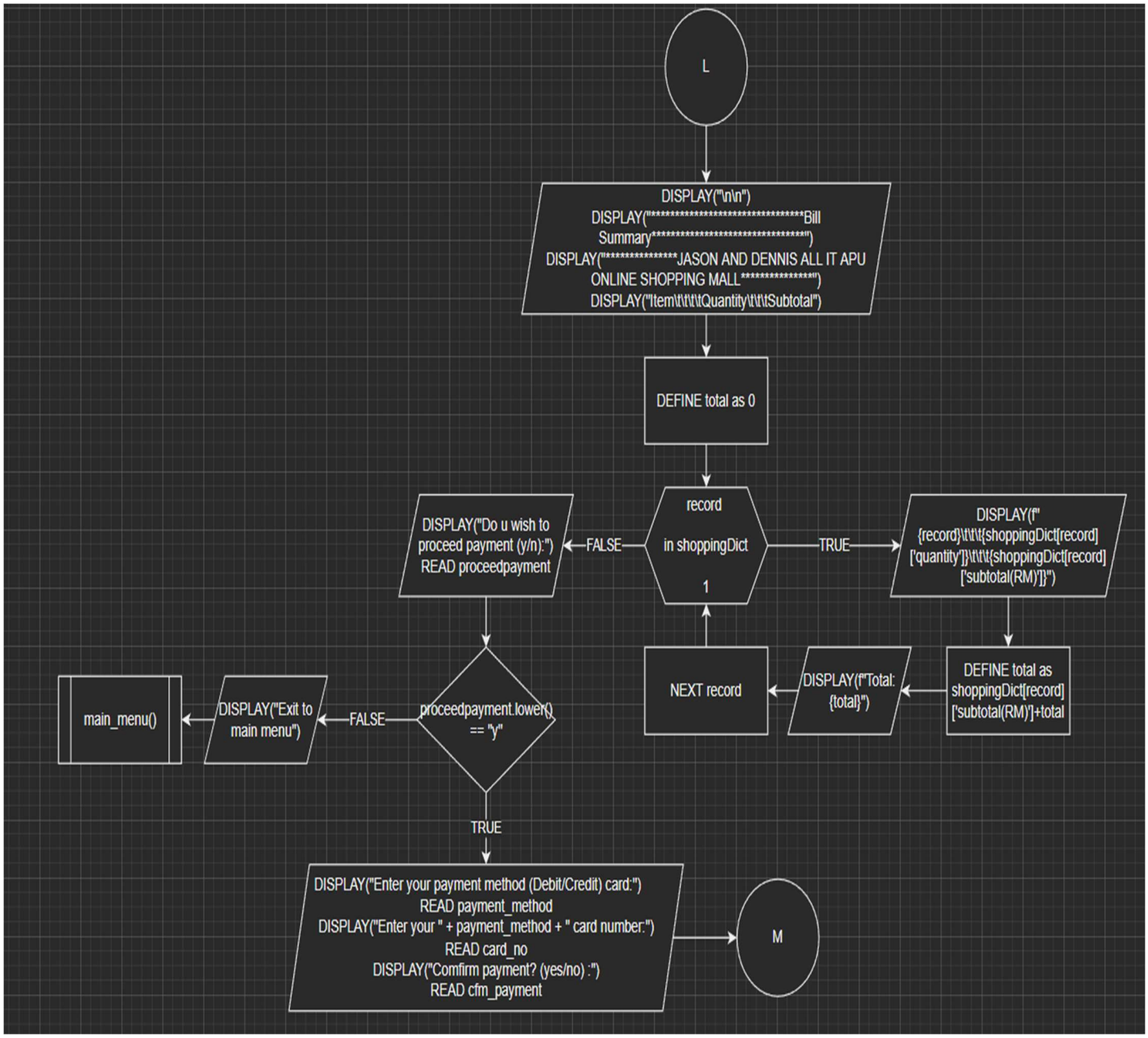

B
ind != -1
TRUE
FALSE
DISPLAY("")
DISPLAY("Logged in Successfully!")
DISPLAY("Welcome Back," + username)
DISPLAY("Happy Shopping!")
DISPLAY("")
DISPLAY(" --- Welcome to Jason and Dennis ALL IT APU Online Shopping Mall ---")
DISPLAY("1. View all item details")
DISPLAY("2. Place order")
DISPLAY("3. Exit")
DISPLAY("Enter your choice: ")
READ ch
DISPLAY("Login failed \n")
main_menu()
ch == 1
TRUE
FALSE
with open("Item_Details.txt") in READ MODE as f
DISPLAY(f.read())
DISPLAY("Do you want to place an order? (yes/no): ")
READ placeorder
ch == 2
TRUE
FALSE
place_order()
ch == 3
TRUE
FALSE
DISPLAY("End of program \n")
DISPLAY("Wrong input!")
main_menu()
main_menu()
placeorder == "yes"
TRUE
place_order()
FALSE
placeorder == "no"
TRUE
DISPLAY("Back to main menu!")
main_menu()
FALSE
DISPLAY("Wrong Input!")
main_menu()
ENDDEFINE

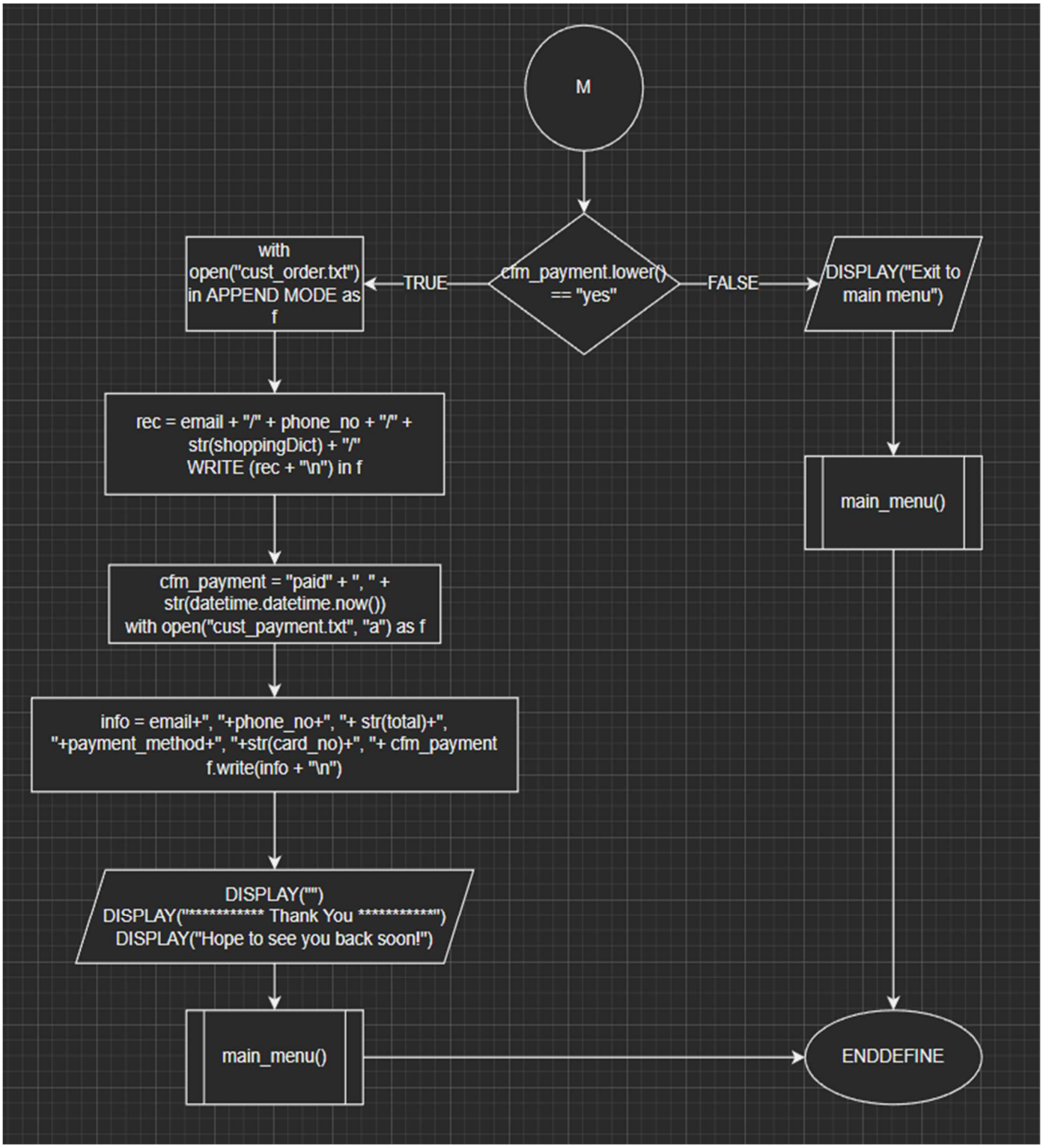
view_all_items()



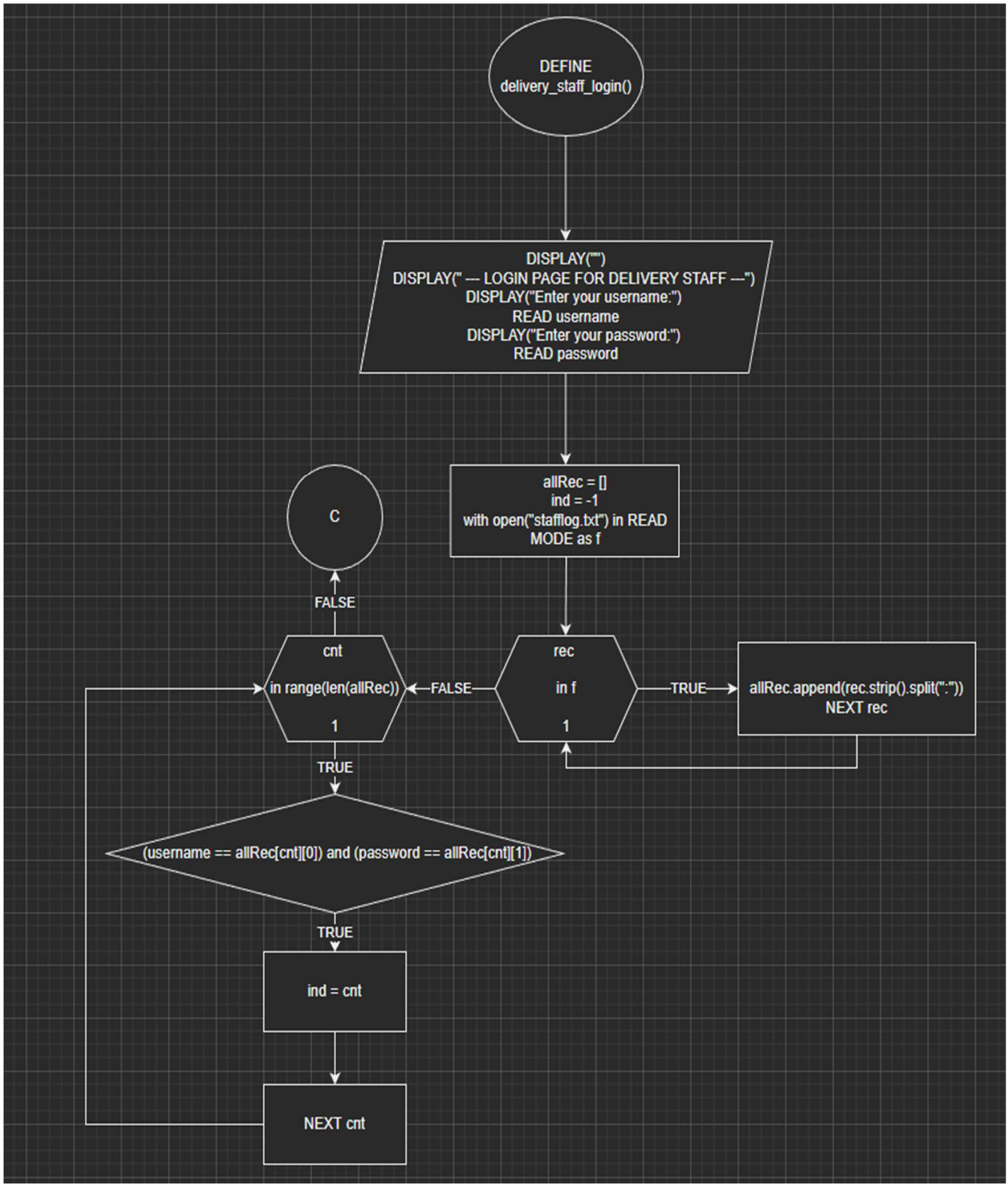
place_order()

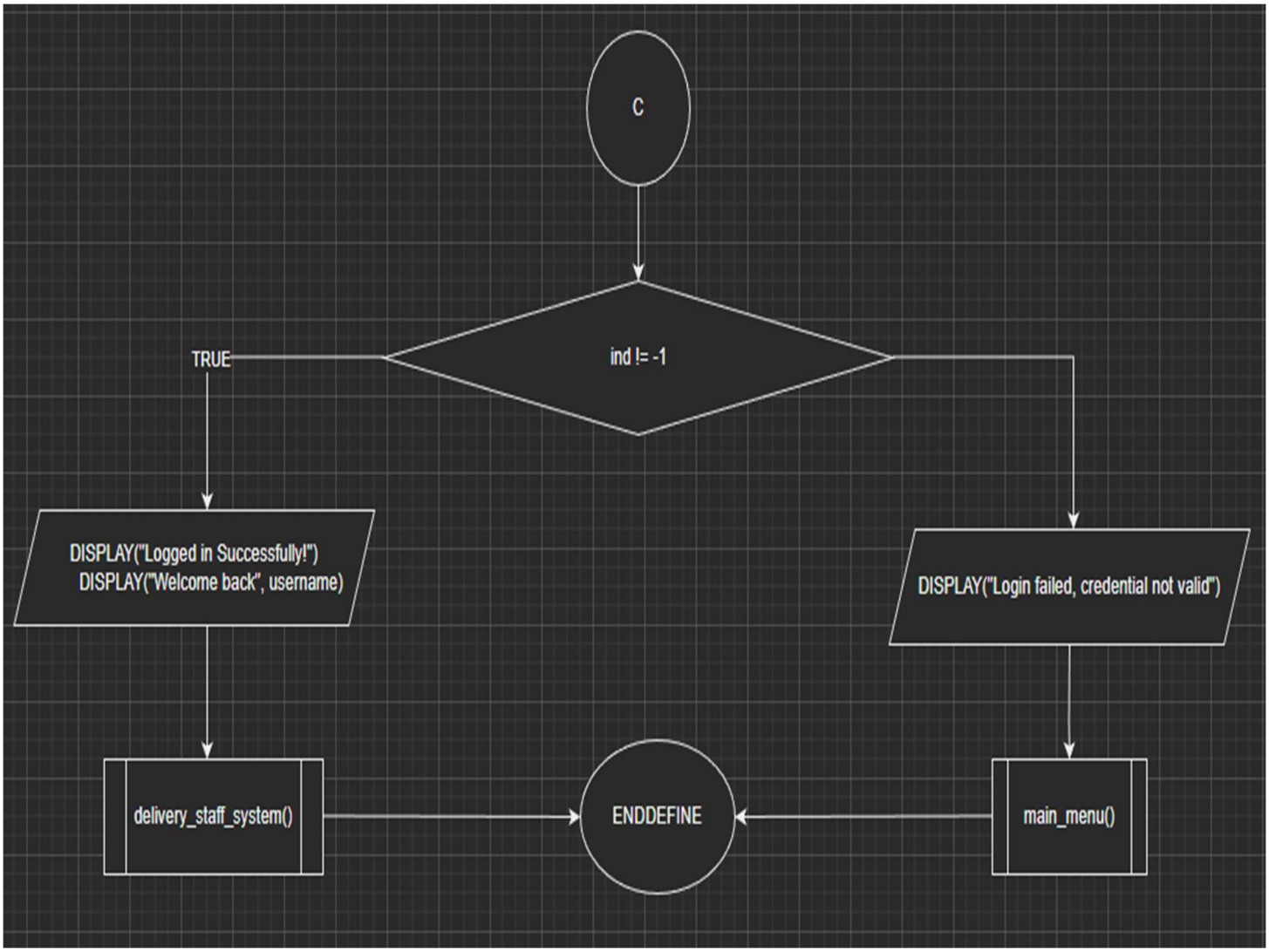

DEFINE place_order()
shoppingDict as DICTIONARY
OPEN ("All_Items.txt") as my_file
file_line = my_file.readline()
ItemsAvailable = my_file.readlines()
my_file.close()
DISPLAY("Enter your email address:")
READ email
DISPLAY("Enter your phone number:")
READ phone_no
DISPLAY("")
DISPLAY("********** Items Available in Our Store **********\n")
DISPLAY("Item Code: Item Name: Item Type: Item Quantity: Item Price")
item in ItemsAvailable 1
TRUE
itmCode = item.split(":")[0]
itmName = item.split(":")[1]
itmType = item.split(":")[2]
itmQuantity = item.split(":")[3]
itmPrice = item.split(":")[4]
NEXT item
FALSE
DISPLAY(f"{itmCode}: {itmName}: {itmType}: {itmQuantity}: {itmPrice}")
allRec = []
with open("All_Items.txt") in READ MODE as f
rec in f 1
TRUE
allRec.append(rec.strip().split(":"))
NEXT rec
FALSE
DISPLAY("Do you wish to proceed (y/n): ").lower()
READ proceedShopping
proceedShopping == "y"
TRUE
K
DISPLAY("Enter item code to place order: ")
READ item_added
ind = -1
cnt in range(len(allRec)) 1
item_added in allRec[cnt][0]
TRUE
ind = cnt
NEXT cnt
FALSE
ind != -1
TRUE
DISPLAY(allRec[ind][0] +":"+ allRec[ind][1] +":"+ allRec[ind][2] +":"+ allRec[ind][3] +":"+ allRec[ind][4] +" :"+ " Enter quantity: ")
READ item_qty
shoppingDict.update({"item: "+item_added: {"quantity":item_qty,"subtotal(RM)":float(allRec[ind][4])*item_qty}})
FALSE
DISPLAY("Item not found. Please try again.")
DISPLAY("GOING BACK ORDER PAGE")
DISPLAY(shoppingDict)
ind1 = -1
cnt1 in range(len(allRec)) 1
item_qty <= int(allRec[ind][3])
TRUE
ind1 = cnt1
NEXT cnt1
FALSE
ind1 != -1
TRUE
DEFINE ori_qty as allRec[ind][3]
DEFINE new_qty as INTEGER(ori_qty) - item_qty
DEFINE allRec[ind][3] as STRING(new_qty)
with open("All_Items.txt") in WRITE MODE as f
recList in allRec 1
TRUE
rec = ":".join(recList) + "\n"
WRITE (rec) in f
NEXT recList
FALSE
DISPLAY("ADD TO CART SUCCESSFUL")
DISPLAY("Do you wish to add more items(y/n): ")
READ proceed_Shopping.lower()
DISPLAY("FAIL TO ADD TO CART")
DISPLAY("Item stock quantity not enough!")
DISPLAY("Please re-enter the quantity of product needed according to our stock quantity.")
DISPLAY("PLEASE PLACE ORDER AGAIN")
place_order()
proceed_Shopping == "y"
FALSE
L
TRUE
continue
K


L
DISPLAY("\n\n")
DISPLAY("**********Bill Summary**********")
DISPLAY("**********JASON AND DENNIS ALL IT APU ONLINE SHOPPING MALL**********")
DISPLAY("Item\t\t\t\tQuantity\t\t\tSubtotal")
DEFINE total as 0
record in shoppingDict
1
TRUE
DISPLAY(f"{record}\t\t\t{shoppingDict[record]['quantity']}\t\t\t{shoppingDict[record]['subtotal(RM)']}")
DEFINE total as shoppingDict[record]['subtotal(RM)']+total
DISPLAY(f"Total: {total}")
NEXT record
FALSE
DISPLAY("Do u wish to proceed payment (y/n):")
READ proceedpayment
proceedpayment.lower() == "y"
FALSE
DISPLAY("Exit to main menu")
main_menu()
TRUE
DISPLAY("Enter your payment method (Debit/Credit) card:")
READ payment_method
DISPLAY("Enter your " + payment_method + " card number:")
READ card_no
DISPLAY("Comfirm payment? (yes/no) :")
READ cfm_payment
M

