A · P · U

ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

CT108-3-1-PYP-L-3

Python Programming

Intake Code: APU1F2307SE

Lecturer Name: AMARDEEP SINGH A/L UTTAM SINGH

Hand in Date: Week 13 (28th Oct 2023)

Group Members:

1. **Ho Yan Xun** **TP073669**
2. **Yap Zhu Sheng** **TP073670**

Introduction

This is a documentation for the Inventory Management System for Personal Protection Equipment (PPE). It documents the architecture, functionality, limitations, and proper usage of the software. The software has the capabilities of receiving, distributing equipment to hospitals as well as stock management and transaction recording.

The Inventory Management System is built using python and is a menu driven and can be run within a terminal. This means that it does not have a GUI (Graphical User Interface). This kind of approach makes the application more lightweight and can be automated with scripts. The system is also designed with input validation to catch any sort of errors and prevent the system from crashing.

Since the images of flowchart and pseudocode provided in this documentation can be blurry due to compression, a GitHub repository is available at the link below, a PDF file containing the flowcharts and the pseudocode of this program can be found there.

GitHub Repository Link:

https://github.com/AaronHo-0716/InventoryManagementSystem

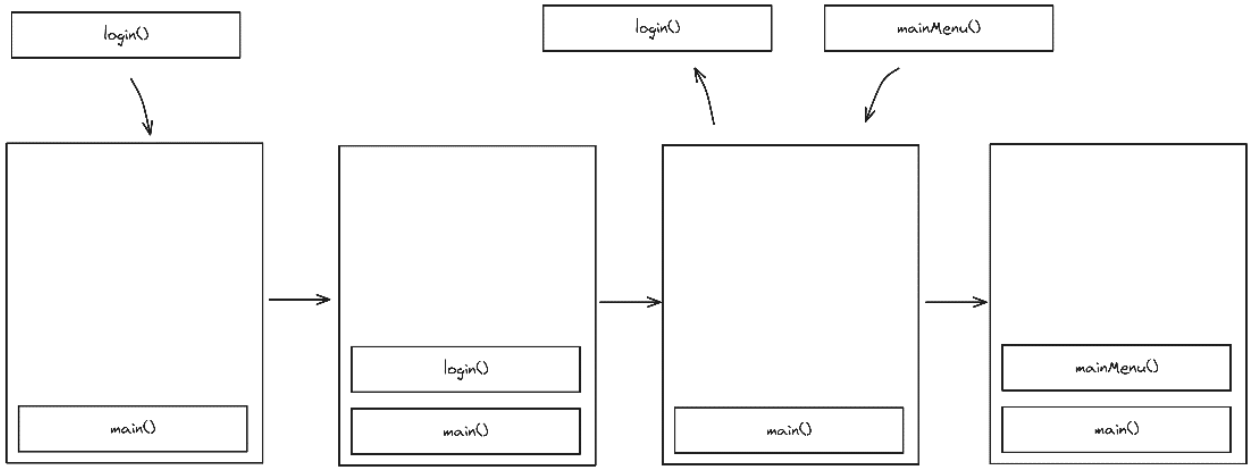
Assumptions / Limitations

1. The user can only enter 6 PPE items, each with different item code, but same names are manageable.
    a. HC Head Cover
    b. FS Face Shield
    c. MS Mask
    d. GL Gloves
    e. GW Gown
    f. SC Shoe Covers
2. The user can only enter the details for 3-4 hospitals and suppliers. Once the numbers of suppliers and hospitals are entered during the initialization phase of the program, it cannot be changed. Only the name and contact numbers of the suppliers are editable once initialized. The hospital code and supplier code cannot be changed once entered.
3. Each user will be required have their individual account, either as an admin or a staff to access the program. Only user accounts that are "Admin" can manage users.
4. The first account created during the initialization phase of the program acts as a master account that cannot be deleted or otherwise lose its privileges. This is to ensure there is always an account with the privilege to manage other users.
5. The program creates different text files that contains different information in each stage of the program. If the text files are corrupted or modified manually, malfunction of the program might appear.
6. Any codename cannot be duplicated within the same type, for example, two PPE items cannot share the same item code, or else it might cause confusion because items, suppliers and hospitals are identified based on their codenames.
7. An infinite amount of user can be created, but a user that is already logged in into the program needs to log out first before a second user can login and use the program. If the program is duplicated within the same directory to allow more user to run the program at the same time, conflicts may occur while writing changes to the text files.