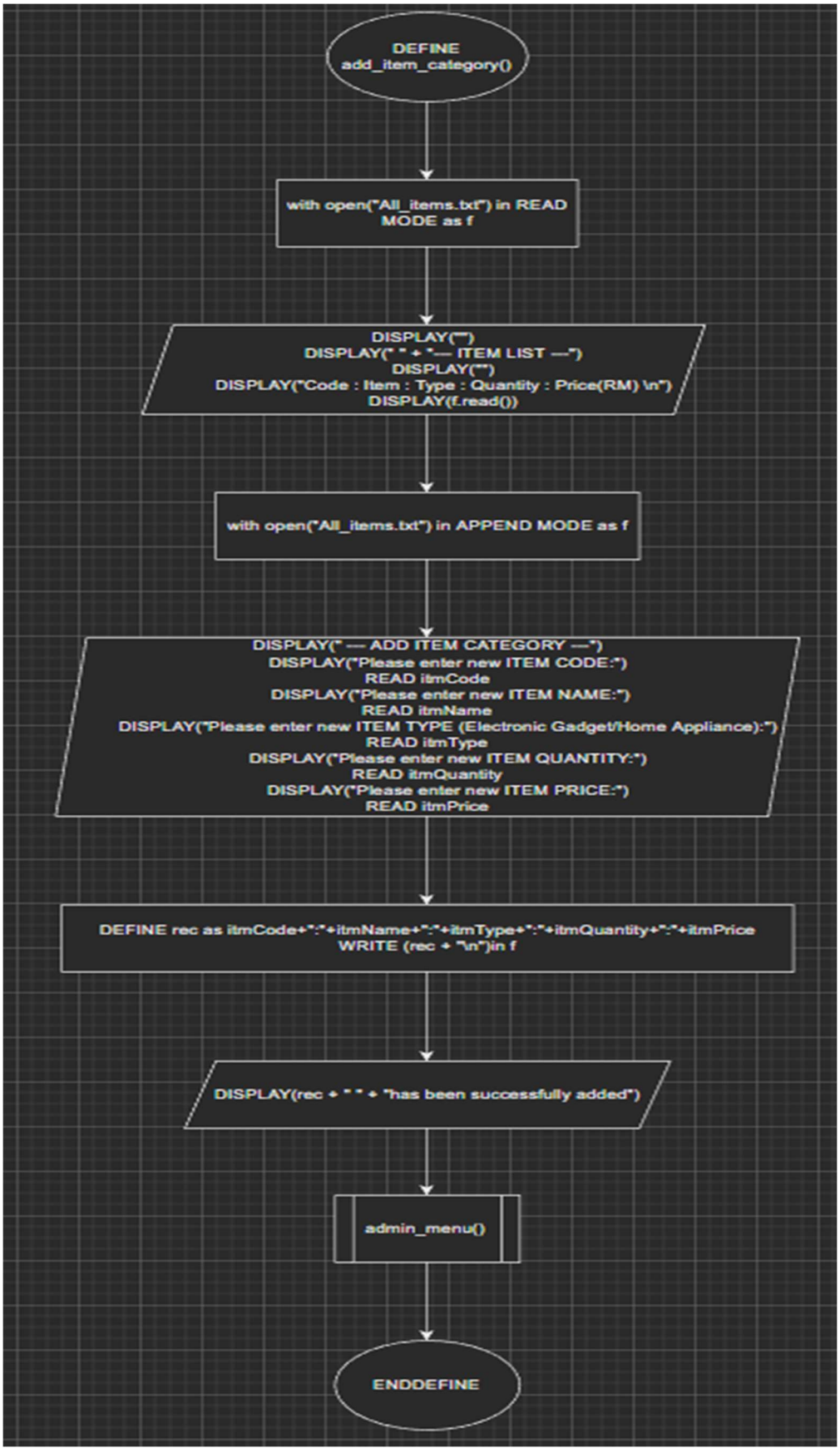

M
cfm_payment.lower() == "yes"
TRUE
with open("cust_order.txt") in APPEND MODE as f
FALSE
DISPLAY("Exit to main menu")
rec = email + "/" + phone_no + "/" + str(shoppingDict) + "/"
WRITE (rec + "\n") in f
main_menu()
cfm_payment = "paid" + ", " + str(datetime.datetime.now())
with open("cust_payment.txt", "a") as f
info = email+", "+phone_no+", "+ str(total)+", "+payment_method+", "+str(card_no)+", "+ cfm_payment
f.write(info + "\n")
DISPLAY("")
DISPLAY("********** Thank You **********")
DISPLAY("Hope to see you back soon!")
main_menu()
ENDDEFINE

delivery_staff_login()

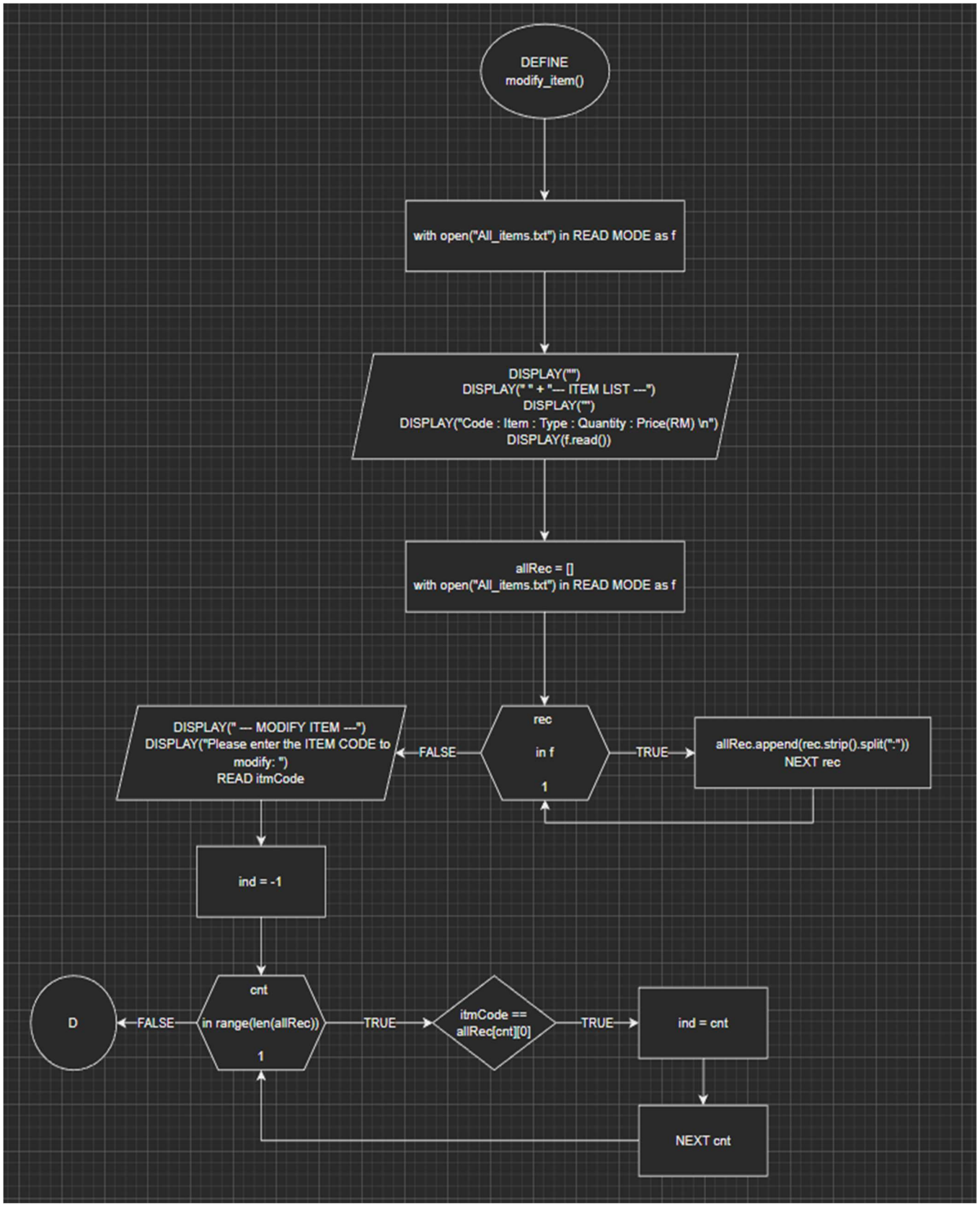


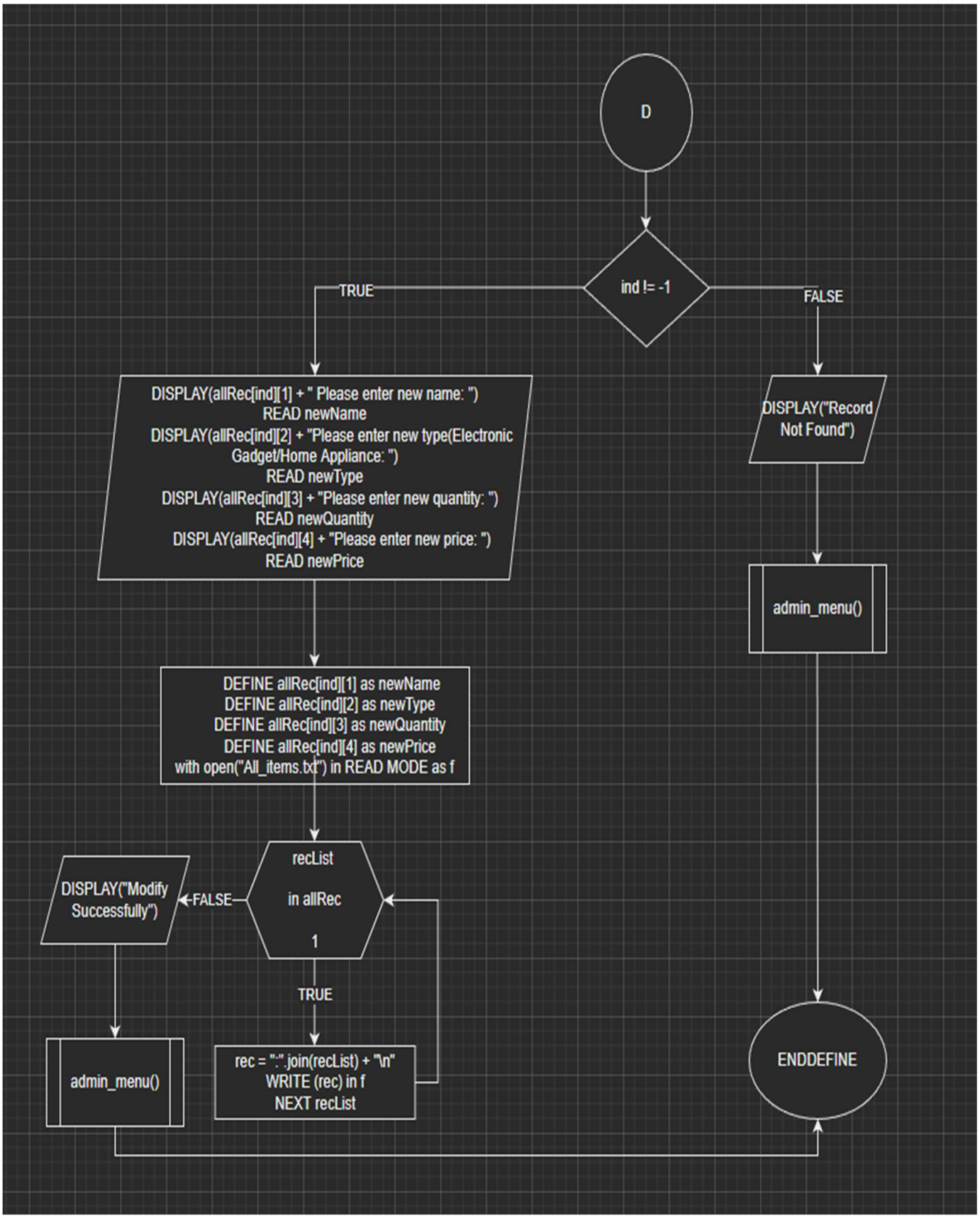

C
ind != -1
TRUE
DISPLAY("Logged in Successfully!")
DISPLAY("Welcome back", username)
DISPLAY("Login failed, credential not valid")
delivery_staff_system()
ENDDEFINE
main_menu()

add_item_category()

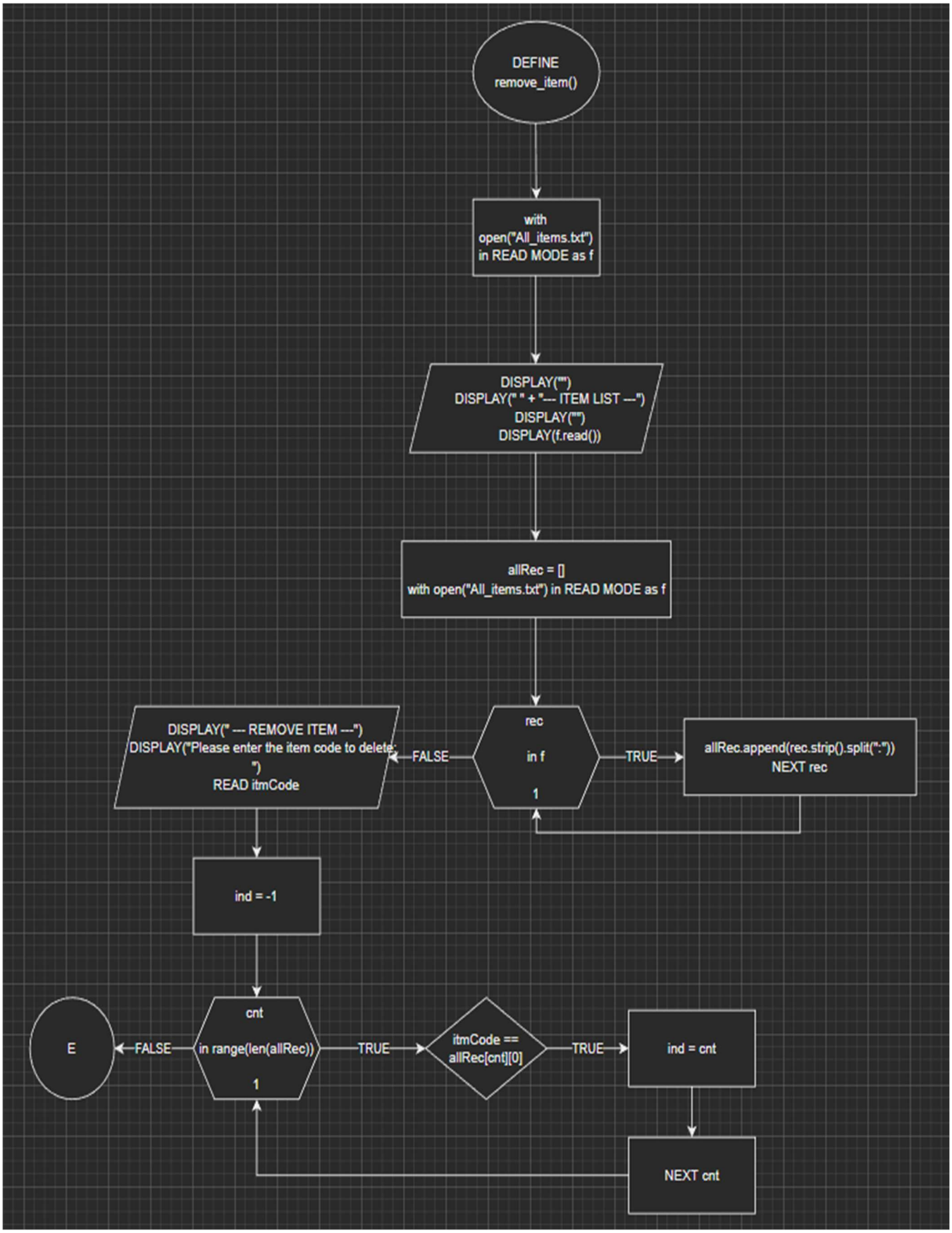


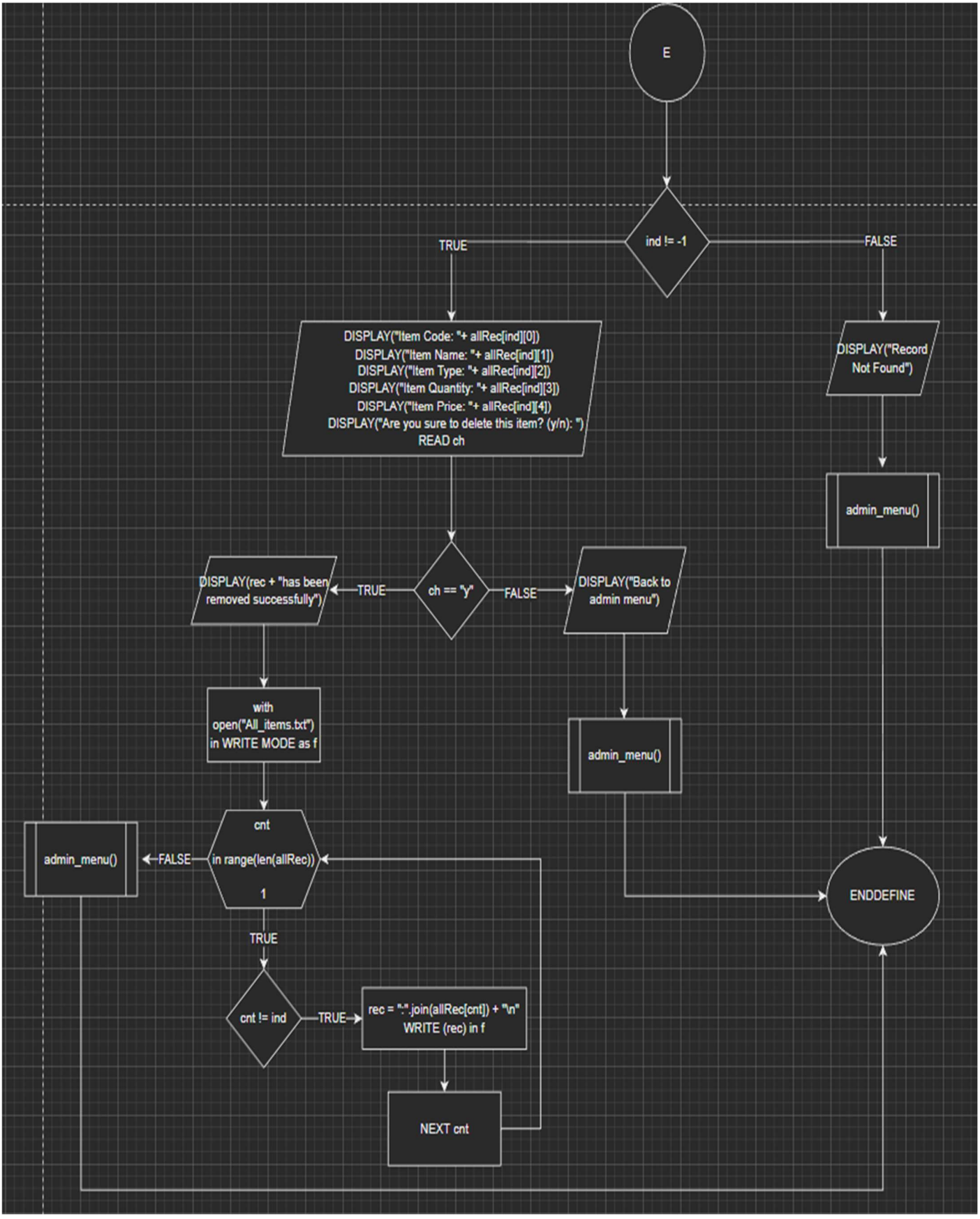
modify_item()


DEFINE
modify_item()
with open("All_items.txt") in READ MODE as f
DISPLAY("")
DISPLAY(" " + "--- ITEM LIST ---")
DISPLAY("")
DISPLAY("Code : Item : Type : Quantity : Price(RM) \n")
DISPLAY(f.read())
allRec = []
with open("All_items.txt") in READ MODE as f
rec
in f
1
TRUE
allRec.append(rec.strip().split(":"))
NEXT rec
FALSE
DISPLAY(" --- MODIFY ITEM ---")
DISPLAY("Please enter the ITEM CODE to modify: ")
READ itmCode
ind = -1
cnt
in range(len(allRec))
1
TRUE
itmCode == allRec[cnt][0]
TRUE
ind = cnt
NEXT cnt
FALSE
D

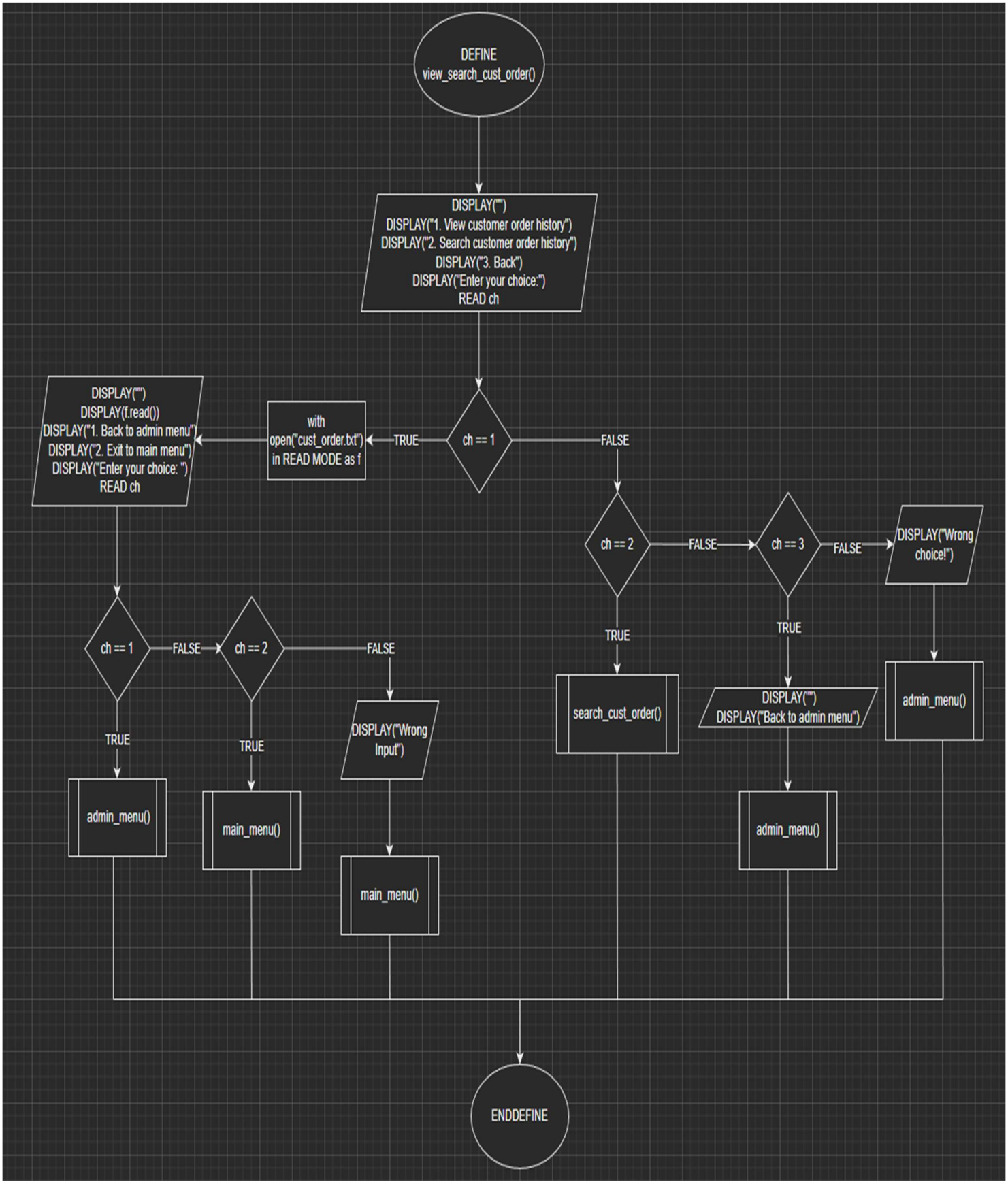

D
ind != -1
TRUE
FALSE
DISPLAY(allRec[ind][1] + " Please enter new name: ")
READ newName
DISPLAY(allRec[ind][2] + "Please enter new type(Electronic Gadget/Home Appliance: ")
READ newType
DISPLAY(allRec[ind][3] + "Please enter new quantity: ")
READ newQuantity
DISPLAY(allRec[ind][4] + "Please enter new price: ")
READ newPrice
DISPLAY("Record Not Found")
admin_menu()
DEFINE allRec[ind][1] as newName
DEFINE allRec[ind][2] as newType
DEFINE allRec[ind][3] as newQuantity
DEFINE allRec[ind][4] as newPrice
with open("All_items.txt") in READ MODE as f
recList
in allRec
1
FALSE
DISPLAY("Modify Successfully")
TRUE
admin_menu()
rec = ":".join(recList) + "\n"
WRITE (rec) in f
NEXT recList
ENDDEFINE

remove_item()

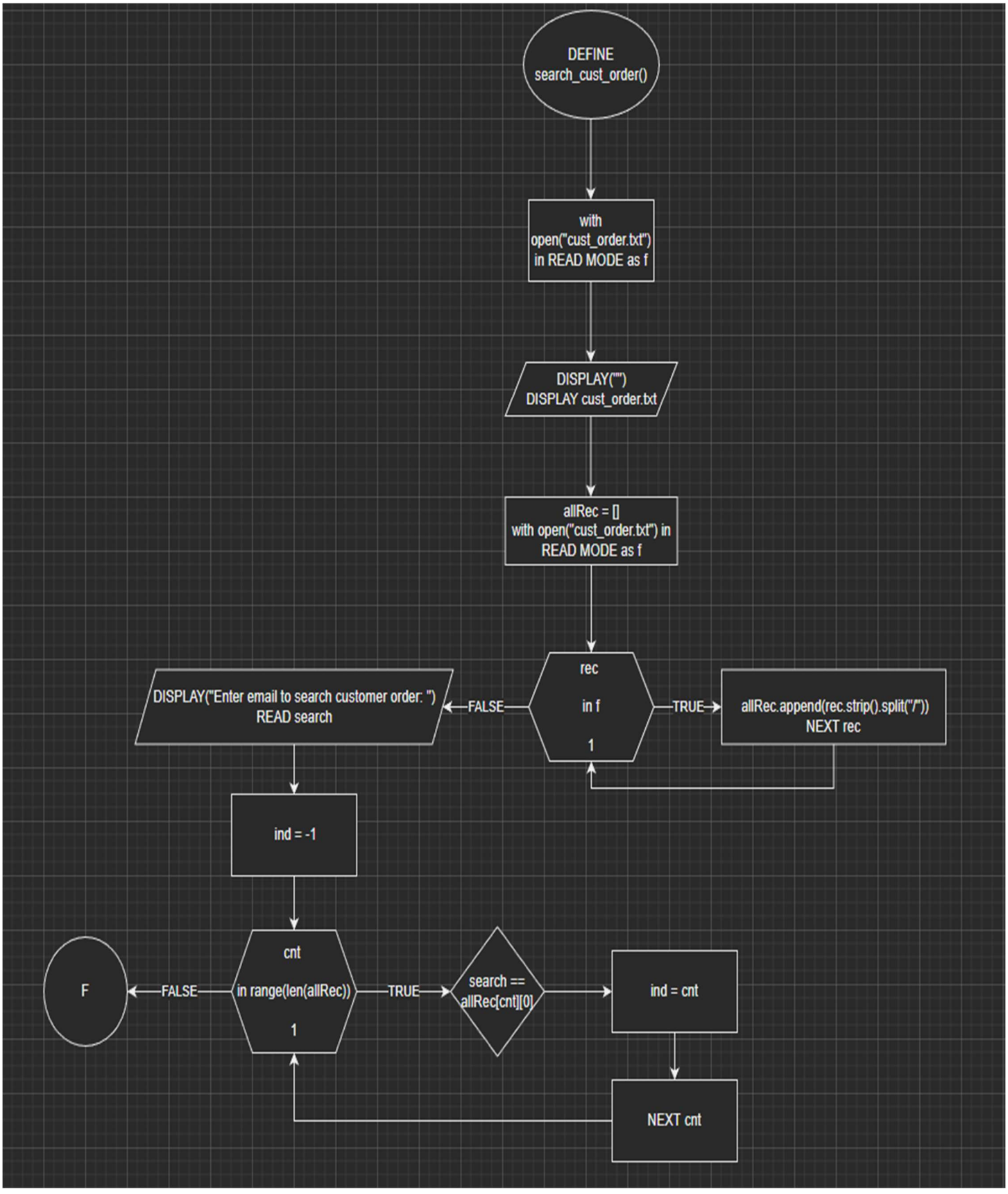

DEFINE
remove_item()
with
open("All_items.txt")
in READ MODE as f
DISPLAY("")
DISPLAY(" " + "--- ITEM LIST ---")
DISPLAY("")
DISPLAY(f.read())
allRec = []
with open("All_items.txt") in READ MODE as f
rec
in f
1
FALSE
TRUE
allRec.append(rec.strip().split(":"))
NEXT rec
DISPLAY(" --- REMOVE ITEM ---")
DISPLAY("Please enter the item code to delete: ")
READ itmCode
ind = -1
cnt
in range(len(allRec))
1
FALSE
TRUE
itmCode == allRec[cnt][0]
TRUE
ind = cnt
NEXT cnt
E

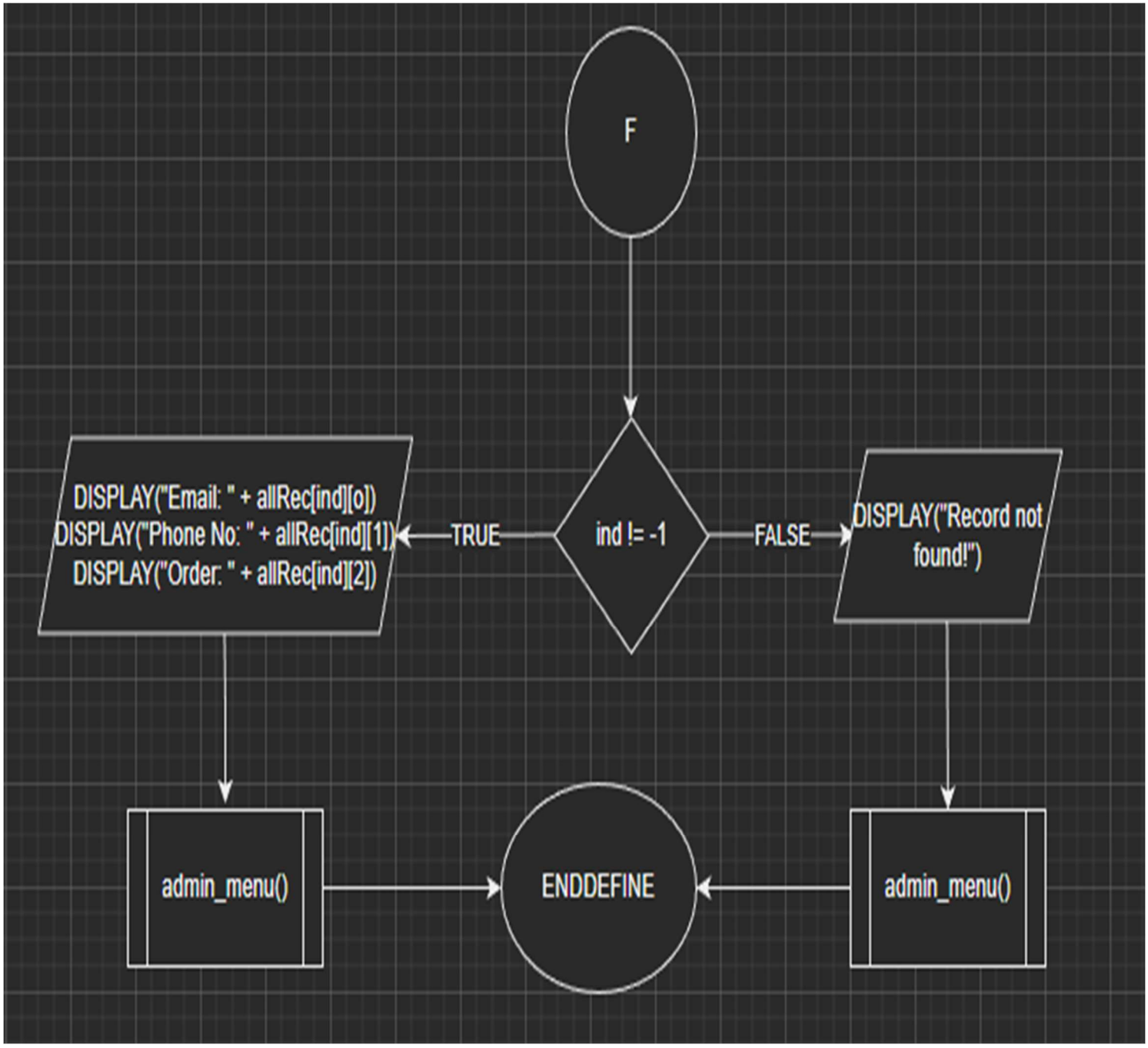

E
ind != -1
TRUE
FALSE
DISPLAY("Item Code: "+ allRec[ind][0])
DISPLAY("Item Name: "+ allRec[ind][1])
DISPLAY("Item Type: "+ allRec[ind][2])
DISPLAY("Item Quantity: "+ allRec[ind][3])
DISPLAY("Item Price: "+ allRec[ind][4])
DISPLAY("Are you sure to delete this item? (y/n): ")
READ ch
DISPLAY("Record Not Found")
admin_menu()
ch == "y"
TRUE
FALSE
DISPLAY(rec + "has been removed successfully")
DISPLAY("Back to admin menu")
with open("All_items.txt") in WRITE MODE as f
admin_menu()
cnt in range(len(allRec)) 1
FALSE
admin_menu()
TRUE
ENDDEFINE
cnt != ind
TRUE
rec = ":".join(allRec[cnt]) + "\n"
WRITE (rec) in f
NEXT cnt

view_search_cust_order()

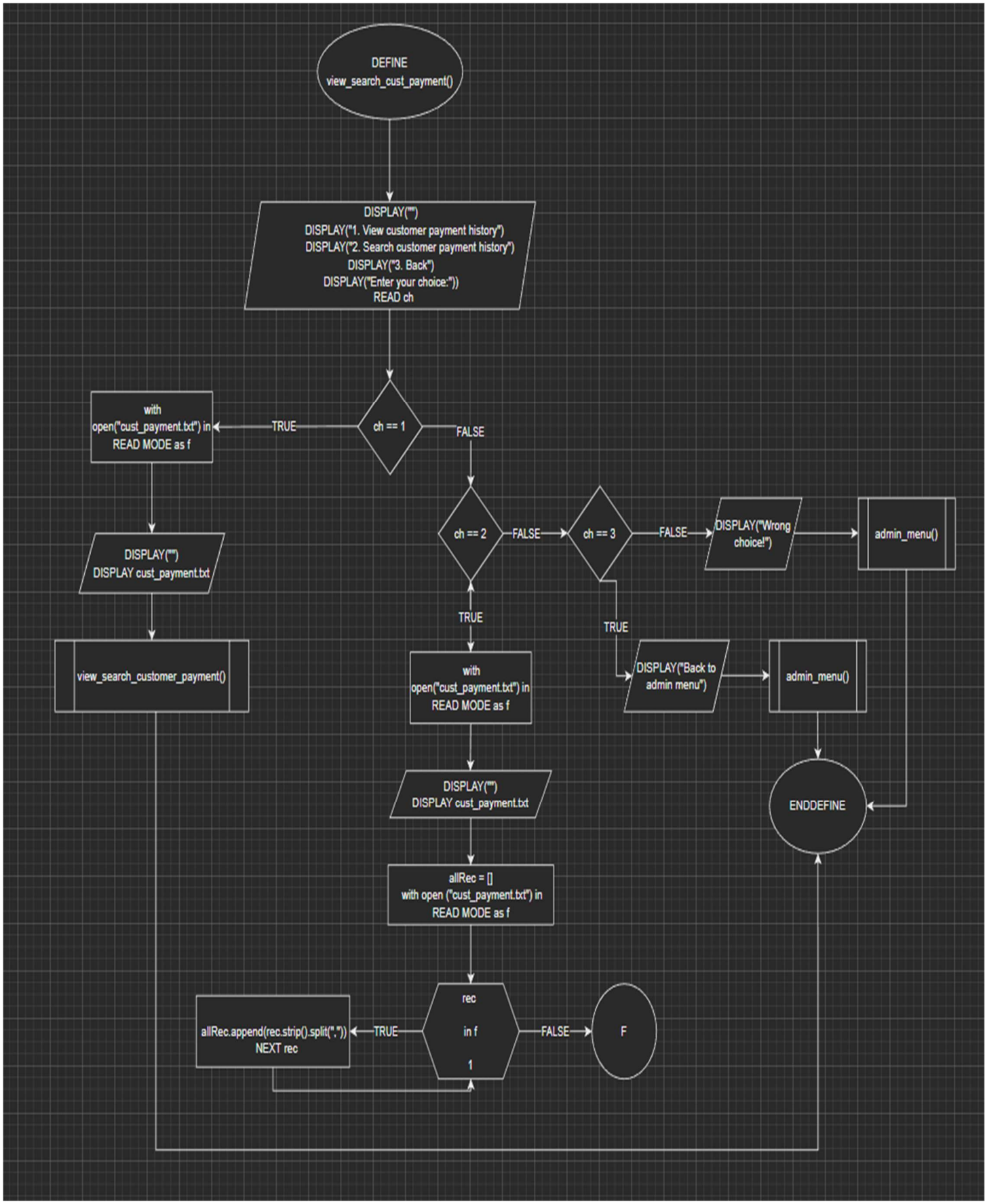


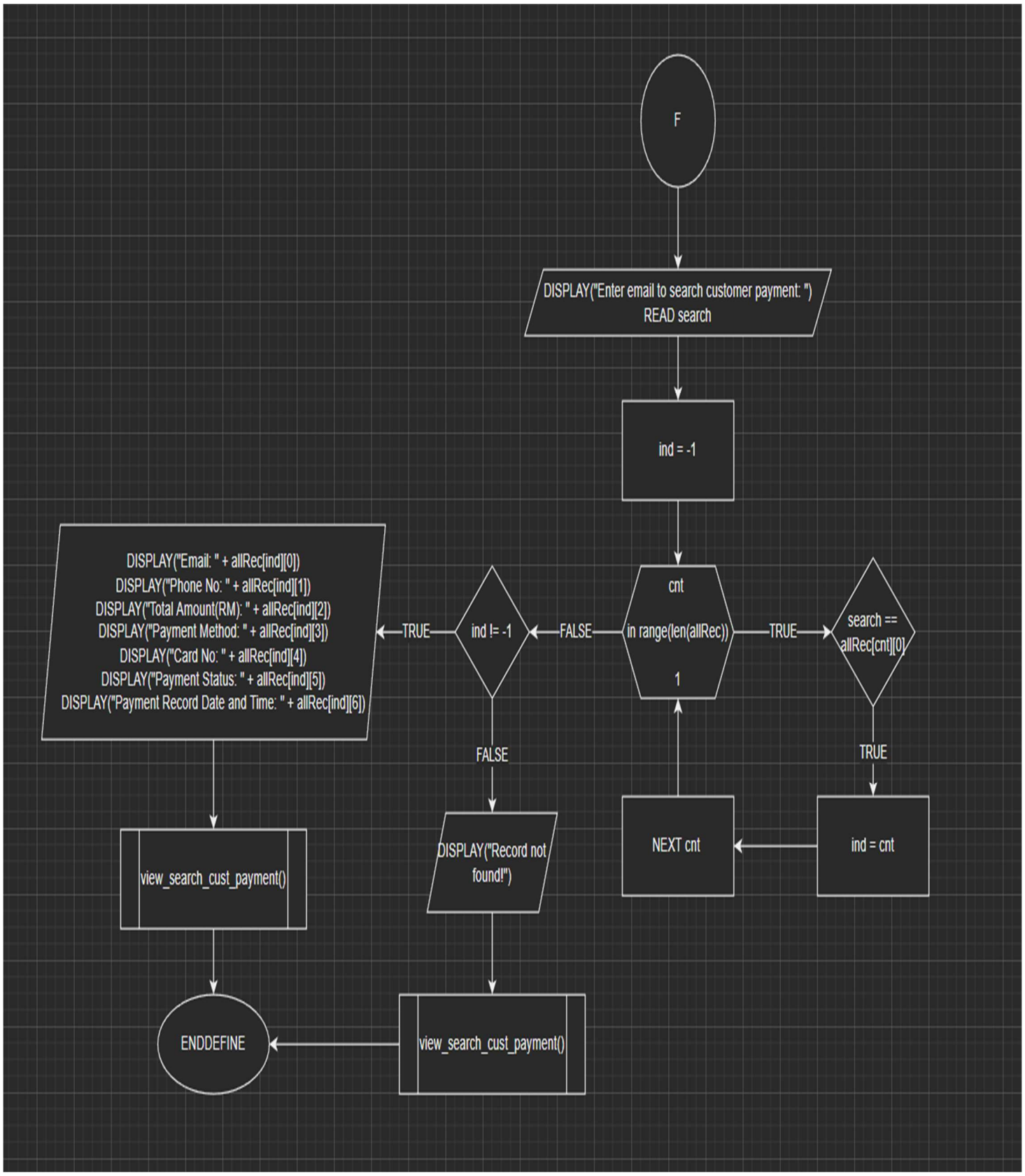
search_cust_order()