Design of the program

The program uses a functional programming approach. Some of the functions act as individual pages of the system, similar to a computer file system, each function menu is like a different directory that handles different functions (eg. mainMenu() > inventory() > transactionHistory())

The structure also implements the “stack” data structure in which each function stacks on top of each other and each function pops off the function stack according to a “last in first out” principle. To prevent the function call stack in python to grow indefinitely, functions are wrap inside a sentinel loop that breaks under certain conditions that also pops the function from the function call stack, instead of one function calling another.



Example of the function call stack for login and entering the main menu

Consequently, if another user wishes to use the program while there is already a user logged in. The user will be required to log out, which remove all the opened function out of the “stack” to return to the loginMenu, until then only one function is left at the bottom of the “stack”. After the user logs in, the functions that are called will “stack” up again.

The program consists a few lines of code that acts as the entry point of the program. From there, different functions for different purposes will be called and exit simultaneously.

Program’s error and bugs are kept to a minimum. Errors and exceptions such as invalid input and duplicated entry are handled by printing error messages and returned until valid input is entered.

Comments are also added to the code to help explain how the program works.

The program only uses one external library, datetime, it is to supply classes and methods for manipulating date and time. Date and time are recorded during transaction of PPE items in the program.

Functions in the program

1. main

This is the entry point for the program when the program is running. A short portion of the script will check if the Python script is being run as a program or imported as a module. If it is being run as a program, the main function will be called.

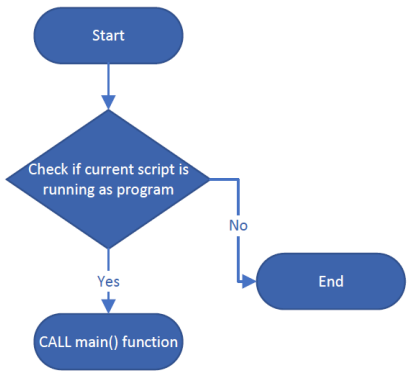
Pseudocode

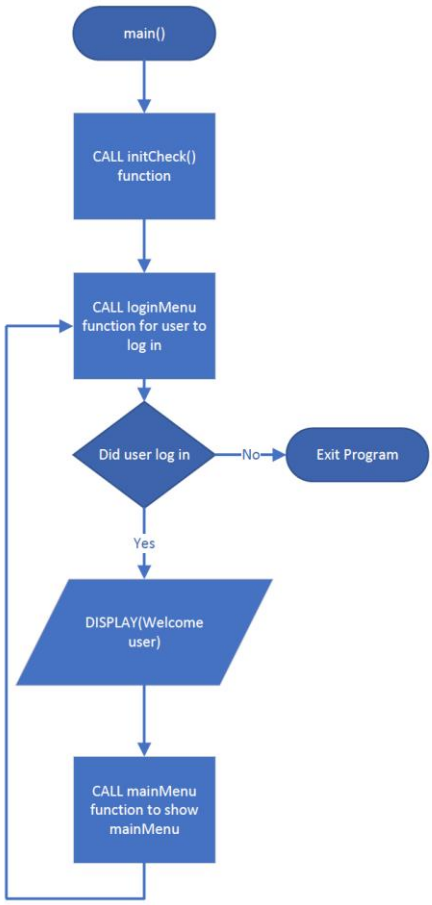
```
main function
    CALL initCheck function
    ENTER LOOP
        CALL loginMenu function for user log in
        READ user info from loginMenu function
        Check if user logged in
            IF TRUE
                DISPLAY(Welcome user)
                CALL mainMenu function to show mainMenu to user as admin/staff
            IF False
                EXIT PROGRAM

CHECK IF current script is running as program
IF True
    CALL main function
ELSE
    do nothing
```