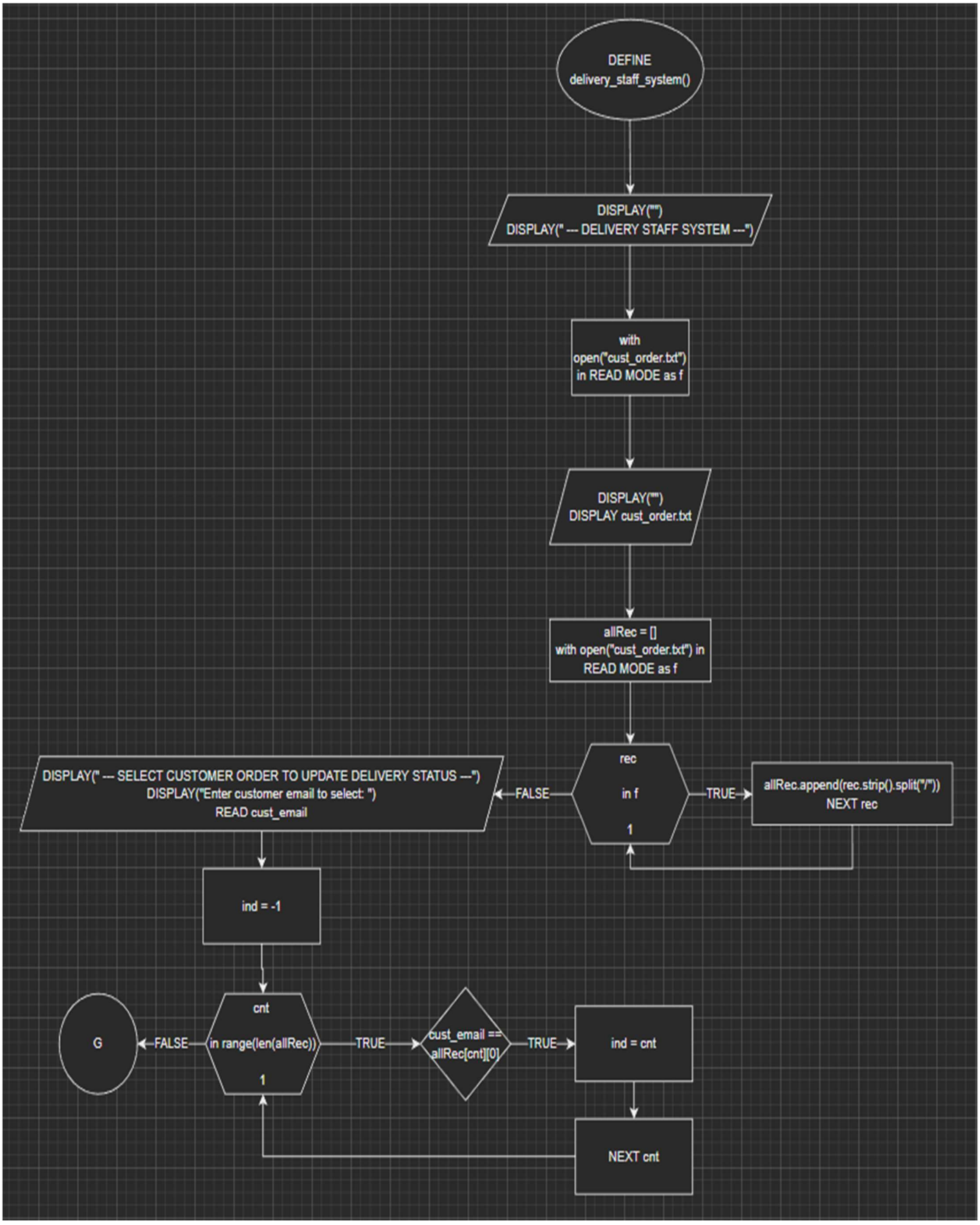

F
ind != -1
TRUE
FALSE
DISPLAY("Email: " + allRec[ind][o])
DISPLAY("Phone No: " + allRec[ind][1])
DISPLAY("Order: " + allRec[ind][2])
DISPLAY("Record not found!")
admin_menu()
ENDDEFINE
admin_menu()

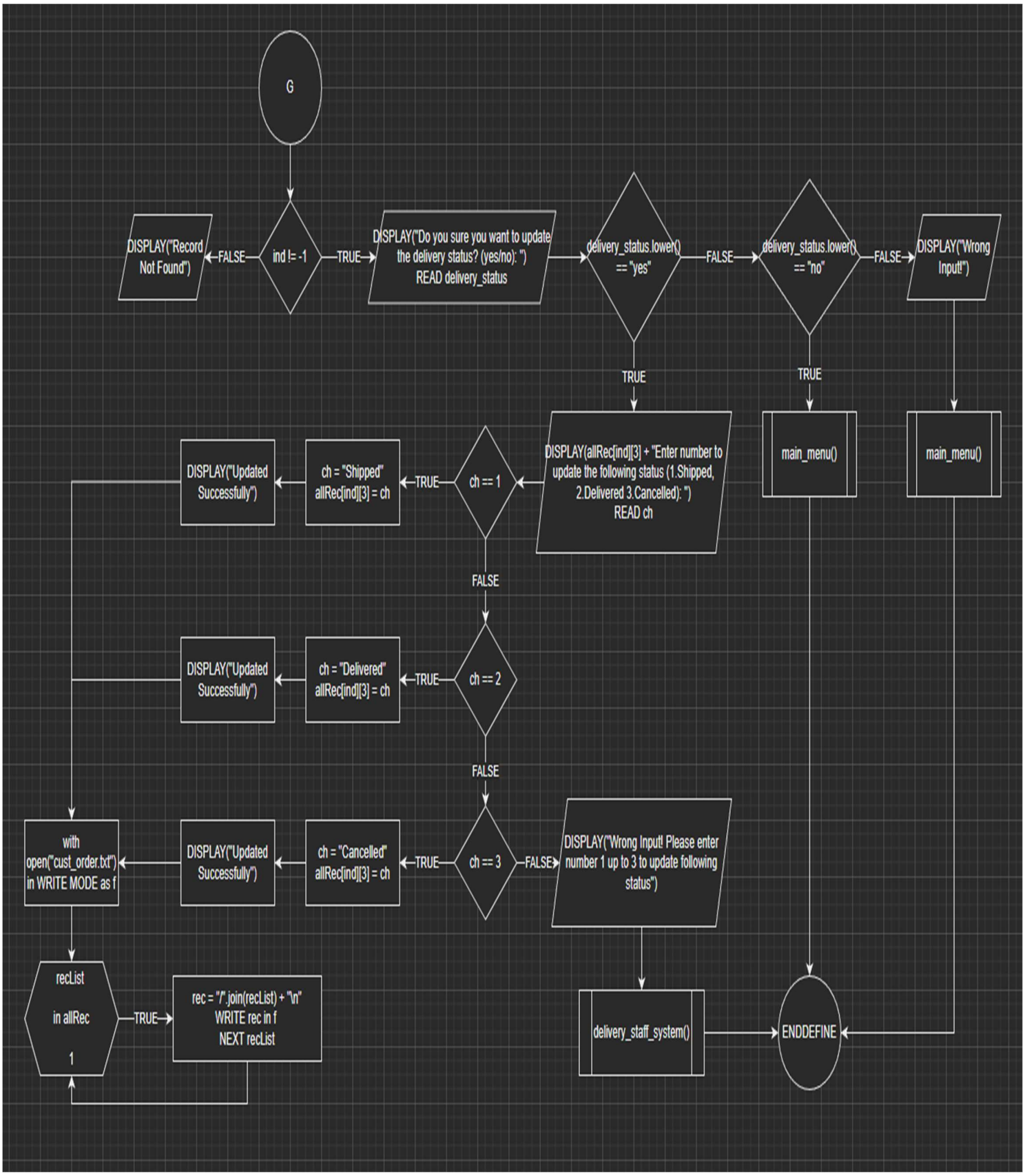
view_search_cust_payment()



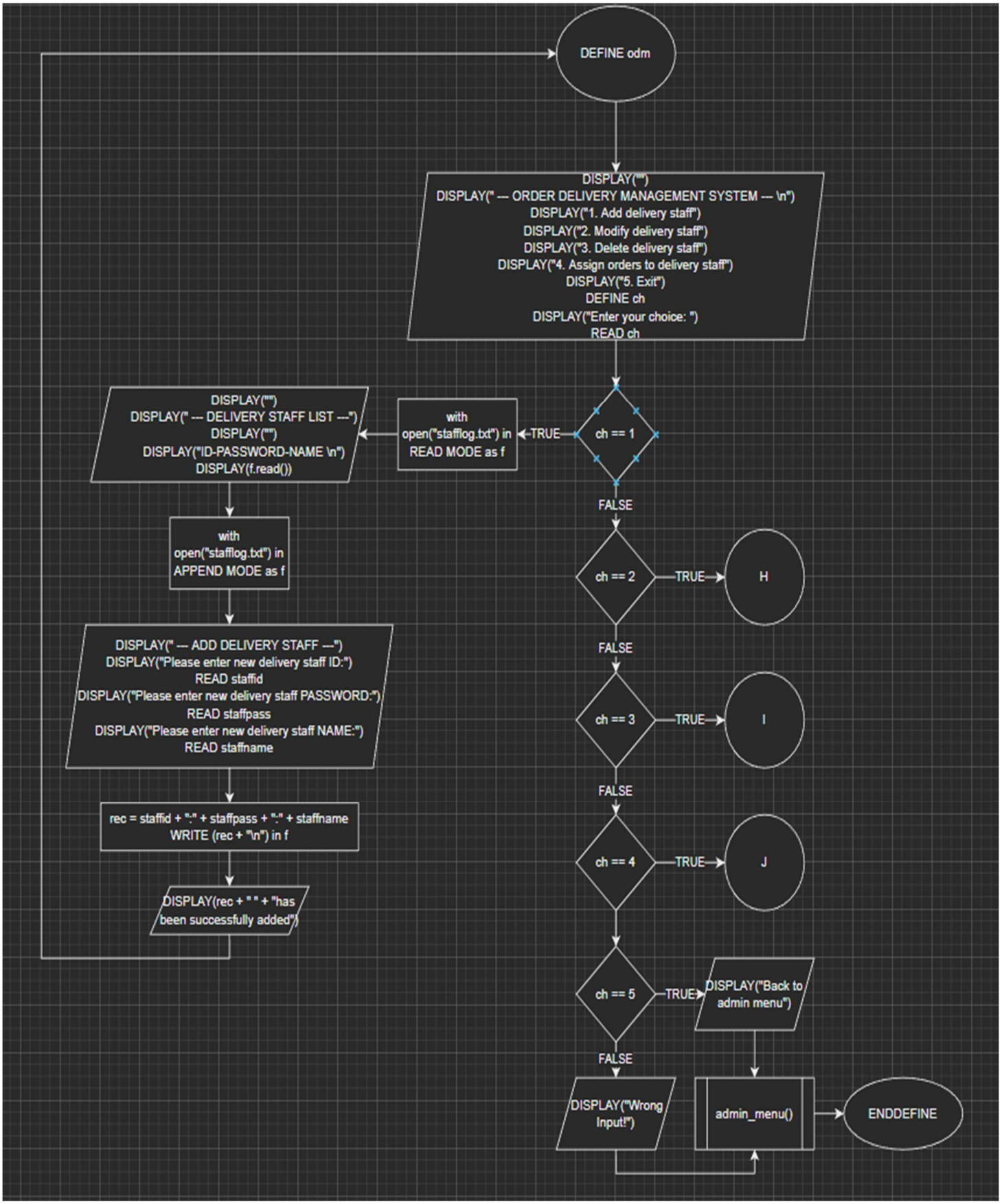

F
DISPLAY("Enter email to search customer payment: ")
READ search
ind = -1
cnt
in range(len(allRec))
1
search == allRec[cnt][0]
TRUE
FALSE
TRUE
ind != -1
TRUE
FALSE
DISPLAY("Email: " + allRec[ind][0])
DISPLAY("Phone No: " + allRec[ind][1])
DISPLAY("Total Amount(RM): " + allRec[ind][2])
DISPLAY("Payment Method: " + allRec[ind][3])
DISPLAY("Card No: " + allRec[ind][4])
DISPLAY("Payment Status: " + allRec[ind][5])
DISPLAY("Payment Record Date and Time: " + allRec[ind][6])
ind = cnt
NEXT cnt
DISPLAY("Record not found!")
view_search_cust_payment()
view_search_cust_payment()
ENDDEFINE

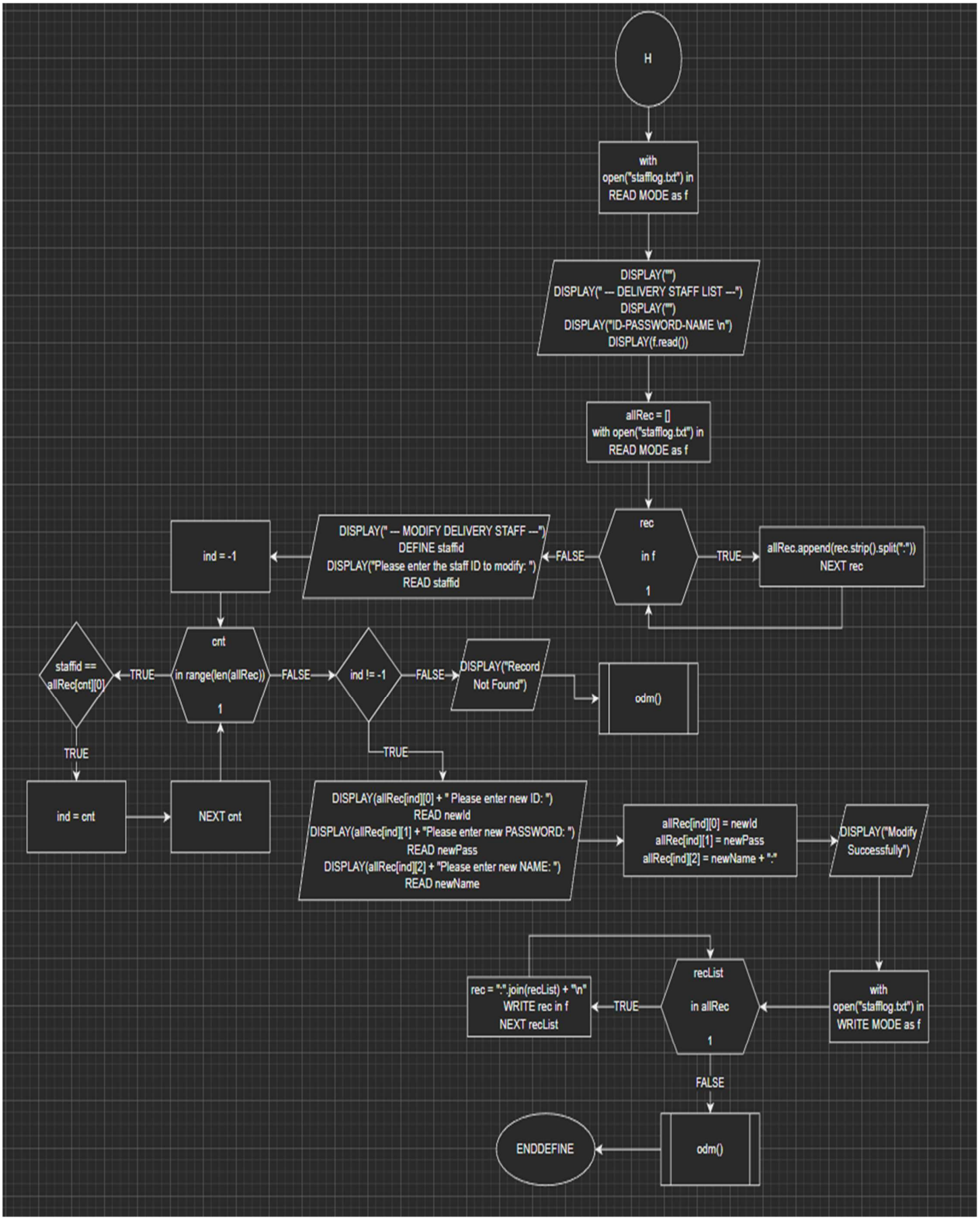
delivery_staff_system()

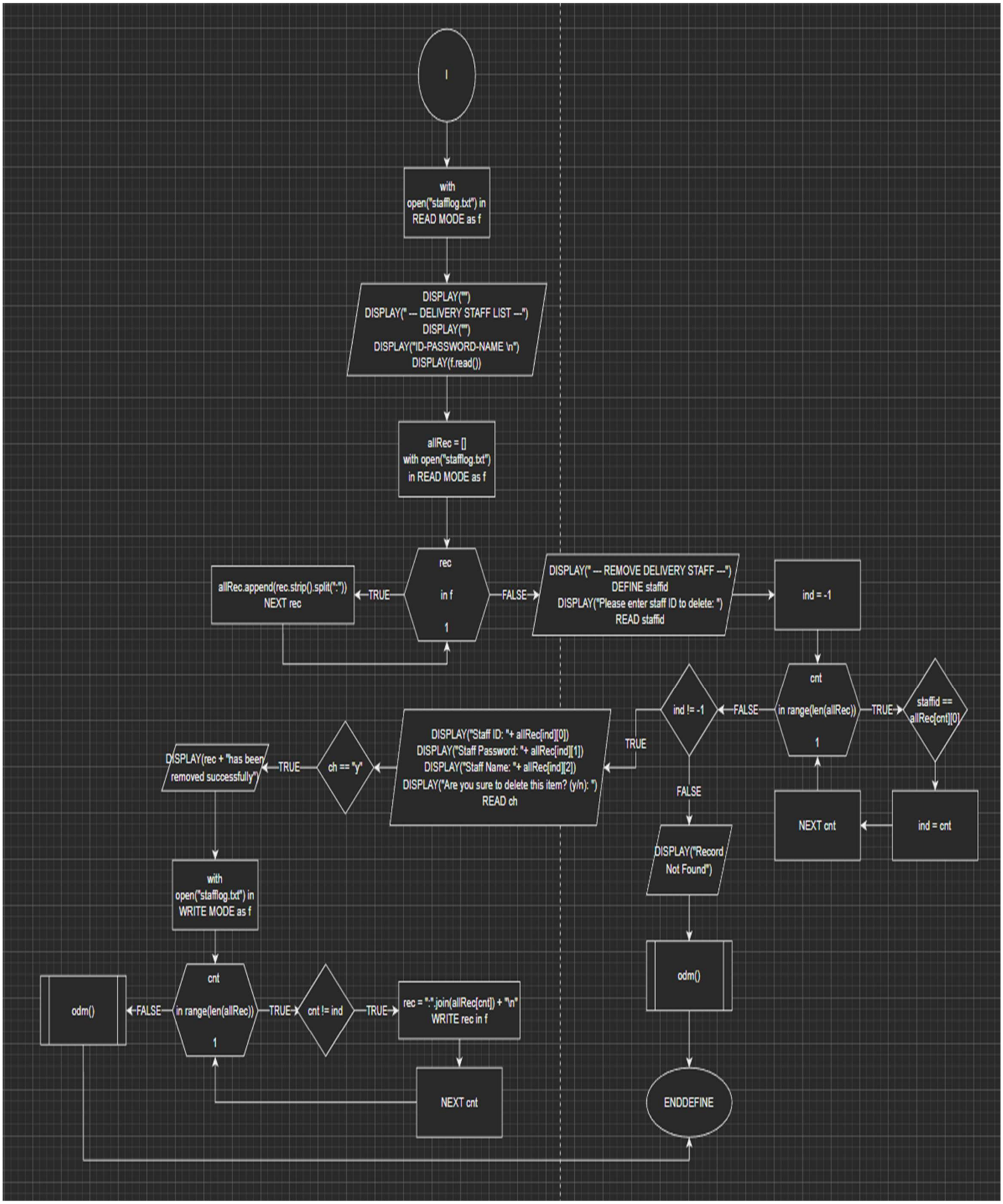


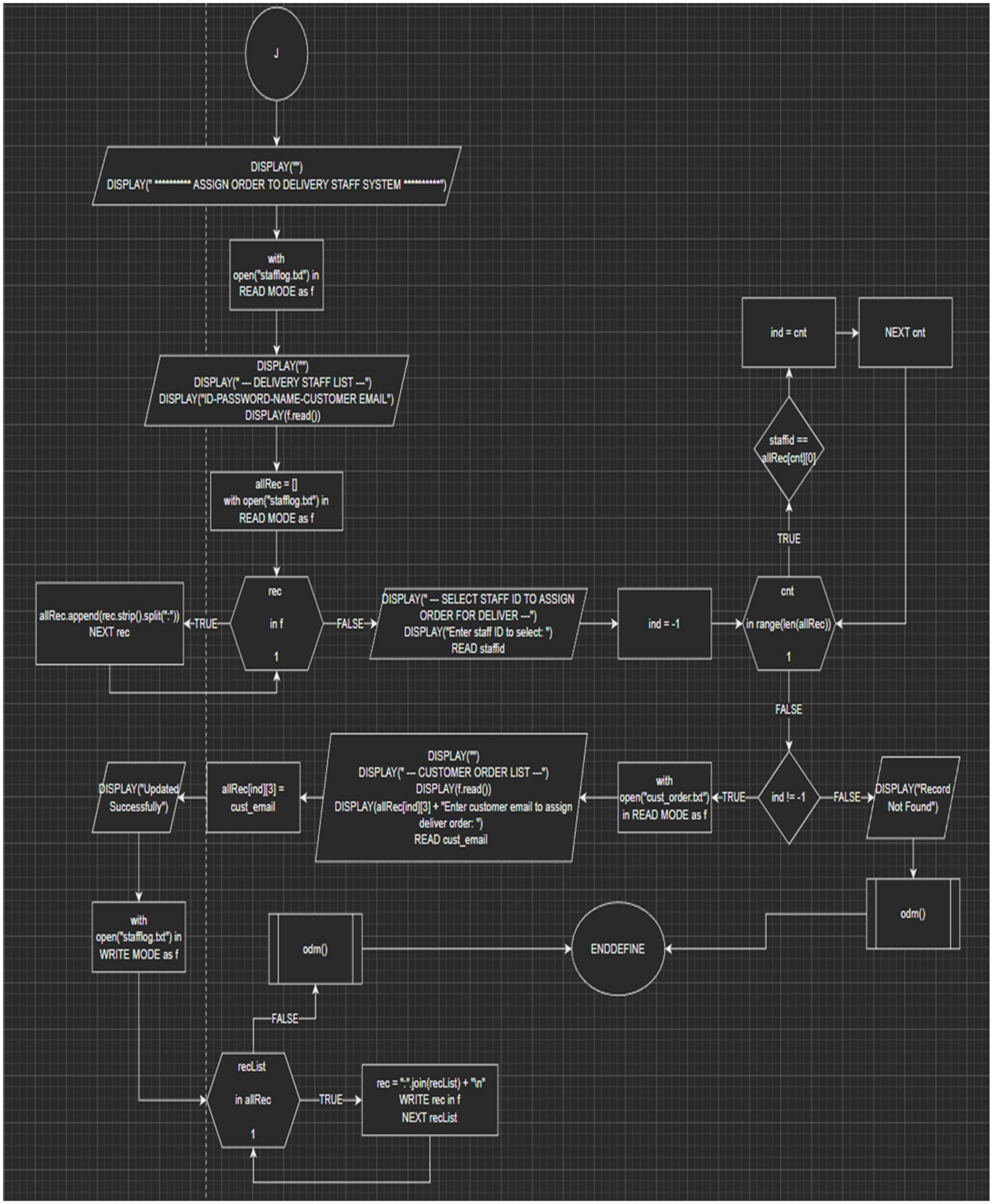

G
ind != -1
FALSE
DISPLAY("Record Not Found")
TRUE
DISPLAY("Do you sure you want to update the delivery status? (yes/no): ")
READ delivery_status
delivery_status.lower() == "yes"
FALSE
delivery_status.lower() == "no"
FALSE
DISPLAY("Wrong Input!")
TRUE
TRUE
DISPLAY(allRec[ind][3] + "Enter number to update the following status (1.Shipped, 2.Delivered 3.Cancelled): ")
READ ch
main_menu()
main_menu()
ch == 1
TRUE
ch = "Shipped"
allRec[ind][3] = ch
DISPLAY("Updated Successfully")
FALSE
ch == 2
TRUE
ch = "Delivered"
allRec[ind][3] = ch
DISPLAY("Updated Successfully")
FALSE
ch == 3
TRUE
ch = "Cancelled"
allRec[ind][3] = ch
DISPLAY("Updated Successfully")
FALSE
DISPLAY("Wrong Input! Please enter number 1 up to 3 to update following status")
with open("cust_order.txt") in WRITE MODE as f
recList
in allRec
1
TRUE
rec = "/".join(recList) + "\n"
WRITE rec in f
NEXT recList
delivery_staff_system()
ENDDEFINE

odm()


DEFINE odm
DISPLAY("")
DISPLAY(" --- ORDER DELIVERY MANAGEMENT SYSTEM --- \n")
DISPLAY("1. Add delivery staff")
DISPLAY("2. Modify delivery staff")
DISPLAY("3. Delete delivery staff")
DISPLAY("4. Assign orders to delivery staff")
DISPLAY("5. Exit")
DEFINE ch
DISPLAY("Enter your choice: ")
READ ch
ch == 1
TRUE
with open("stafflog.txt") in READ MODE as f
DISPLAY("")
DISPLAY(" --- DELIVERY STAFF LIST ---")
DISPLAY("")
DISPLAY("ID-PASSWORD-NAME \n")
DISPLAY(f.read())
FALSE
ch == 2
TRUE
H
with open("stafflog.txt") in APPEND MODE as f
DISPLAY(" --- ADD DELIVERY STAFF ---")
DISPLAY("Please enter new delivery staff ID:")
READ staffid
DISPLAY("Please enter new delivery staff PASSWORD:")
READ staffpass
DISPLAY("Please enter new delivery staff NAME:")
READ staffname
FALSE
ch == 3
TRUE
I
rec = staffid + ":" + staffpass + ":" + staffname
WRITE (rec + "\n") in f
DISPLAY(rec + " " + "has been successfully added")
FALSE
ch == 4
TRUE
J
ch == 5
TRUE
DISPLAY("Back to admin menu")
FALSE
DISPLAY("Wrong Input!")
admin_menu()
ENDDEFINE


H
with open("stafflog.txt") in READ MODE as f
DISPLAY("")
DISPLAY(" --- DELIVERY STAFF LIST ---")
DISPLAY("")
DISPLAY("ID-PASSWORD-NAME \n")
DISPLAY(f.read())
allRec = []
with open("stafflog.txt") in READ MODE as f
rec in f 1
TRUE
allRec.append(rec.strip().split(":"))
NEXT rec
FALSE
DISPLAY(" --- MODIFY DELIVERY STAFF ---")
DEFINE staffid
DISPLAY("Please enter the staff ID to modify: ")
READ staffid
ind = -1
cnt in range(len(allRec)) 1
staffid == allRec[cnt][0]
TRUE
FALSE
ind != -1
FALSE
DISPLAY("Record Not Found")
odm()
TRUE
ind = cnt
NEXT cnt
TRUE
DISPLAY(allRec[ind][0] + " Please enter new ID: ")
READ newId
DISPLAY(allRec[ind][1] + "Please enter new PASSWORD: ")
READ newPass
DISPLAY(allRec[ind][2] + "Please enter new NAME: ")
READ newName
allRec[ind][0] = newId
allRec[ind][1] = newPass
allRec[ind][2] = newName + ":"
DISPLAY("Modify Successfully")
with open("stafflog.txt") in WRITE MODE as f
recList in allRec 1
TRUE
rec = ":".join(recList) + "\n"
WRITE rec in f
NEXT recList
FALSE
odm()
ENDDEFINE


I
with open("stafflog.txt") in READ MODE as f
DISPLAY("")
DISPLAY(" --- DELIVERY STAFF LIST ---")
DISPLAY("")
DISPLAY("ID-PASSWORD-NAME \n")
DISPLAY(f.read())
allRec = []
with open("stafflog.txt") in READ MODE as f
rec in f
TRUE
allRec.append(rec.strip().split("."))
NEXT rec
FALSE
DISPLAY(" --- REMOVE DELIVERY STAFF ---")
DEFINE staffid
DISPLAY("Please enter staff ID to delete: ")
READ staffid
ind = -1
cnt in range(len(allRec)) 1
TRUE
staffid == allRec[cnt][0]
ind = cnt
NEXT cnt
FALSE
ind != -1
TRUE
DISPLAY("Staff ID: "+ allRec[ind][0])
DISPLAY("Staff Password: "+ allRec[ind][1])
DISPLAY("Staff Name: "+ allRec[ind][2])
DISPLAY("Are you sure to delete this item? (y/n): ")
READ ch
ch == "y"
TRUE
DISPLAY(rec + "has been removed successfully")
with open("stafflog.txt") in WRITE MODE as f
cnt in range(len(allRec)) 1
TRUE
cnt != ind
TRUE
rec = ":".join(allRec[cnt]) + "\n"
WRITE rec in f
NEXT cnt
FALSE
odm()
DISPLAY("Record Not Found")
odm()
ENDDEFINE


J
DISPLAY("")
DISPLAY(" ********** ASSIGN ORDER TO DELIVERY STAFF SYSTEM **********")
with open("stafflog.txt") in READ MODE as f
DISPLAY("")
DISPLAY(" --- DELIVERY STAFF LIST ---")
DISPLAY("ID-PASSWORD-NAME-CUSTOMER EMAIL")
DISPLAY(f.read())
allRec = []
with open("stafflog.txt") in READ MODE as f
rec in f 1
TRUE
allRec.append(rec.strip().split("-"))
NEXT rec
FALSE
DISPLAY(" --- SELECT STAFF ID TO ASSIGN ORDER FOR DELIVER ---")
DISPLAY("Enter staff ID to select: ")
READ staffid
ind = -1
cnt in range(len(allRec)) 1
TRUE
staffid == allRec[cnt][0]
ind = cnt
NEXT cnt
FALSE
ind != -1
TRUE
with open("cust_order.txt") in READ MODE as f
DISPLAY("")
DISPLAY(" --- CUSTOMER ORDER LIST ---")
DISPLAY(f.read())
DISPLAY(allRec[ind][3] + "Enter customer email to assign deliver order: ")
READ cust_email
allRec[ind][3] = cust_email
DISPLAY("Updated Successfully")
FALSE
DISPLAY("Record Not Found")
odm()
ENDDEFINE
odm()
with open("stafflog.txt") in WRITE MODE as f
recList in allRec 1
TRUE
rec = ":".join(recList) + "\n"
WRITE rec in f
NEXT recList
FALSE

4.0 Explanation of the Program Source Code

Source code is the instructions a programmer entered into the computer in a form that humans can understand. In order for a computer to comprehend and operate this code, it must first be translated from its original source format (the source code) into a machine-readable format (the machine code). Object Code is the name given to the transformed code. Some examples include variables, operators, loops, decision trees, managing errors, importing new code, working with files, and declaring new functions.

4.1 Explanation of Basic Operation in Program Source Code

Variables

A python variable is an area of memory set up specifically for storing data. Variables are used to keep track of the information the program needs to run. It is the fundamental building block of computer programs. Nonetheless, during program execution, the value stored in a variable may be altered. It is forbidden to use any characters other than the underscore (_) in a Python variable name, and the initial letter of a variable name must always be lowercase. It's recommended that programmers give the variable a name that has some relation to the information it stores int, str, float, bool, list, etc. are all examples of popular data types.

String

```
username = input("Enter Username: ")
password = input("Enter password: ")
password1 = input("Confirm password: ")
```

Figure 1

Username, Password, and Password1 are all required from the user in Figure 1 and saved to the corresponding variables. All possible character sets may now be used as input by the user.

```
item_added = input("Enter item code to place order: ")
```

Figure 1.1

As shown in Figure 1.1, the user must enter a string as an item code in a predetermined format such as A001, B002, etc. In this case, the string will include the numeric and special characters found in the variable item added.

Integer

```
card_no = int(input("Enter your " + payment_method + " card number:"))
```

Figure 1.2

Using int, the user is forced to enter data in the integer format in Figure 1.2. Numbers without a decimal point are the only format that will do. A single integer, which is all that will be included in the card number, is entered into the variable called card_no.

List

```
allRec = []
```

Figure 1.3

```
if (itmCode == allRec[cnt][0]):
```

Figure 1.4

A blank list object is generated in the variable content of Figure 1.3. Element strings are shown in Figure 1.4.

Arithmetic Operators

```
itmCode = input("Please enter new ITEM CODE:")
itmName = input("Please enter new ITEM NAME:")
itmType = input("Please enter new ITEM TYPE (Electronic Gadget/Home Appliance):")
itmQuantity = input("Please enter new ITEM QUANTITY:")
itmPrice = input("Please enter new ITEM PRICE:")
rec = itmCode+":"+itmName+":"+itmType+":"+itmQuantity+":"+itmPrice
```

Figure 1.5 Addition

As can be seen in Figure 1.5 above, the variables itmCode, itmName, itmType, itmQuantity, and itmPrice will all be combined into one single variable referred to as rec. Concatenation describes this operation.

```
new_qty = int(ori_qty) - item_qty
```

Figure 1.6 Subtraction

An integer subtraction is performed between the two numeric variables ori_qty and item_qty which store in a variable called new_qty

```
float(allRec[ind][4])*item_qty
```

Figure 1.7 Multiplication

As shown in Figure 1.7, the integer variable allRec[indv][4] will be converted into a floating-point value and multiplied by the integer variable item_qty.

Comparison Operators

```
if password == password1:
```

Figure 1.8 Equal sign

The "==" operator compares the value on the left side of the statement to the value on the right side, as seen in Figure 1.8. This is useful for comparing anything that can be represented by a string, including numeric values and Booleans.

```
if (ind != -1):
```

Figure 1.9 Not Equal Sign

The '! =' not equal to operator is used to determine whether the left operand which is the ind variable is not equal to the right operand value of -1 as shown in Figure 1.9. If they are not the same value, then it has to be True.

```
if (item_qty <= int(allRec[ind][3])):
```

Figure 2.0 Less than or Equal to Sign

In Figure 2.0 The less than equal to operator "<=" evaluates to true if the left operand item qty is less than or equal to the right operand int (allRec[ind][3]). Both operands must be integers, hence they must both be of the same data type.

Logical Operators

```
if (username == allRec[cnt][0]) and (password == allRec[cnt][1]):
```

Figure 2.1 And operators

According to Fig. 2.1, the and operator only returns True if both the right and left expressions evaluate to True. Otherwise, it will return False if any of the expressions is false.

String Operators

```
allRec.append(rec.strip().split(":"))
```

Figure 2.2 Strip() and Split()

As shown in Figure 2.2, the strip() operator is used to get rid of any extra spaces or blanks. The string is processed by the split() method, which then returns the resulting list. If a parameter is provided, it will be used to separate the items in the list.

```
for cnt in range(len(allRec)):
```

Figure 2.3 len() function

The len() method was used to determine how many items are shown in Fig. 2.3. The len() function takes a string as input and outputs the number of characters in that string.

```
if proceedpayment.lower() == "y":
    payment_method = input("Enter your payment method (Debit/Credit) card:")
    card_no = int(input("Enter your " + payment_method + " card number:"))
    cfm_payment = input("Comfirm payment? (yes/no) :")
```

Figure 2.4 IF

Figure 2.4 makes use of a basic if statement, which examines the condition that follows and, if True, executes the statements that follow. If the condition evaluates to False, the lines indented below the if statement will be executed before the main program flow resumes.

```
if (ind != -1):
    print("Logged in Successfully!")
    print("Welcome back", username)
    delivery_staff_system()
else:
    print("Login failed, credential not valid")
    main_menu()
```

Figure 2.5 condition of if and else

Referring to Figure 2.5, an if-else block is used when the flow should first verify the if statement's condition to see whether it is True, and then proceed to execute the lines intended below the if statement before returning the flow to the main code if the condition is True. If the 'if' statement's condition evaluates to false, the statements contained in the following else condition block are carried out instead.

```
if ch == 1:
    admin_login(adminList)
elif ch == 2:
    register()
elif ch == 3:
    login()
elif ch == 4:
    view_all_items()
elif ch == 5:
    delivery_staff_login()
elif ch == 6:
    print("end of program")

else:
    print("Wrong Choice! Please enter option 1-6!")
    main_menu()
```

Figure 2.6 condition of if, elif and else

Checking the 'if' condition comes first in Figure 2.6. If the condition is met, it runs the statements within the if block, and else it exits to the main menu(). If the condition is not met, control will go to the next 'elif' and examine its state. If the Boolean value is True, the predetermined statements are carried out. Finally, if the condition evaluates to false, the code in the else block will be run.

```
if delivery_status.lower() == "yes":
    ch = int(input(allRec[ind][3] + "Enter number to update the following status (1.Shipped, 2.Delivered 3.Cancelled): "))
    if ch == 1:
        ch = "Shipped"
        allRec[ind][3] = ch
        print("Updated Successfully")
```

Figure 2.7 Nested IF

If the condition in the main if statement in Figure 2.7 evaluates to false, then the statement does not proceed to the 'if' in the intended statement below. The block is entered, and the first if statement is performed if the condition is true. When the block's code has been fully executed, control is returned to the main program.

```
with open("cust_order.txt","r") as f:
    for rec in f:
        allRec.append(rec.strip().split("/"))
```

Figure 2.8 For Loop

This is shown by the usage of the for loop in Figure 2.8. A For loop allows you to quickly go through a list and access all of the objects' trivial parts. Range(x) may also be used with the for loop, with access granted to each element in turn. The instructions indented beneath for loop are performed whenever the for loop reads the file handle l and performs the loop where the value of each element is put in rec. Following the completion of the first code, the for-loop advances to the next value. After all values have been accessed, the loop ends.

```
proceedShopping = input("Do you wish to proceed (y/n): ").lower()
while (proceedShopping == "y"):
    item_added = input("Enter item code to place order: ")
```

Figure 2.9 While Loop

The while loop is shown in Figure 2.9. To begin processing, a while loop is initiated with an initialization value. The while loop's condition is then assessed, and if it evaluates to true, the sentence indented below the while loop is carried out. Until the counter reaches the maximum value, the while loop will continue to execute. At the point when the while condition evaluates to false, the while loop terminates.

File Handling

```
with open("item_Details.txt", "r")as f:
    print(f.read())
```

Figure 3.0 open file for reading mode

The text file "item Details" is opened in Figure 3.0 with the read mode set to "r," which indicates that the file will be opened for the purpose of reading. When we want to open the file, we use the with open statement, and the file is opened using the "f" file handle. In read mode, you are able to carry out a variety of operations, such as print(f.read()), which prints the information that extracted from the item Details.txt file.

```
with open("All_items.txt","w") as f:
    for recList in allRec:
        rec = ":".join(recList) + "\n"
        f.write(rec)
```

Figure 3.1 open file for writing mode

The file is opened in the write mode in Figure3.1. However, one must exercise caution while using this function , since opening a file in the write mode will cause any prior data in the list to be overwritten. Opening the file in the write mode is shown in Figure3.1. If a file does not already exist, it may be used to create it if the appropriate mode is set to write. The data may

be written into the text file with the usage of the write () method. The file handle is currently located at the beginning of the file.

```
with open("credentials.txt", "a") as f:
    rec = username +":"+password
    f.write(rec +"\n")
    print("You have registered successfully!")
```

Figure 3.2 open file for appending mode

It is possible to append data to a text file by using the file that was opened in the append mode as shown in Figure 3.2. The handle of the file is located at the very end of the text file. It is possible to utilize it to write or insert new data into the text file at the end without running the danger of overwriting the data that is already present in the file.

Datetime

```
import datetime
cfm_payment = "paid" + ", " + str(datetime.datetime.now())
```

The datetime module used by the Python programming language consists of a single module. This indicates that it is not composed of two distinct sorts of data. You are able to import this datetime module in such a manner that it allows for the manipulation of dates and times. datetime is a module that comes pre-installed with Python. It is not necessary for developers to install it in a separate environment. It's not hard to find.

Applying the datetime module in the program to store the datetime of the customer payment history when the customer enter the input of “yes” in the variable of cfm_payment it will store the datetime in the cust_payment.txt file in the current time by using datetime.datetime.now().

Functions of the program

1) admin_login():

Admin login system, allows admin user to login the system through this function

2) admin_menu():
   This function is an admin menu which contain the accessibility of admin and the menu will display all choice that admin user can access.

3) register():
   Register page, allows users to register an new account within this function to further access other functions.

4) login():
   A login page function to allows registered users access to view_all_items() function and place_order() function.

5) view_all_items():
   This function allows all user include unregistered user to view the available items in store.

6) place_order():
   This function allows registered user to place an order according to the stock quantity and after placed an order the respective item quantity will be deducted from the store. This function also allows the user to make payment after the user proceed to the payment.

7) delivery_staff_login():
   Delivery staff login page, allows the delivery staff user to login and access to delivery_staff_system() function.

8) add_item_category():
   Allow admin to add new item to the store.

9) modify_item():
   Allow admin to modify item that present in the store.

10) remove_item():

Allow admin to remove item that present in the store.

11) view_search_cust_order():

This function allow admin to view customer order and access to search_cust_order() function.

12) search_cust_order():

This function allow admin to search specific customer order by entering the customer email.

13) view_search_cust_payment():

This function allow admin to view customer payment and search specific customer payment by entering the customer email.

14) delivery_staff_system():

Allow the delivery staff to update the customer order's delivery status by entering customer email to select and update.

15) odm():

This function is called order delivery management which simply use for the admin user to add, modify, remove the delivery staff and as well as the assign order function to delivery staff. Within this odm() function it allows the admin user to choose the options that mentioned above to access.

16) main_menu():

Main menu page, display the main menu to all the user from start of the program.