Python Code

```
#entry point of the program
def main():
    initCheck()
    while True:
        # loginStatus, userID, userName, userType
        loginInfo = loginMenu()

        print(" _____ __        __  _____   ")
        print("|      \\  \\      /  \\/      \\ ")
        print(" \\\\▓▓▓▓▓ ▓▓\\    /  ▓▓ ▓▓▓▓▓\\\\")
        print("  | ▓▓ | ▓▓\\\\ /  ▓▓ ▓▓__\\\\▓▓")
        print("  | ▓▓ | ▓▓▓\\\\  ▓▓▓\\\\▓▓    \\\\ ")
        print("  | ▓▓ | ▓▓\\▓▓ ▓▓ ▓▓_\\\\▓▓▓▓▓\\\\")
        print(" _| ▓▓_| ▓▓ \\\\▓▓| ▓▓  \\\\__| ▓▓")
        print("|   ▓▓ \\\\ ▓▓  \\\\▓ | ▓▓\\\\▓▓    ▓▓")
        print(" \\\\▓▓▓▓▓\\\\▓▓      \\\\▓▓ \\\\▓▓▓▓▓ ")

        print(f"\nWelcome {loginInfo[2]}")

        mainMenu(loginInfo)

#checks whether current script is run as program or imported as a module
#if running as program, main function will be called
if __name__ == "__main__":
    main()
```

FlowCharts





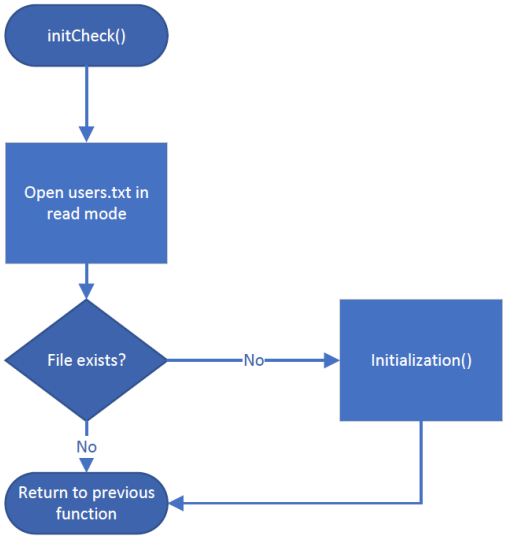
2. initCheck

This function performs checking to see if it is the first time running the program by detecting the existence of the text file containing user information. If the text file doesn't exist, the program will proceed to call the Initialization function for initialization process.

Pseudocode

```
initCheck function
    TRY
        open users.txt file in read mode
    IF file doesn't exist
        DISPLAY(Initializing system ... )
        CALL initilization function
```

FlowChart



Python Code

```
def initCheck():
    try:
        open('users.txt', "r")
    except FileNotFoundError:
        print("Initializing system ... ")
        initialization()
```

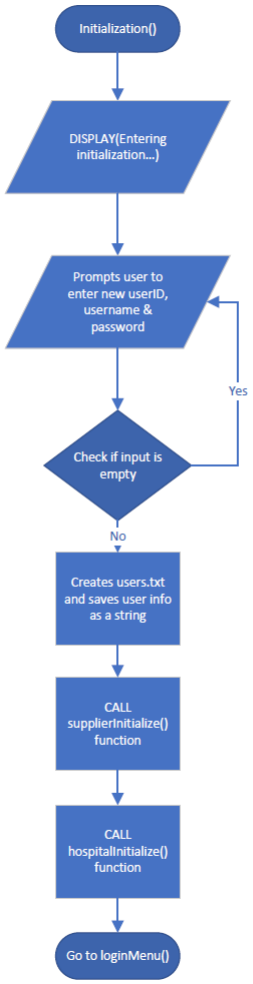
3. Initialization

This function prompts the user to create an account before proceeding. The user will be required to enter a username, a userID and a password to create an admin account. The function will then create a users.txt file and saves the user info as a string. This function will then call the supplierInitialize and hospitalInitialize function.