4.2 Explanation of the Concept of Program Source Code

```
#admin login page
def admin_login(adminList):
    print("")
    print(" --- ADMIN LOGIN PAGE ---")
    username = input("Enter your Username: ")
    password = input("Enter your Password: ")
    allRec = []
    ind = -1
    with open("adminlog.txt", "r") as f:
        for rec in f:
            allRec.append(rec.strip().split(":"))
    for cnt in range(len(allRec)):
        if (username == allRec[cnt][0]) and (password == allRec[cnt][1]):
            ind = cnt
    if (ind != -1):
        print("***************** ADMIN SYSTEM ***************** \n")
        print("Logged in Successfully!")
        print("Welcome back",username)
        print("")
        admin_menu()
    else:
        print("Login failed, credential not valid")
        main_menu()
```

Figure 4 Admin Login

The design in figure 1 begins with the administrative sign-in screen. The text file adminlog.txt serves the purpose of logging in as the admin user account. The print() method will be used to generate the login page's headline, and the input() function will be used to receive the user's username and password. Eventually, we'll call as f and open the adminlog.txt files using the read function (file). As a further step, we'll use a for loop to repeatedly process the file. In the loop the data that include in the adminlog.txt file will be separated in (:) by using add, strip and split function. By using strip, all extraneous characters or spaces will be omitted (). Within the list, a for loop in the range of the words as 'for cnt in range(len(allRec))' will be used once the data in adminlog.txt has been separated. To check whether the supplied username and password are legitimate, we use the if function within the loop, checking to see if (username == allRec[cnt][0]) and (password == allRec[cnt][1]) return true. If all of these conditions hold true, the program will execute print() and load the administration menu page; otherwise, it will execute print(login failed) and return to the main menu.

```
#admin menu
def admin_menu():
    print(" --- ADMIN MENU ---")
    print("1. Add item category")
    print("2. Modify item")
    print("3. Delete item")
    print("4. View/Search customer order")
    print("5. View/Search customer payment" )
    print("6. Order Delivery Management")
    print("7. Exit")
    ch = int(input("Enter your choice: "))
    if ch == 1:
        add_item_category()
    elif ch == 2:
        modify_item()
    elif ch == 3:
        remove_item()
    elif ch == 4:
        view_search_cust_order()
    elif ch == 5:
        view_search_cust_payment()
    elif ch == 6:
        odm()
    elif ch == 7:
        print("End of program \n")
        main_menu()
    else:
        print("Wrong input!")
        print("Wrong choice! \n")
        main_menu()
```

Figure 5 Admin Menu

The second design in figure 2 function, admin menu(), is used to access the administration panel. Adding a new category of items, editing an existing one, removing an item, seeing and searching orders, viewing and searching payments, managing orders, and exiting are all options inside the administration menu. At the outset of the process, we'll generate a menu that the administrator may access through print (). The input() method then allows the user to choose a value between 1 to 7. The if function will be used to determine whether or not to provide the user access to the other functions after reading their input as ch. Incorrect input will result in a printout of "WrongInput!" and a redirect to the menu home page.

```
#register page
def register():
    print("")
    print("--- REGISTER PAGE ---")
    username = input("Enter Username: ")
    password = input("Enter password: ")
    password1 = input("Confirm password: ")

    if password == password1:

        with open("credentials.txt", "a") as f:
            rec = username +":"+password
            f.write(rec +"\n")
            print("You have registered successfully!")
        login()
    else:
        print("Password is not same as above! \n")
        register()
```

Figure 6 Register Page

In this part is the register function that shown in figure 3 which allow user to register an account for access to the system lately. In the top of this function begin with print() to display the register page banner. After the print function the user will be asked for their username, password and password1 for registering a new account by using input(). The password1 is for a confirmation of password which can use for running the function with if function in a word of if password == password1, in this logic whenever the user password is equal to the password1 will proceed to open credentials.txt file as append which is a file to store user username and password. The username and password will be appended into the file f.write(rec +"\n") as username : password. If the password1 not equal to password it will print(Password is not same as above!) and go back to register page.

```
#login page and access to view all item details and place order function
def login():
    print("")
    print("--- LOGIN PAGE ---")
    username = input("Enter your Username: ")
    password = input("Enter your Password: ")
    allRec = []
    ind = -1
    with open("credentials.txt", "r") as f:
        for rec in f:
            allRec.append(rec.strip().split(":"))
    for cnt in range(len(allRec)):
        if (username == allRec[cnt][0]) and (password == allRec[cnt][1]):
            ind = cnt
    if (ind != -1):
        print("")
        print("Logged in Successfully!")
        print("Welcome Back," + username)
        print("Happy Shopping!")
        print("")
        print(" --- Welcome to Jason and Dennis ALL IT APU Online Shopping Mall ---")
        print("1. View all item details")
        print("2. Place order")
        print("3. Exit")
        ch = int(input("Enter your choice: "))
        if ch == 1:
            with open("item_Details.txt", "r")as f:
                print(f.read())
                placeorder = input("Do you want to place an order? (yes/no): ")
                if placeorder == "yes":
                    place_order()
                elif placeorder == "no":
                    print("Back to main menu!")
                    main_menu()
                else:
                    print("Wrong Input!")
                    main_menu()

        elif ch == 2:
            place_order()
        elif ch == 3:
            print("End of program \n")
            main_menu()
        else:
            print("Wrong input!")
            main_menu()
    else:
        print("Login failed \n")
        main_menu()
```

Figure 7 Login Page

This section serves as a login page as shown in figure 4, allowing registered users to see product information and, more recently, the ability to make orders. Because of this, the logic of the login system is quite similar to the admin login logic described in the first explanation. After the user logs in successfully, the program prints a menu with three options ('1', '2, and '3) and prompts them to choose one using input(). If the user selects option (1), the programme opens the item Details.txt file in read mode, where it lists every item in the shop and its prices. When the file is read, the item information is printed. Then it will prompt the customer to continue with the ordering process. If the user clicks "yes" here, they will be sent to the order placement screen; clicking "no" will cancel the action and return them to the main menu. Should the input not fall between 1 and 2, a "Wrong Input!" message and a return to the main menu will be shown. When the function prompts for the user's input, if they enter "2," they'll be sent straight

to the order placement page; if they enter "3," they'll be taken back to the main menu. If the user's selections fall outside the offered options, the function will exit with a "print()" message.

```
#view all items page and access to register page
def view_all_items():
    print("")
    print(" --- VIEW ALL ITEM PAGE ---")
    print("")
    f=open("Category_Details.txt", "r")
    print(f.read())
    f.close()
    print("")
    f=open("All_items.txt", "r")
    print(f.read())
    f.close()
    print("")
    print("1.Registered")
    print("2.Unregistered")
    print("3.Back")
    print("")
    ch = int(input("Please enter your choice:"))
    if ch == 1:
        login()
    elif ch == 2:
        print("")
        print("1. Register to access more details")
        print("2. Exit")
        ch = int(input("Please enter your choice:"))
        if ch == 1:
            register()
        else:
            print("Exit to main menu")
            main_menu()

    elif ch == 3:
        print("Back to main menu")
        main_menu()
    else:
        print("Wrong Input \n")
        main_menu()
```

Figure 8 View All Items

All users, whether are registered or not, should be able to browse the store's inventory using this feature. The design began with the function print(), which created the first screen. Files named category_details.txt and all_items.txt will open when use file = open() with the read and print(f.read()) functions, respectively, so you can see the items in each category and all of the items in stock. After using f.open to open a file, we must then use f.close() to close it. Close the file to see a menu asking whether you're a registered user or not when print() returns. There will be three choices available here, the first of which, "registered," will prompt the user to confirm their registration status before redirecting them to the login page. Option two,

"unregistered," displays a menu instructing the user to sign up for an account in order to see the rest of the options and clicking that menu's "2" button takes them to the registration page. If a person is simply looking around and doesn't want to sign up, they may return to the main menu by selecting "2." The third selection is an exit button that takes the user back to the main menu. If the user selects a number that out of the range from 1–3, the options will print() and the user will be returned to the main menu.

```
#allow registered user to place order
def place_order():
    shoppingDict = {}

    my_file = open("All_items.txt")
    file_line = my_file.readline()
    itemsAvailable = my_file.readlines()
    my_file.close()

    email = input("Enter your email address:")
    phone_no = input("Enter your phone number:")
    print("")
    print("********** Items Available in Our Store ********** \n")
    print("Item Code: Item Name: Item Type: Item Quantity: Item Price")
    for item in itemsAvailable:
        itmCode = item.split(":")[0]
        itmName = item.split(":")[1]
        itmType = item.split(":")[2]
        itmQuantity = item.split(":")[3]
        itmPrice = item.split(":")[4]

        print(f"{itmCode}: {itmName}: {itmType}: {itmQuantity}: {itmPrice}")


    allRec = []
    with open("All_items.txt", "r") as f:
        for rec in f:
            allRec.append(rec.strip().split(":"))
    proceedShopping = input("Do you wish to proceed (y/n): ").lower()
    while (proceedShopping == "y"):
        item_added = input("Enter item code to place order: ")
        ind = -1
        for cnt in range(len(allRec)):
            if (item_added in allRec[cnt][0]):
                ind = cnt
        if (ind != -1):
            item_qty = int(input(allRec[ind][0] +":"+ allRec[ind][1] +":"+ allRec[ind][2] +":"+ allRec[ind][3] +":"+ allRec[ind][4] +" >"+ " Enter quantity: "))
            shoppingDict.update({"Item: "+item_added:{"quantity":item_qty,"subtotal(RM)":float(allRec[ind][4])*item_qty}})
            print(shoppingDict)
            ind1 = -1
            for cnt1 in range(len(allRec)):
                if (item_qty <= int(allRec[ind][3])):
                    ind1 = cnt1
            if (ind1 != -1):
                ori_qty = allRec[ind][3]
                new_qty = int(ori_qty) - item_qty
                allRec[ind][3] = str(new_qty)
                with open("All_items.txt","w") as f:
                    for recList in allRec:
                        rec = ":".join(recList) + "\n"
                        f.write(rec)
                    print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ADD CART SUCCESSFUL~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
```

Figure 9 Place Order

In Figure 9 shown that function of place order, in this function it allows the registered customer to add multiple items in cart by simply enter the item code that available in store. Begin with this function by define the shoppingDict variable as dictionary. After that, open All_items.txt as my_file which the file is used for storing the information of the available item in the store, the reason to open the All_items.txt file is to read the lines by using define my_file as myfile.readlines() function. Subsequently, the user will prompt to enter the email address and

phone number before placing an order, after the using input() the email address and phone number. It will print() the all items into well separate item list by using the for loop, split and print method. I will define the item variable in itemAvailable in a for loop and split the item by using split(":"), after splitting all the item in itemAvailable, the for loop will keep on doing until the end of the list in itemAvailable and it will display all the item details by using print(). The user will be getting an input permission of proceed to place an order within the choices of (y/n). If the user enter "y" which mean yes in short form, it go through a while loop, within the while loop the user will be ask to enter the item code that wanted to purchase, and after placing an order, it will ask the user to enter the amount of quantity of the respective item.

```
        else:
            print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~FAIL TO ADD CART~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
            print("Item stock quantity not enough!!")
            print("Please re-enter the quantity of product needed according to our stock quantity.")
            print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~PLEASE PLACE ORDER AGAIN~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
            place_order()
    else:
        print("Item not found. Please try again.")
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~GOING BACK ORDER PAGE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        place_order()
```

Figure 10

In this part, if the user enter the item quantity that exceed from the stock quantity in store will be display a message of "FAIL TO ADD CART" and "Item stock quantity not enough!" and send the user back to place_order() again that shown in figure 10.

```
proceed_Shopping = input("Do you wish to add more items(y/n): ").lower()
if (proceed_Shopping == "y"):
    continue
else:
    print("\n\n")
    print("*******************************Bill Summary*******************************")
    print("***************JASON AND DENNIS ALL IT APU ONLINE SHOPPING MALL***************")
    print("Item\t\t\t\tQuantity\t\t\tSubtotal")
    total = 0
    for record in shoppingDict:
        print(f"{record}\t\t\t{shoppingDict[record]['quantity']}\t\t\t{shoppingDict[record]['subtotal(RM)']}")
        total = shoppingDict[record]['subtotal(RM)']+total
        print(f"Total: {total}")
    proceedpayment = input("Do u wish to proceed payment (y/n):")
    if proceedpayment.lower() == "y":
        payment_method = input("Enter your payment method (Debit/Credit) card:")
        card_no = int(input("Enter your " + payment_method + " card number:"))
        cfm_payment = input("Comfirm payment? (yes/no) :")
        if cfm_payment.lower() == "yes":
            with open("cust_order.txt", "a") as f:
                rec = email + "/" + phone_no + "/" + str(shoppingDict) + "/"
                f.write(rec + "\n")
            cfm_payment = "paid"
            with open("cust_payment.txt", "a") as f:
                info = email + ", " + phone_no + ", " + str(total) + ", " + payment_method + ", " + str(card_no) + ", " + cfm_payment
                f.write(info + "\n")

            print("")
            print("********** Thank You **********")
            print("Hope to see you back soon!")
            main_menu()
        else:
            print("Exit to main menu")
            main_menu()

    else:
        print("Back to main menu")
        main_menu()
    break
```

Figure 11 Make Payment

In figure 11, the user will be receive the bill summary receipt that according to the item and prices that calculate from the function using multiplication method. The receipt will be printed out and according to the item code following with the item quantity that placed and the subtotal of the item. After receiving the bill summary, the user will be prompt to enter the proceed payment input within the choice of (y/n). If the user enter the option of “y” , the user will need to be enter the payment method and card number. After entered all the input(), the user will be once again receive an input() to confirm payment within the choices of (yes/no). If “yes” the order info will be stored into the cust_order.txt file and the payment info will be stored into the cust_payment.txt.

```
#delivery staff login page
def delivery_staff_login():
    print("")
    print(" --- LOGIN PAGE FOR DELIVERY STAFF ---")
    username = input("Enter your username:")
    password = input("Enter your password:")
    allRec = []
    ind = -1
    with open("stafflog.txt", "r") as f:
       for rec in f:
          allRec.append(rec.strip().split(":"))
    for cnt in range(len(allRec)):
       if (username == allRec[cnt][0]) and (password == allRec[cnt][1]):
          ind = cnt
    if (ind != -1):
        print("Logged in Successfully!")
        print("Welcome back", username)
        delivery_staff_system()
    else:
        print("Login failed, credential not valid")
        main_menu()
```

Figure 12 Delivery Staff Login

There's a login section for the delivery staff showing in figure 7, using the same logic as the admin login section in the first explanation. Input is used after an initial call to print() to ask for an staff ID and password (). Open the file using open("stafflog.txt") in read mode to see the staff IDs and passwords it contains. If the username and password the staff enters match those in the text file, the message "Logged in Successfully" and "Welcome back, username" are printed and the user is granted access to the delivery staff system. “Login failed, credential not valid” will be display and return to previous page which is the main menu page if any one of the criteria are not satisfied.

```
#allow admin to add items in inventory
def add_item_category():
    with open("All_items.txt", "r") as f:
        print("")
        print(" " + "--- ITEM LIST ---")
        print("")
        print("Code : Item : Type : Quantity : Price(RM) \n")
        print(f.read())
    with open("All_items.txt", "a") as f:
        print(" --- ADD ITEM CATEGORY ---")
        itmCode = input("Please enter new ITEM CODE:")
        itmName = input("Please enter new ITEM NAME:")
        itmType = input("Please enter new ITEM TYPE (Electronic Gadget/Home Appliance):")
        itmQuantity = input("Please enter new ITEM QUANTITY:")
        itmPrice = input("Please enter new ITEM PRICE:")
        rec = itmCode+":"+itmName+":"+itmType+":"+itmQuantity+":"+itmPrice
        f.write(rec + "\n")
        print(rec + " " + "has been successfully added")
    admin_menu()
```

Figure 13 Add Item Category

The administrator may add new item category in the system by using the "add item category" feature, which requires just the entry of the new item's code, name, type, quantity, and price. The function begins by reading the contents of a text file named all items.txt into memory in read mode. This file describes all of the products sold by the shop, including prices and other important information for customers. Using print(f.read()), which is produced after opening the text file in read mode, we may display the information contained in all item.txt to the administrator. The all items.txt file will be reopened in append mode after displaying the list of item data to the admin user. In this setting, the admin user's input() is written as with open("All items.txt", "a") as f: after opening the text file, the admin user is prompted to enter the new item information that were determined to include into inventory through input() (). After the admin user enters new item details, the details are linked into a rec variables using +":" to link all the item details that admin user just entered, the rec variables are written and appended into the text file using f.write(rec + "\n"), and the programme then prints (rec + " " + "has been successfully added") to let the admin user know that the new item details were added successfully and return them to the admin menu.

```
#allow admin to modify items in inventory
def modify_item():
    with open("All_items.txt" ,"r") as f:
        print("")
        print(" " + "--- ITEM LIST ---")
        print("")
        print("Code : Item : Type : Quantity : Price(RM) \n")
        print(f.read())

    allRec = []
    with open("All_items.txt","r") as f:
        for rec in f:
            allRec.append(rec.strip().split(":"))
    print(" --- MODIFY ITEM ---")
    itmCode = input("Please enter the ITEM CODE to modify: ")
    ind = -1
    for cnt in range(len(allRec)):
        if (itmCode == allRec[cnt][0]):
            ind = cnt
    if (ind != -1):
        newName = input(allRec[ind][1] + " Please enter new name: ")
        newType = input(allRec[ind][2] + "Please enter new type(Electronic Gadget/Home Appliance: ")
        newQuantity = input(allRec[ind][3] + "Please enter new quantity: ")
        newPrice = input(allRec[ind][4] + "Please enter new price: ")
        allRec[ind][1] = newName
        allRec[ind][2] = newType
        allRec[ind][3] = newQuantity
        allRec[ind][4] = newPrice
        print("Modify Successfully")
    else:
        print("Record Not Found")
        admin_menu()

    with open("All_items.txt","w") as f:
         for recList in allRec:
             rec = ":".join(recList) + "\n"
             f.write(rec)
    admin_menu()
```

Figure 14 Modify Item

Figure 9 show that the function of modify item which allow the admin to modify items in inventory based on the item code. Begin with this function with open all_item.txt file for reading mode and the particular file as f. After open the item file will display the item list by using print() function f.read(). After display the item list from the item text file, the text file will once open again for reading mode as f (file), the reason why opens the file as reading mode is to read all the records into a list of a lists. Because once the data is in the list, it can go through the data very easily. Subsequently, a variable name as allRec will be created as an empty list initially and once the file opened, it going to read all the record into the list as allRec In this particular time, the loop function will be used in here, by using for loop and written as for rec as variable in all_item.txt file and the rec will be one entire line which store in the control variable name as rec and that line is going to be a string. In this case, the string will be convert into a list by using strip and spilt method, the rec variable will be appended into the list name as allRec and split by ":". Once the loop is ended and the file closed. The

admin user will prompt to enter the item code to modify the item that wanted to modify. After receiving the user input(), I will going to define the index value as -1 and go through a for loop and the for loop I will using a cnt variable in the range function as in length of allRec and compare the item code value that enter by the user just now to the all_items text file by using if function and written as if (itmCode == allRec[cnt][0]): whenever the itmCode value that entered from the admin user is present in the allRec[cnt][0] and it means that the record have found that need to be modify. And the cnt value is going to store into ind. In this time, the loop will be ended, after coming out from the for loop. The system going to check the ind is not equal to -1 anymore means that the record have been found and the function will let the admin user to modify the item name, type, quantity and the price by using input() function. After the admin user modified and entered all the new info. The entered info will be store into allRec[ind][] respectively to replace the old value into new value. After replacing the old value into new value, the system will open back the all_items.txt file and this time will be using “w” mode which is the overwrite mode as f (file) and using for loop for recList in allRec and within the for loop the recList variable will join into rec in “:” and add a new line character at the end that will be written as rec = ":".join(recList) + "\n". After join the recList into rec, the rec will be write into the file and it will keep on doing until it reach the end of the list and all the records will be written back.

```python
#allow delivery staff to select customer order and update the delivery status
def delivery_staff_system():
    print("")
    print(" --- DELIVERY STAFF SYSTEM ---")
    with open("cust_order.txt" ,"r") as f:
        print("")
        print(f.read())

    allRec = []
    with open("cust_order.txt","r") as f:
        for rec in f:
            allRec.append(rec.strip().split("/"))
    print(" --- SELECT CUSTOMER ORDER TO UPDATE DELIVERY STATUS ---")
    cust_email = input("Enter customer email to select: ")
    ind = -1
    for cnt in range(len(allRec)):
        if (cust_email == allRec[cnt][0]):
            ind = cnt
    if (ind != -1):
        delivery_status = input("Do you sure you want to update the delivery status? (yes/no): ")
        if delivery_status.lower() == "yes":
            ch = int(input(allRec[ind][3] + "Enter number to update the following status (1.Shipped, 2.Delivered 3.Cancelled): "))
            if ch == 1:
                ch = "Shipped"
                allRec[ind][3] = ch
                print("Updated Successfully")
            elif ch == 2:
                ch = "Delivered"
                allRec[ind][3] = ch
                print("Updated Successfully")
            elif ch == 3:
                ch = "Cancelled"
                allRec[ind][3] = ch
                print("Updated successfully")
            else:
                print("Wrong Input! Please enter number 1 up to 3 to update following status")
                delivery_staff_system()

        elif delivery_status.lower() == "no":
            main_menu()
        else:
            print("Wrong Input!")
            main_menu()


    else:
        print("Record Not Found")
        delivery_staff_system()

    with open("cust_order.txt","w") as f:
            for recList in allRec:
                rec = "/".join(recList) + "\n"
                f.write(rec)
```

Figure 15 Delivery Staff System

As shown in Figure 10, the system's delivery staff function enables them to keep customers up to date on the progress of their orders by inputting the customers' email addresses. Using this idea, the function first opens the "cust order.txt" file in reading mode as f (file), then uses the print() function to show the customer order history from the text file by printing out the result of the f.read() operation. In order to update the appropriate customer order, the user will then be prompted to input the customer email into the cust_email variable. Once the cust email that entered by the user is present in the allRec[cnt][0], which means that the record have found to be update the delivery status, the if function will be written as if (cust email ==

allRec[cnt][0]) and the index value will be defined as -1. The for loop will use a cnt variable in the range function as in length of allRec. The cnt value will be saved in the ind field. In this time after exiting the for loop, the loop will be finished. If the ind variable is not equal to -1, the record has been located, and the function will ask the delivery staff user through input() whether to continue updating the delivery status of the client order or not. In the interim, if the input is affirmative, the user on the delivery team will be prompted to update the delivery status of the following customer order to one of three possible states—"1.Shipped," "2.Delivered," or "3.Cancelled"—by entering the corresponding number in the input() function. After the delivery staff user input all the new information, the delivery status was updated. Information supplied here will be saved in allRec[ind][3] and used to modify the customer's order delivery. After the delivery status has been updated, the system will reopen the cust order.txt file, but this time it will do so in "w" mode, which is the overwrite mode of "f" (file), and it will use a for loop for recList in allRec; inside this loop, the recList variable will join into rec in "/," and a new line character will be appended to the end, as written: rec = ":".join(recList). The rec will be written back into the file after joining the recList into rec, and this process will continue until the end of the list is reached.