Pseudocode

```
initilization function
    DISPLAY(Entering initilization)
    ENTER LOOP
        PROMPTS user to enter new userID
        PROMPTS user to enter new userName
        PROMPTS user to enter password

        IF userID OR userName OR password is empty
            DISPLAY(Please fill in all the details.)
            RESTART LOOP

        EXIT LOOP

    convert user input into a string as(userID,userName,"Admin",password)

    OPEN users.txt IN WRITE MODE
        WRITE string into users.txt

    CALL supplierInitialize function
    CALL hospitalInitialize function
    DISPLAY(Initialization complete.)
```

Flowchart



Python Code

```
# Creates users.txt and propmts user to create a admin account
def initialization():
    print("\nEntering initialization, please enter the userID and password for creating an admin
account")
    while True:
        userID = input("Please enter your userID: ")
        userName = input("Please enter your name: ")
        password = input("Please enter your password: ")

        if userID == "" or userName == "" or password == "": #input cannot be empty
            print("Please fill in all the details\n")
            continue

        break

    users = ",".join([userID, userName, "Admin", password])

    with open("users.txt", "w") as f:    #saves user info inside users.txt
        f.write(users)
    supplierInitialize() #called to initialize suppliers.txt by adding suppliers' info
    hospitalInitialize() #called to initialize hospitals.txt by adding hospitals' info
    print("Initialization complete.\n")
```

4. supplierInitialize

This function checks if the supplier's info has been initialized or not by detecting the existence of suppliers.txt. If not, the user will be prompted to enter suppliers' code, suppliers' name and supplier's contact number. Only a minimum of 3 and a maximum of 4 suppliers can be entered.

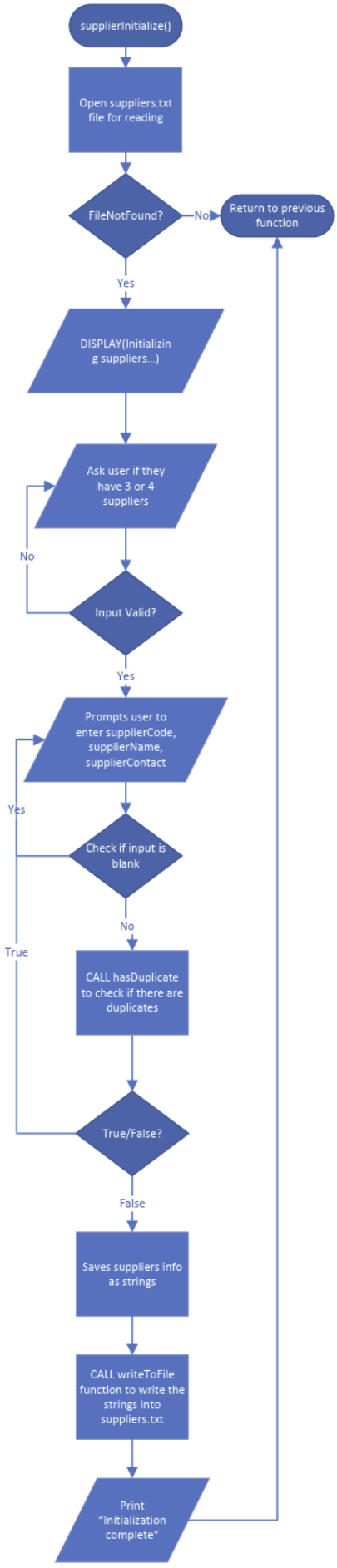
Pseudocode

```
supplierInitialize function
    OPEN suppliers.txt file in read mode
    IF suppliers.txt file not found
        DISPLAY(Initializing suppliers.txt)
        ENTER LOOP
            ASK user if they have 3 or 4 suppliers
                IF 3
                    supplierAmount = 3
                    EXIT LOOP
                IF 4
                    supplierAmount = 4
                    EXIT LOOP
                IF others
                    DISPLAY(Please enter 3 or 4 only)
                    RESTART LOOP
        ENTER LOOP
            PROMPTS user to enter 3 or 4 supplier codes
                CALL hasDuplicate function to check if suppliers code is duplicated
                IF True
                    RESTART LOOP
                IF False
                    CONTINUE
            PROMPTS user to enter 3 or 4 supplier names
                CALL hasDuplicate function to check if suppliers names is duplicated
                IF True
                    RESTART LOOP
                IF False
                    CONTINUE
            PROMPTS user to enter 3 or 4 supplier contact number
                CALL hasDuplicate function to check if suppliers contact number is
duplicated           IF True
                        RESTART LOOP
                    IF False
                        CONTINUE

            IF supplierCode or supplierName or supplierContact is empty
                RESTART LOOP
            ELSE
                EXIT LOOP

        ARRANGE supplier info into a string

        CALL writeToFile function to save supplier info into suppliers.txt
        DISPLAY(Initialization complete)
    IF ERROR exist
        PRINT OUT ERROR

    EXIT FUNCTION
```