5.0 The Sample Explanation of Input and Output of the System

FIRST MAIN MENU

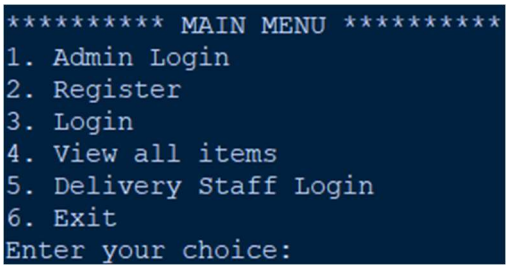


Fig 5.1

When the application first starts, the first primary menu will appear, as shown in Fig. 5.1 above. Different user types are presented with 6 alternatives. The admin can alter the item, quantity, pricing, and other information using the first option "1." For those who haven't created an account yet, option "2" is the second option. The registered consumer can log in using their registered account as the option "3." For access to the website where you may examine all your things as well as register, choose option "4". Staff who deliver updates is the first choice, "5". The last choice, "6," is to exit the program.

ADMIN LOGIN

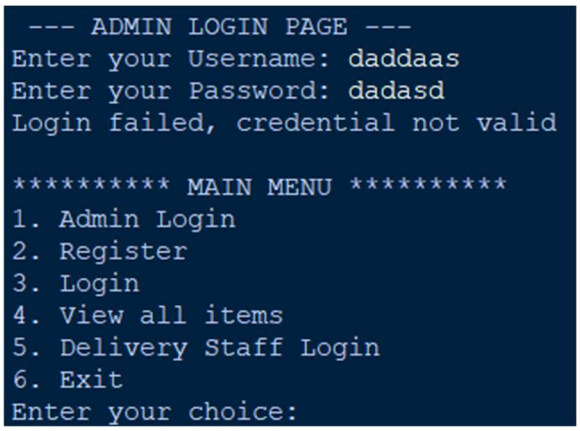


Fig **5.2**

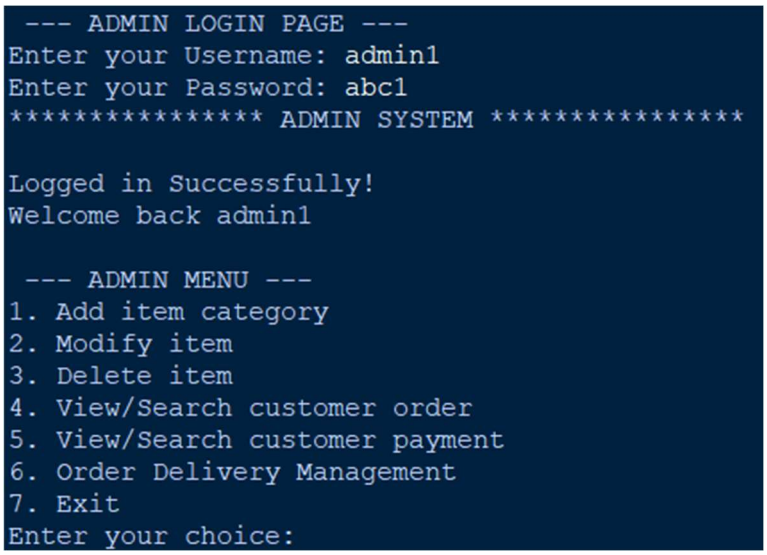


Fig **5.3**

To stop non-admins from entering, the admin login page is displayed when you select option "1" in the admin menu. To log in to the account, the administrator must enter their username and password. Looking at Fig. 5.2, you can see that if you enter the erroneous username and password, the login will fail and take you back to the main screen. If you enter the correct username and password, the login will be successful, and the Admin Menu will appear. The first choice, "1," is for adding an item category, and the second, "2," is for modifying an item. The Admin Menu has seven different options, as shown in Fig. 5.3. The delete item option is the third option "3". The fourth option, "4", allows you to view or search a customer's order. Next, "5" allows you to view or search a customer's payment. Next, "6" allows you to use order delivery management. Finally, "7" allows you to exit the program.

Add Item Gategory

```
 --- ITEM LIST ---

Code : Item : Type : Quantity : Price(RM)


001:Iphone14:Electronic Gadget:7:5000
002:Iphone13:Electronic Gadget:2:4000
003:Samsung:Electronic Gadget:0:3400
004:Huawei:Electronic Gadget:5:2300
005:Robot Vacuum:Home Appliance:5:700
006:Air Purifier:Home Appliance:2:400
007:Smart Fan:Home Appliance:5:200
008:Smart Camera:Home Appliance:5:350
009:Smart Door:Home Appliance:2:930
010:Smart TV:Home Appliance:5:2200
011:Ipad:Electronic Gadget:9:1500
012:Stereo Speaker:Electronice Gadget:5:200
013:Smart Watch:Electronic Gadget:5:150
014:Monitor:Electronic Gadget:4:200
015:Printer:Electronic Gadget:5:250
016:Headphone:Electronic Gadget:3:50
017:Airpod:Electronic Gadget:2:750

 --- ADD ITEM CATEGORY ---
Please enter new ITEM CODE:018
Please enter new ITEM NAME:Apple
Please enter new ITEM TYPE (Electronic Gadget/Home Appliance):Electronnic Gadget
Please enter new ITEM QUANTITY:9
Please enter new ITEM PRICE:3000
018:Apple:Electronnic Gadget:9:3000 has been successfully added
```

Fig **5.4**

```
 --- ITEM LIST ---

Code : Item : Type : Quantity : Price(RM)


001:Iphone14:Electronic Gadget:7:5000
002:Iphone13:Electronic Gadget:2:4000
003:Samsung:Electronic Gadget:0:3400
004:Huawei:Electronic Gadget:5:2300
005:Robot Vacuum:Home Appliance:5:700
006:Air Purifier:Home Appliance:2:400
007:Smart Fan:Home Appliance:5:200
008:Smart Camera:Home Appliance:5:350
009:Smart Door:Home Appliance:2:930
010:Smart TV:Home Appliance:5:2200
011:Ipad:Electronic Gadget:9:1500
012:Stereo Speaker:Electronice Gadget:5:200
013:Smart Watch:Electronic Gadget:5:150
014:Monitor:Electronic Gadget:4:200
015:Printer:Electronic Gadget:5:250
016:Headphone:Electronic Gadget:3:50
017:Airpod:Electronic Gadget:2:750
018:Apple:Electronnic Gadget:9:3000
```

Fig **5.5**

When "1" is entered in the Admin Menu, the admin can enter new category data in Fig. 5.4, including the new item code, name, type, quantity, and price. When entered correctly, as seen in Fig. 5.5, it will be added to the item list.

Modify Item

```
 --- MODIFY ITEM ---
Please enter the ITEM CODE to modify: 017
Airpod Please enter new name: CPU
Electronic GadgetPlease enter new type(Electronic Gadget/Home Appliance: Home Appliance
2Please enter new quantity: 4
750Please enter new price: 1300
Modify Successfully
```

Fig **5.6**

```
 --- MODIFY ITEM ---
Please enter the ITEM CODE to modify: 1u2je
Record Not Found
```

Fig 5.7

```
 --- ITEM LIST ---


001:Iphone14:Electronic Gadget:7:5000
002:Iphone13:Electronic Gadget:2:4000
003:Samsung:Electronic Gadget:0:3400
004:Huawei:Electronic Gadget:5:2300
005:Robot Vacuum:Home Appliance:5:700
006:Air Purifier:Home Appliance:2:400
007:Smart Fan:Home Appliance:5:200
008:Smart Camera:Home Appliance:5:350
009:Smart Door:Home Appliance:2:930
010:Smart TV:Home Appliance:5:2200
011:Ipad:Electronic Gadget:9:1500
012:Stereo Speaker:Electronice Gadget:5:200
013:Smart Watch:Electronic Gadget:5:150
014:Monitor:Electronic Gadget:4:200
015:Printer:Electronic Gadget:5:250
016:Headphone:Electronic Gadget:3:50
017:CPU:Home Appliance:4:1300
018:Apple:Electronnic Gadget:9:3000
```

Fig **5.8**

After entering "2" in the Admin Menu, the admin must first enter the item code to establish whether or not the item already exists. If not, the notification in Fig. 5.7 will display "Record

Not Found" and return you to the Admin Menu. If it already exists, the administrator can start by modifying the name, type, quantity, and price. Figure 5.8 depicts the change in the item list as a result of the adjustment.

Delete Item

```
 --- REMOVE ITEM ---
Please enter item code to delete: 018
Record Not Found
```

Fig 5.9

```
 --- REMOVE ITEM ---
Please enter item code to delete: 016
Item Code: 016
Item Name: Headphone
Item Type: Electronic Gadget
Item Quantity: 3
Item Price: 50
Are you sure to delete this item? (y/n): y
018:Apple:Electronnic Gadget:9:3000
has been removed successfully
```

Fig **5.10**

```
 --- ITEM LIST ---


001:Iphone14:Electronic Gadget:7:5000
002:Iphone13:Electronic Gadget:2:4000
003:Samsung:Electronic Gadget:0:3400
004:Huawei:Electronic Gadget:5:2300
005:Robot Vacuum:Home Appliance:5:700
006:Air Purifier:Home Appliance:2:400
007:Smart Fan:Home Appliance:5:200
008:Smart Camera:Home Appliance:5:350
009:Smart Door:Home Appliance:2:930
010:Smart TV:Home Appliance:5:2200
011:Ipad:Electronic Gadget:9:1500
012:Stereo Speaker:Electronice Gadget:5:200
013:Smart Watch:Electronic Gadget:5:150
014:Monitor:Electronic Gadget:4:200
015:Printer:Electronic Gadget:5:250
017:CPU:Home Appliance:4:1300
018:Apple:Electronnic Gadget:9:3000
```

Fig **5.11**

Input "3" in the Admin Menu, delete item is similar to change item in that the admin must first identify whether the item currently exists or not. Otherwise, the notice in 5.9 will display

"Record Not Found" and return you to the Admin Menu. If it already exists, the administrator can explicitly specify the item in the item list shown in Figure 5.11.

View/Search customer order

```
1. View customer order history
2. Search customer order history
3. Back
Enter your choice:1

Order History:
Email/ Phone Number/ Order/ Delivery Status

jason@gmail.com/012-321-321/{'Item: 001': {'quantity': 1, 'subtotal(RM)': 5000.0}, 'Item: 017': {'quantity': 2, 'subtotal(RM)': 1500.0}}/
karen@gmail.com/012-123-123/{'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}, 'Item: 006': {'quantity': 1, 'subtotal(RM)': 400.0}}/
dennis@gmail.com/0123456789/{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}/Shipped

1. Back to admin menu
2. Exit to main menu
Enter your choice: 1
```

Fig **5.12**

```
1. View customer order history
2. Search customer order history
3. Back
Enter your choice:2

Order History:
Email/ Phone Number/ Order/ Delivery Status

jason@gmail.com/012-321-321/{'Item: 001': {'quantity': 1, 'subtotal(RM)': 5000.0}, 'Item: 017': {'quantity': 2, 'subtotal(RM)': 1500.0}}/
karen@gmail.com/012-123-123/{'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}, 'Item: 006': {'quantity': 1, 'subtotal(RM)': 400.0}}/
dennis@gmail.com/0123456789/{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}/Shipped

Enter email to search customer order: dennis@gmail.com
Email: dennis@gmail.com
Phone No: 0123456789
Order: {'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}
```

Fig **5.13**

```
Enter email to search customer order: eqewqeweq@gmail.com
Record not found!
```

Fig 5.14

When you type "4" into the Admin Menu, it offers three options: view customer order history, search customer order history, and back. When you enter view customer order history, it displays the previous orders that the client has placed (Fig 5.12). The only difference between search customer order history and view customer order history is that you may search for specify customer order history by inputting an email address (Fig 5.13). If the email address does not exist in the order history, the message "Record not found" will show in Fig 5.14. The back option is used to return to the Admin Menu.

View/Search customer payment

```
1. View customer payment history
2. Search customer payment history
3. Back
Enter your choice:1

Email, Phone No, Total Amount(RM), Payment Method, Card No, Payment Status

jason@gmail.com, 012-321-321, 6500.0, Debit, 128, paid
dennis@gmail.com, 0123456789, 18200.0, 12345, 12345, paid
```

Fig 5.15

```
1. View customer payment history
2. Search customer payment history
3. Back
Enter your choice:2

Email, Phone No, Total Amount(RM), Payment Method, Card No, Payment Status, Date and Time

jason@gmail.com, 012-321-321, 1500.0, Debit, 23910, paid, 2022-11-29 17:27:29.974520
idk@gmail.com, 013-321-432, 5400.0, Credit, 9031, paid, 2022-11-29 17:28:12.208111

Enter email to search customer payment: idk@gmail.com
Email: idk@gmail.com
Phone No:  013-321-432
Total Amount(RM):  5400.0
Payment Method:  Credit
Card No:  9031
Payment Status:  paid
Payment Record Date and Time:  2022-11-29 17:28:12.208111
```

Fig 5.16

```
Enter email to search customer payment: weqwewqewqe@gmail.com
Record not found!
```

Fig 5.17

The sole difference between View or Search Customer Payment and View or Search Customer Order is the ability to view payment history. View Consumer Payment History shows when a customer paid for an item or not (Fig 5.15). When you type search customer payment history, it displays the view payment history and allows you to search for a specific customer's payment history by email (Fig 5.16). If the email address is not discovered in the order history, the message "Record not found" will appear in Fig. 5.17, and the back button will send you to the admin menu.

Order Delivery Management

```
 --- ORDER DELIVERY MANAGEMENT SYSTEM ---

1. Add delivery staff
2. Modify delivery staff
3. Delete delivery staff
4. Assign orders to delivery staff
5. Exit
Enter your choice:
```

Fig 5.18

There are 5 types of function in order delivery management:

- 1 - Add delivery staff
- 2 - Modify delivery staff
- 3 - Delete delivery staff
- 4 - Assign orders to delivery staff
- 5 - Exit

Add delivery staff

```
 --- ORDER DELIVERY MANAGEMENT SYSTEM ---

1. Add delivery staff
2. Modify delivery staff
3. Delete delivery staff
4. Assign orders to delivery staff
5. Exit
Enter your choice: 1

 --- DELIVERY STAFF LIST ---

ID-PASSWORD-NAME

staff1:abc1:Jinyit:
staff2:abc2:Xinzhe:

 --- ADD DELIVERY STAFF ---
Please enter new delivery staff ID:staff3
Please enter new delivery staff PASSWORD:abc3
Please enter new delivery staff NAME:Dennis
staff3:abc3:Dennis has been successfully added
```

Fig 5.19

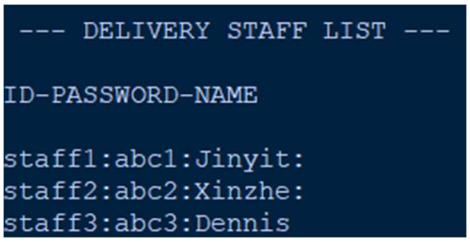


Fig 5.20

The first option in the order delivery management system, "Add Delivery Staff," was created so that our admin could add a new employee to the delivery staff. The administrator must provide the ID, password, and name as shown in Fig. 5.19 in order to create a new staff member. The system will show "(name) has been successfully inserted" after entry, indicating that the process is finished (Fig 20).

Modify delivery staff

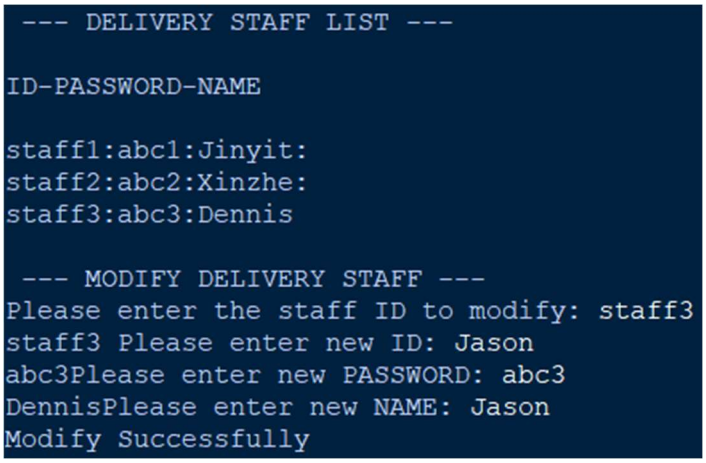


Fig 5.21

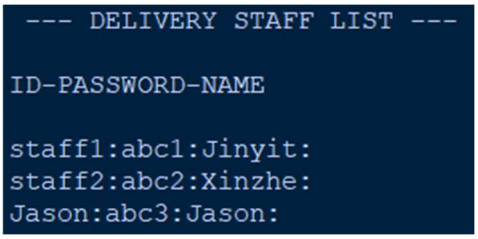


Fig 5.22

The second function "2" in order delivery management allows admin to update a specific staff member's ID, password, and name. After completing the input, the system will display "Modify Successfully", and the delivery staff list will be updated.

Delete delivery staff

```
 --- DELIVERY STAFF LIST ---

ID-PASSWORD-NAME

staff1:abc1:Jinyit:
staff2:abc2:Xinzhe:
Jason:abc3:Jason:

 --- REMOVE DELIVERY STAFF ---
Please enter staff ID to delete: Jason
Staff ID: Jason
Staff Password: abc3
Staff Name: Jason
Are you sure to delete this item? (y/n): y
Jason:abc3:Jason:
has been removed successfully
```

Fig 5.23

```
 --- REMOVE DELIVERY STAFF ---
Please enter staff ID to delete: staff3
Record Not Found

 --- ORDER DELIVERY MANAGEMENT SYSTEM ---

1. Add delivery staff
2. Modify delivery staff
3. Delete delivery staff
4. Assign orders to delivery staff
5. Exit
Enter your choice:
```

Fig 5.24

```
 --- DELIVERY STAFF LIST ---

ID-PASSWORD-NAME

staff1:abc1:Jinyit:
staff2:abc2:Xinzhe:
```

Fig 5.25 In order delivery management, the third function, Delete Delivery Staff, is utilised to remove a specific employee. To delete a staff member, an administrator must first enter the staff ID (Fig 5.23). If the admin input staff ID does not exist in the delivery staff list, the system

will display "Record Not Found" and redirect you to the order delivery management system, as shown in Fig. 5.24. If the employee ID already exists in the delivery staff list, the removal will be successful (Fig 5.25).