Python Code

```
#initilization for suppliers, and to check if suppliers.txt exists
def supplierInitialize():
    try:
        open("suppliers.txt", "r")
        #checks if suppliers.txt exists or not, if yes the program will be returned to the previous
function, if no user will be required to enter suppliers' info
    except FileNotFoundError:
        print("\nInitializing suppliers.txt ... \n")
        while True:
            suppliers = []
            try:
                while True:
                    supplierAmount = input("Do you have 3 or 4 suppliers: ") #as in the assignment
question, only 3-4 are allowed
                    match supplierAmount:
                        case "3":
                            break
                        case "4":
                            break
                        case _:
                            print("\nPlease enter 3 or 4 only.\n")
                            continue
                while True:
                    print("\nPlease enter the details of "+str(supplierAmount)+ " suppliers only.")
                    print("Example: AA,BB,CC\n")

                    #suppliers' code, name and contact number cannot be duplicated
                    while True:
                        supplierCode = list(input("Please enter the all the supplier code with comma in
between: ").strip().split(','))
                        #item supplier can be "JJ,Ab,Pf,GSK,JJ,Ab"
                        supplierError = hasDuplicates(supplierCode)
                        if not supplierError:
                            break

                    while True:
                        supplierName = list(input("Please enter the all the supplier name with comma in
between: ").strip().split(','))
                        #supplierName can be Johnson & Johnson,Abott,Pfizer,GlaxoSmithKline
                        supplierError = hasDuplicates(supplierName)
                        if not supplierError:
                            break

                    while True:
                        supplierContact = list(input("Please enter the all the supplier contact number
with comma in between: ").strip().split(','))
                        #supplierContact can be 123,456,789,101112
                        supplierError = hasDuplicates(supplierContact)
                        if not supplierError:
                            break

                    if supplierCode =="" or supplierName == "" or supplierContact == "": #cannot be
left empty
                        continue
                    else:
                        break
                #arrange supplierCode, supplierName, supplierContact into a string for each supplier
                for i in range(0,int(supplierAmount)):
                    suppliers.append([supplierCode[i], supplierName[i], supplierContact[i]])

                writeToFile("suppliers.txt", sorted(suppliers))
                print("Initializing complete")
                break

            except IndexError:
                print("\nError in input, please try again.\n")

            except Exception as e:
                print(e)

    return
```

Flowchart


supplierInitialize()
Open suppliers.txt file for reading
FileNotFound?
No
Return to previous function
Yes
DISPLAY(Initializing suppliers...)
Ask user if they have 3 or 4 suppliers
No
Input Valid?
Yes
Prompts user to enter supplierCode, supplierName, supplierContact
Yes
Check if input is blank
No
True
CALL hasDuplicate to check if there are duplicates
True/False?
False
Saves suppliers info as strings
CALL writeToFile function to write the strings into suppliers.txt
Print "Initialization complete"

5. hospitalInitialize

Similar to the supplierInitialize function, this function checks if the hospitals' info exists. If not, the user will be required to enter 3-4 hospitals' code and name. The details will be saved as strings in hospitals.txt.

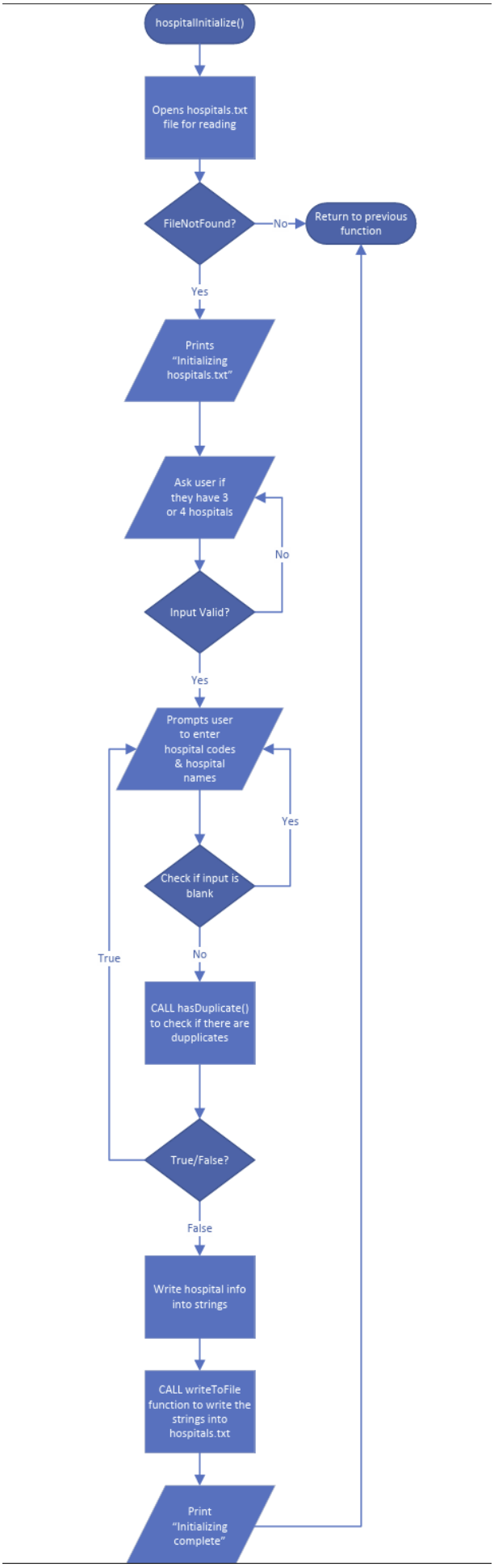
Pseudocode

```
hospitalInitialize function
    OPEN hospitals.txt in read mode
    IF file doesn't exist
        DISPLAY(Initializing system... )
        ENTER LOOP
            ASK user if they have 3 or 4 hospitals
                IF 3
                    hospitalAmount = 3
                    EXIT LOOP
                IF 4
                    hospitalAmount = 4
                    EXIT LOOP
                IF OTHERS
                    DISPLAY(Please enter 3 or 4 hospitals only.)
                    RESTART LOOP
        ENTER LOOP
            PROMPTS user to enter 3 or 4 hospitals code
            CALL hasDuplicate function to check if hospitals code is duplicated
                IF True
                    RESTART LOOP
                IF False
                    CONTINUE
            PROMPTS user to enter 3 or 4 hospital names
                CALL hasDuplicate function to check if hospitals code is duplicated
                IF True
                    RESTART LOOP
                IF False
                    CONTINUE

            ARRANGE hospital info into a string

            CALL writeToFile function to save hospitals info to hospitals.txt
            DISPLAY(Initialization complete)
            EXIT LOOP
    IF error exists
        PRINT OUT ERROR
    EXIT FUNCTION
```