Assign orders to delivery staff

```
********** ASSIGN ORDER TO DELIVERY STAFF SYSTEM **********

 --- DELIVERY STAFF LIST ---
ID-PASSWORD-NAME-CUSTOMER EMAIL
staff1:abc1:Jinyit:
staff2:abc2:Xinzhe:

 --- SELECT STAFF ID TO ASSIGN ORDER FOR DELIVER ---
Enter staff ID to select: staff2

 --- CUSTOMER ORDER LIST ---
Order History:
Email/ Phone Number/ Order/ Delivery Status

jason@gmail.com/012-321-321/{'Item: 001': {'quantity': 1, 'subtotal(RM)': 5000.0}, 'Item: 017': {'quantity': 2, 'subtotal(RM)': 1500.0}}/
karen@gmail.com/012-123-123/{'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}, 'Item: 006': {'quantity': 1, 'subtotal(RM)': 400.0}}/
dennis@gmail.com/0123456789/{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}/Shipped

Enter customer email to assign deliver order: dennis@gmail.com
Updated Successfully
```

Fig 5.26

```
 --- SELECT STAFF ID TO ASSIGN ORDER FOR DELIVER ---
Enter staff ID to select: ewqweqweq@gmail.com
Record Not Found

 --- ORDER DELIVERY MANAGEMENT SYSTEM ---

1. Add delivery staff
2. Modify delivery staff
3. Delete delivery staff
4. Assign orders to delivery staff
5. Exit
Enter your choice: 4
```

Fig 5.27

```
********** ASSIGN ORDER TO DELIVERY STAFF SYSTEM **********

 --- DELIVERY STAFF LIST ---
ID-PASSWORD-NAME-CUSTOMER EMAIL
staff1:abc1:Jinyit:
staff2:abc2:Xinzhe:dennis@gmail.com
```

Fig 5.28

Choose employees to send certain email items using the fourth feature of the order delivery management system, assign orders to delivery personnel. The staff ID, which admin must enter to do it (Fig 5.26). When updating the delivery staff list, it will say "Updated Successfully" if

the staff ID is already in the delivery staff (Fig 5.28). It will say "Record Not Found" and take you back to the order delivery management system if the employee ID is not from the delivery personnel list (Fig 5.27).

REGISTER

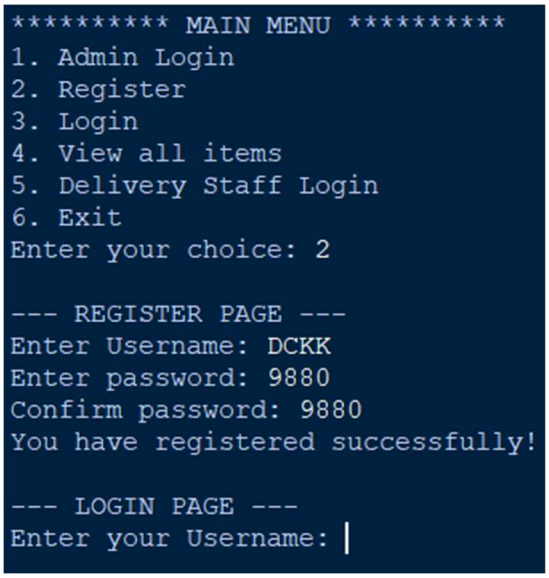


Fig 5.29

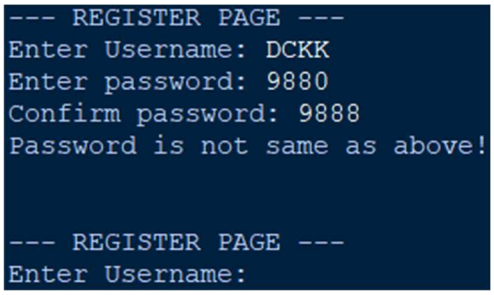


Fig 5.30

The system displays the register page when the user types "2" into the Main Menu. The user must enter a username, password, and password confirmation (Fig 5.29). If the password is different from the confirm password, it will display "Password is not the same as above!" and redirect to the registration page. For registration to be successful, the password and confirm password must match, and then the login page will be presented (Fig 5.30).

LOGIN PAGE

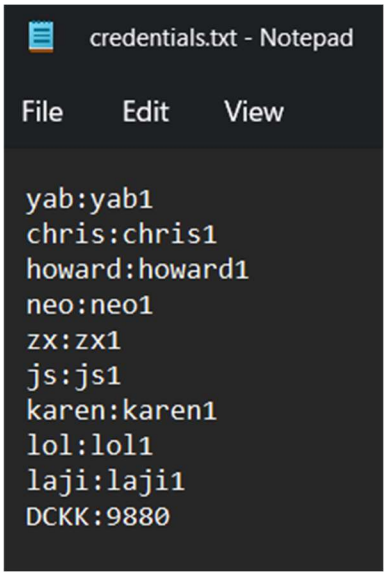


Fig 5.31

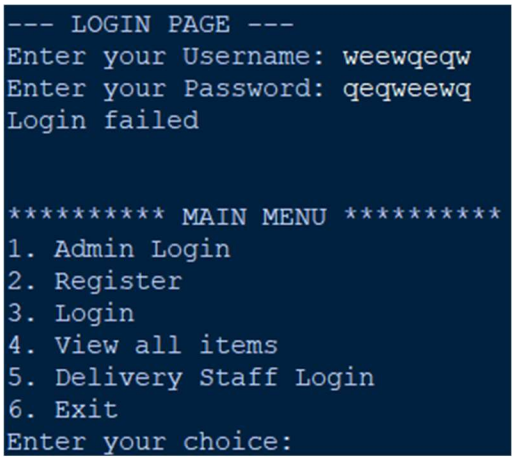


Fig 5.32

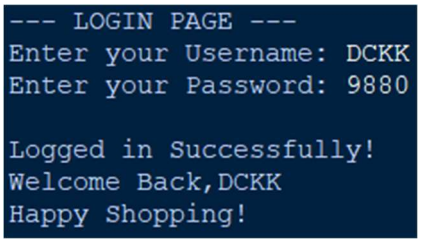


Fig 5.33

The function “3” login page in Main Menu, user must input username and password to order to login. If username and password is not same with credentials.txt (Fig 5.31), it will display “Login Failed” and back to Main Menu. If username and password is same with credentials.txt

(Fig 5.31), it will display “Logged in Successfully! Welcome Back, (username) Happy Shopping!”.

```
 --- Welcome to Jason and Dennis ALL IT APU Online Shopping Mall ---
1. View all item details
2. Place order
3. Exit
Enter your choice: 1
Item code     Item name              Type               Price(RM)
  001.        Iphone 14          Electronic Gadget          5000
  002.        Iphone 13          Electronic Gadget          4000
  003.        Samsung Galaxy     Electronic Gadget          3400
  004.        Huawei Nova        Electronic Gadget          2300
  005.        Robot Vacuum       Home Appliance              700
  006.        Air Purifier       Home Appliance              400
  007.        Smart Fan          Home Appliance              200
  008.        Smart Camera       Home Appliance              350
  009.        Smart Door         Home Appliance              930
  010.        Smart TV           Home Appliance             2600
  011.        Ipad               Electronic Gadget          1500
  012.        Stereo Speaker     Electronic Gadget           200
  013.        Smart Watch        Electronic Gadget           150
  014.        Monitor            Electronic Gadget           600
  015.        Printer            Electronic Gadget           250
  016.        Headphone          Electronic Gadget            50
  017.        Airpod             Electronic Gadget           750
Do you want to place an order? (yes/no): yes
Enter your email address:dennis@gmail.com
Enter your phone number:0123456789
```

Fig 5.34

```
 --- Welcome to Jason and Dennis ALL IT APU Online Shopping Mall ---
1. View all item details
2. Place order
3. Exit
Enter your choice: 2
Enter your email address:dennis@gmail.com
Enter your phone number:0123456789
```

Fig 5.35

It will say "Welcome to Jason and Dennis ALL IT APU Online Shopping Mall" after a successful login, and there are three types of functions: view all item details, place order, and exit. Place an order and see all item information are nearly identical; the only difference is that view all item details allows you to see every item that the online shopping mall owns (Fig 5.34), while place item order is limited to placing an item order (Fig 5.35). User's phone number and email address are required before placing an order.

```
********** Items Available in Our Store ***********

Item Code: Item Name: Item Type: Item Quantity: Item Price
001: Iphone14: Electronic Gadget: 7: 5000

002: Iphone13: Electronic Gadget: 6: 4000

003: Samsung: Electronic Gadget: 5: 3400

004: Huawei: Electronic Gadget: 5: 2300

005: Robot Vacuum: Home Appliance: 5: 700

006: Air Purifier: Home Appliance: 2: 400

007: Smart Fan: Home Appliance: 5: 200

008: Smart Camera: Home Appliance: 5: 350

009: Smart Door: Home Appliance: 2: 930

010: Smart TV: Home Appliance: 5: 2200

011: Ipad: Electronic Gadget: 9: 1500

012: Stereo Speaker: Electronice Gadget: 5: 200

013: Smart Watch: Electronic Gadget: 5: 150

014: Monitor: Electronic Gadget: 4: 200

015: Printer: Electronic Gadget: 5: 250

016: Headphone: Electronic Gadget: 3: 50

017: Airpod: Electronic Gadget: 2: 750

Do you wish to proceed (y/n):
```

Fig 5.36

While user done input email and phone number, it will show all the item code, item name, item type, item quantity, and item price by looking at Fig 5.36.

```
Do you wish to proceed (y/n): y
Enter item code to place order: samsung
Item not found. Please try again.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~GOING BACK ORDER PAGE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter your email address:
```

Fig 5.37

```
Do you wish to proceed (y/n): y
Enter item code to place order: 003
003:Samsung:Electronic Gadget:5:3400 > Enter quantity: 10
{'Item: 003': {'quantity': 10, 'subtotal(RM)': 34000.0}}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~FAIL TO ADD CART~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Item stock quantity not enough!!
Please re-enter the quantity of product needed according to our stock quantity.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~PLEASE PLACE ORDER AGAIN~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter your email address:
```

Fig 5.38

```
Do you wish to proceed (y/n): y
Enter item code to place order: 003
003:Samsung:Electronic Gadget:5:3400 > Enter quantity: 3
{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ADD CART SUCCESSFUL~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Do you wish to add more items(y/n):
```

Fig 5.39

```
Do you wish to proceed (y/n): y
Enter item code to place order: 003
003:Samsung:Electronic Gadget:5:3400 > Enter quantity: 3
{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ADD CART SUCCESSFUL~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Do you wish to add more items(y/n): y
Enter item code to place order: 002
002:Iphone13:Electronic Gadget:6:4000 > Enter quantity: 2
{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ADD CART SUCCESSFUL~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Do you wish to add more items(y/n):
```

Fig 5.40

To place an order, the user must enter the item code. If the item code does not match an item in our shop, the message "Item not found" will appear. Please try again" and then return to enter email and password until the user enters the same item code (Fig 5.37). After the user has entered the item code, the quantity must be entered. If the item amount is exceeded, the message "Item stock quantity insufficient!!" will appear. Please re-enter the number of products required based on our stock quantity" and then return to enter email and password (Fig 5.38). After the user enters the item code and quantity, the cart is successfully added, and the user is prompted to add further goods (Fig 5.39). If affirmative, the user can continue to enter item codes and quantities until all goods have been added (Fig 5.40).

```
**********Bill Summary**********
***********JASON AND DENNIS ALL IT APU ONLINE SHOPPING MALL***********
Item                    Quantity                    Subtotal
Item: 003               3                   10200.0
Total: 10200.0
Item: 002               2                   8000.0
Total: 18200.0
Do u wish to proceed payment (y/n):y
Enter your payment method (Debit/Credit) card:12345
Enter your 12345 card number:12345
Comfirm payment? (yes/no) :yes

*********** Thank You ***********
Hope to see you back soon!
```

Fig 5.41

```
**********Bill Summary**********
***********JASON AND DENNIS ALL IT APU ONLINE SHOPPING MALL***********
Item                    Quantity                    Subtotal
Item: 003               3                   10200.0
Total: 10200.0
Item: 002               2                   8000.0
Total: 18200.0
Do u wish to proceed payment (y/n):n
Back to main menu

********** MAIN MENU **********
1. Admin Login
2. Register
3. Login
4. View all items
5. Delivery Staff Login
6. Exit
Enter your choice:
```

Fig 5.42

After the user has finished inputting items and quantities, the Bill Summary will appear. It displays the item, quantity, and subtotal, followed by the message "Do you like to proceed payment (y/n)?" If you select no, you will be returned to the Main Menu (Fig 5.42). If you select yes, you must provide your payment method (debit/credit card) and card number (Fig 5.41). Following that, the system will prompt you to confirm payment of the tax. If no, it will go back to the Main Menu. If yes, the Admin Menu will be updated (view/search customer order and payment) and the message "Thank you, hope to see you back soon" will be displayed.

VIEW ALL ITEM

```
 --- VIEW ALL ITEM PAGE ---

No    Category
1.    Electronic Gadget
2.    Home Appliance


001:Iphone14:Electronic Gadget:7:5000
002:Iphone13:Electronic Gadget:2:4000
003:Samsung:Electronic Gadget:2:3400
004:Huawei:Electronic Gadget:5:2300
005:Robot Vacuum:Home Appliance:5:700
006:Air Purifier:Home Appliance:2:400
007:Smart Fan:Home Appliance:5:200
008:Smart Camera:Home Appliance:5:350
009:Smart Door:Home Appliance:2:930
010:Smart TV:Home Appliance:5:2200
011:Ipad:Electronic Gadget:9:1500
012:Stereo Speaker:Electronice Gadget:5:200
013:Smart Watch:Electronic Gadget:5:150
014:Monitor:Electronic Gadget:4:200
015:Printer:Electronic Gadget:5:250
016:Headphone:Electronic Gadget:3:50
017:Airpod:Electronic Gadget:2:750


1.Registered
2.Unregistered
3.Back

Please enter your choice:
```

Fig 5.43

The function “4” in Main Menu, looking at Fig 5.43 there are every item that the online shopping mall owns, and 3 type of function is registered, unregistered, and back.

```
1.Registered
2.Unregistered
3.Back

Please enter your choice:2

1. Register to access more details
2. Exit
Please enter your choice:1

--- REGISTER PAGE ---
Enter Username:
```

Fig 5.44

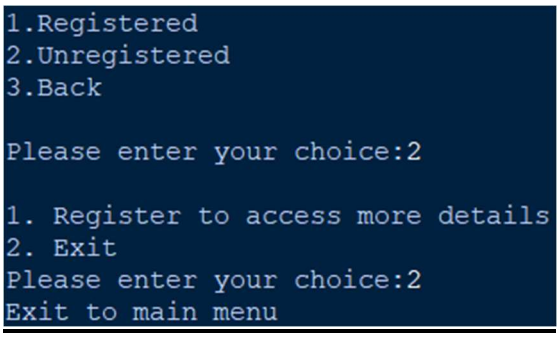


Fig 5.45

If the user selects unregistered, it provides two options: register to access more details and exit. If the user chooses to register to have access to more details, they will be directed to the registration page (Fig 5.44). If the user selects exit, the application will return to the Main Menu (Fig 5.45).

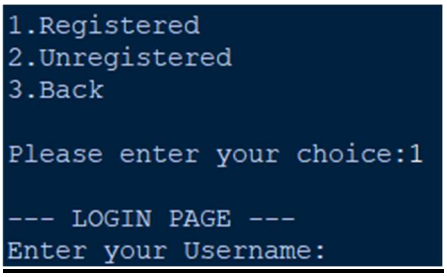


Fig 5.46

If user choose registered, it will go back to Login Page (Fig 5.46).

DELIVERY STAFF LOGIN

```
********** MAIN MENU **********
1. Admin Login
2. Register
3. Login
4. View all items
5. Delivery Staff Login
6. Exit
Enter your choice: 5

 --- LOGIN PAGE FOR DELIVERY STAFF ---
Enter your username:staff1
Enter your password:abc1
Logged in Successfully!
Welcome back staff1
```

Fig 5.47

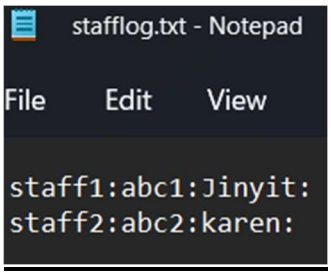


Fig 5.48

```
 --- LOGIN PAGE FOR DELIVERY STAFF ---
Enter your username:qweqweq
Enter your password:qewqeqw
Login failed, credential not valid

********** MAIN MENU **********
1. Admin Login
2. Register
3. Login
4. View all items
5. Delivery Staff Login
6. Exit
Enter your choice:
```

Fig 5.49

The function "5" in the Main Menu requires the user to enter a username and password in order to login delivery staff. If the username and password are not the same as those in stafflog.txt (Fig 5.48), the message "Login failed, credential not valid" will appear, and the user will be returned to the Main Menu. If the username and password are the same, the login will be successful and the message "Welcome back staff" will be displayed.

```
 --- DELIVERY STAFF SYSTEM ---

Order History:
Email/ Phone Number/ Order/ Delivery Status

jason@gmail.com/012-321-321/{'Item: 001': {'quantity': 1, 'subtotal(RM)': 5000.0}, 'Item: 017': {'quantity': 2, 'subtotal(RM)': 1500.0}}/
karen@gmail.com/012-123-123/{'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}, 'Item: 006': {'quantity': 1, 'subtotal(RM)': 400.0}}/
dennis@gmail.com/0123456789/{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}/

 --- SELECT CUSTOMER ORDER TO UPDATE DELIVERY STATUS ---
Enter customer email to select: dckk@gmail.com
Record Not Found
```

Fig 5.50

```
 --- DELIVERY STAFF SYSTEM ---

Order History:
Email/ Phone Number/ Order/ Delivery Status

jason@gmail.com/012-321-321/{'Item: 001': {'quantity': 1, 'subtotal(RM)': 5000.0}, 'Item: 017': {'quantity': 2, 'subtotal(RM)': 1500.0}}/
karen@gmail.com/012-123-123/{'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}, 'Item: 006': {'quantity': 1, 'subtotal(RM)': 400.0}}/
dennis@gmail.com/0123456789/{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}/

 --- SELECT CUSTOMER ORDER TO UPDATE DELIVERY STATUS ---
Enter customer email to select: dennis@gmail.com
Do you sure you want to update the delivery status? (yes/no): yes
Enter number to update the following status (1.Shipped, 2.Delivered 3.Cancelled): 1
Updated Successfully
```

Fig 5.51

```
 --- DELIVERY STAFF SYSTEM ---

Order History:
Email/ Phone Number/ Order/ Delivery Status

jason@gmail.com/012-321-321/{'Item: 001': {'quantity': 1, 'subtotal(RM)': 5000.0}, 'Item: 017': {'quantity': 2, 'subtotal(RM)': 1500.0}}/
karen@gmail.com/012-123-123/{'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}, 'Item: 006': {'quantity': 1, 'subtotal(RM)': 400.0}}/
dennis@gmail.com/0123456789/{'Item: 003': {'quantity': 3, 'subtotal(RM)': 10200.0}, 'Item: 002': {'quantity': 2, 'subtotal(RM)': 8000.0}}/Shipped

 --- SELECT CUSTOMER ORDER TO UPDATE DELIVERY STATUS ---
Enter customer email to select:
```

Fig 5.52

Fig 5.50 shows the order history of email, phone number, order, and delivery status once a staff member has logged in. To order a specific customer and change the delivery status, a staff member must provide the customer's email address. If the email address does not match the one in the order history, it will display "Record Not Found" and return to the customer email field. If the email address is the same as the one in the order history, it will continue to confirm that you want to update the delivery status. If affirmative, it will allow the staff member to select one of three statuses: shipped, delivered, or cancelled. After selecting a status, the order history will be successfully updated.

6.0 Conclusion

To summarize, this module has proven to be very helpful, and the benefits extend beyond only gaining knowledge of how to program using the Python language. Variables, data types, data structures, control structures, and file handling are some of the things that we have learned. We will be able to quickly adjust to whichever programming languages we end up utilizing in the future as long as we have a solid knowledge of these fundamental notions. We have learned to build the system by utilizing pseudocode and flowchart, which has assisted us in planning out our system as we are writing the code. This has allowed us to explore the Python built-in functions less extensively than we have in the past. In addition to that, we were able to obtain a deeper comprehension of the structure of the system according to it. On the basis of this, if we are confronted with a bottleneck, we can easily address it by looking at the pseudocode or flowchart and doing so. Pseudocode and flowchart were also brought to our attention, and we recognized their significance due to the fact that they prevent us from encountering challenges while dealing with others who are unfamiliar with the programming language that we are using.

References

Datetime in Python: Everything You Need to Know | Simplilearn. (2021, March 22). Simplilearn.com. Retrieved December 1, 2022, from https://www.simplilearn.com/tutorials/python-tutorial/datetime-in-python

evangelista, A. (2021, March 11). Online Shopping System Project in Python with Source Code. Itsourcecode.com. Retrieved December 1, 2022, from https://itsourcecode.com/free-projects/python-projects/online-shopping-project-in-python-with-source-code/

Python for Loop (With Examples). (n.d.). Python for Loop (With Examples). Retrieved December 1, 2022, from https://www.programiz.com/python-programming/for-loop

Python Variables: How to Define/Declare String Variable Types. (2019, May 25). Guru99. Retrieved December 1, 2022, from https://www.guru99.com/variables-in-python.html

Python - Algorithm Design. (n.d.). Python - Algorithm Design. Retrieved December 1, 2022, from https://www.tutorialspoint.com/python_data_structure/python_algorithm_design.htm