FlowChart

```
#initialization for hospitals
def  hospitalInitialize():
    try:
        open("hospitals.txt","r") #check if hospitals.txt has been initialized or not
    except FileNotFoundError:
        print("\nInitializing system... \n")
        while True:
            hospitals = []
            while True:
                hospitalAmount = input("Do you have 3 or 4 hospitals: ")
                match hospitalAmount:
                    case "3":
                        break
                    case "4":
                        break
                    case _:
                        print("\nPlease enter 3 or 4 hospitals only.\n")
                        continue
            try:
                print("\nPlease enter the details of "+str(hospitalAmount)+" hospitals only.")
                print("Example: AA,BB,CC\n")

                while True:
                    hospitalCode = list(input("Please enter the all the hospital code with comma in
between: ").strip().split(','))
                    #hospitalCode can be "KKM,KPJ,KBJ,HJ"
                    hospitalError = hasDuplicates(hospitalCode)
                    if not hospitalError:
                        break

                while True:
                    hospitalName = list(input("Please enter the all the hospital name with comma in
between: ").strip().split(','))
                    #hospitalName can be "Klinik Kesihatan Malaysia,Klinik Petaling Jaya,Klinik Bukit
Jalil,Hospital Johor"
                    hospitalError = hasDuplicates(hospitalName)
                    if not hospitalError:
                        break

                for i in range(0,int(hospitalAmount)): #arrange hospitalCode,hospitalName as a string
for each hospital
                    hospitals.append([hospitalCode[i], hospitalName[i]])

                writeToFile("hospitals.txt", sorted(hospitals))
                print("Initializing complete")
                break

            except IndexError:
                print("\nError in input, please try again.\n")

            except Exception as e:
                print(e)
    return
```

Flowchart


hospitalInitialize()
Opens hospitals.txt file for reading
FileNotFound?
No
Return to previous function
Yes
Prints "Initializing hospitals.txt"
Ask user if they have 3 or 4 hospitals
No
Input Valid?
Yes
Prompts user to enter hospital codes & hospital names
Yes
Check if input is blank
True
No
CALL hasDuplicate() to check if there are dupplicates
True/False?
False
Write hospital info into strings
CALL writeToFile function to write the strings into hospitals.txt
Print "Initializing complete"

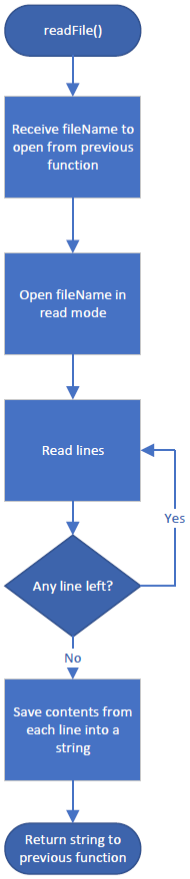
6. readFile

This function opens a specific text file that is given in read mode and parse the string to workable lists.

Pseudocode

```
readFile function with parameter(filePath)
    OPEN(filePath) in read mode
        ENTER LOOP
            READ each line
            IF no line left
                EXIT LOOP
            ELSE
                ADD each line into a content string
    RETURN content string
```

Flowchart



Python Code

```
# function for reading contents of a specific file returning it as a string
def readFile(filePath):
    content = []
    with open(filePath, 'r') as f:
        while True:
            line = f.readline().rstrip('\n') #read each line and add them to the string
            if not line:
                break
            else:
                content.append(list(line.split(',')))

    return content
```

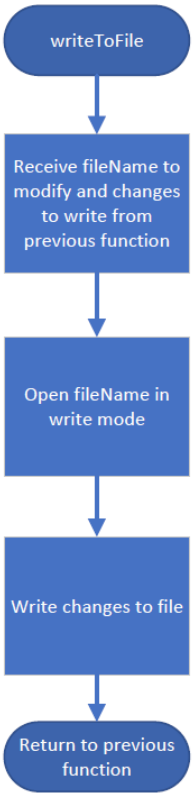
7. writeToFile

This function is for writing changes to a specific file.

Pseudocode

```
writeToFile function with parameter(fileName, changes)
    OPEN(fileName) in write mode
        ENTER LOOP
            ADD changes to file
```

Flowchart



Python Code

```
#function for writing a content to a specific file
def writeToFile(fileName, original):
    with open(fileName, 'w') as f:
        for i in original:
            f.write(','.join(i)) #adds the content to end of the file
            f.write('\n')
```

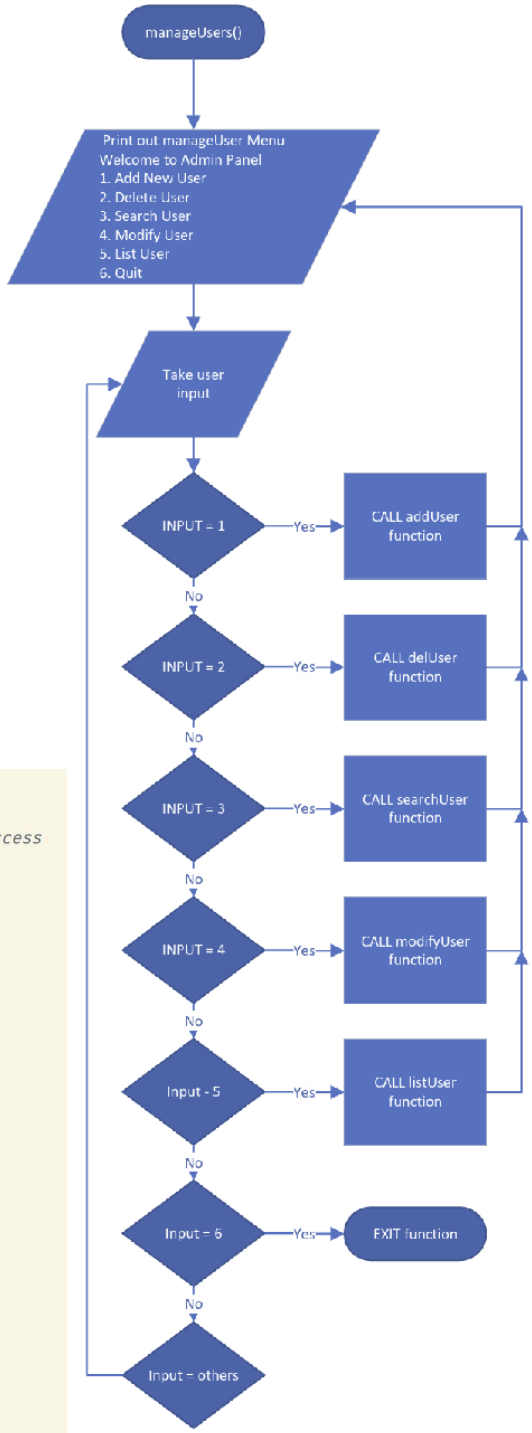
8. manageUsers

This function prints out the menu for managing users for admins only. This function will then call the other functions responsible for managing the users which are adding new user, deleting a user, searching for a user and modifying a user.

Pseudocode

```
manageUsers function
    ENTER LOOP
        DISPLAY(Welcome to Admin Panel)
        DISPLAY(1. Add New User)
        DISPLAY(2. Delete User)
        DISPLAY(3. Search User)
        DISPLAY(4. Modify User)
        DISPLAY(5. List User)
        DISPLAY(6. Quit)

        TAKE user input
            INPUT = 1
                CALL addUser function
            INPUT = 2
                CALL delUser function with PARAMETER(loginInfo)
            INPUT = 3
                CALL searchUser function
            INPUT = 4
                CALL modifyUser with PARAMETER(loginInfo)
            INPUT = 5
                CALL listUsers function
            INPUT = 6
                EXIT
            INPUT = OTHERS
                DISPLAY(Choice entered not valid, pls try again)
```

Flowchart



Python Code

```python
#function for displaying menu for managing users for admin, only admin can access
def manageUsers(loginInfo):
    flag = True
    while flag:
        print("\nWelcome to Admin panel")
        print("1. Add New User")
        print("2. Delete User")
        print("3. Search User")
        print("4. Modify User")
        print("5. List User")
        print("6. Quit")

        choice = input("Select one: ")
        match choice:
            case "1":
                addUser()
            case "2":
                delUser(loginInfo)
            case "3":
                searchUser()
            case "4":
                flag = modifyUser(loginInfo)
            case "5":
                listUsers()
            case "6":
                break
            case _:
                print("Choice entered not valid, pls try again")
```

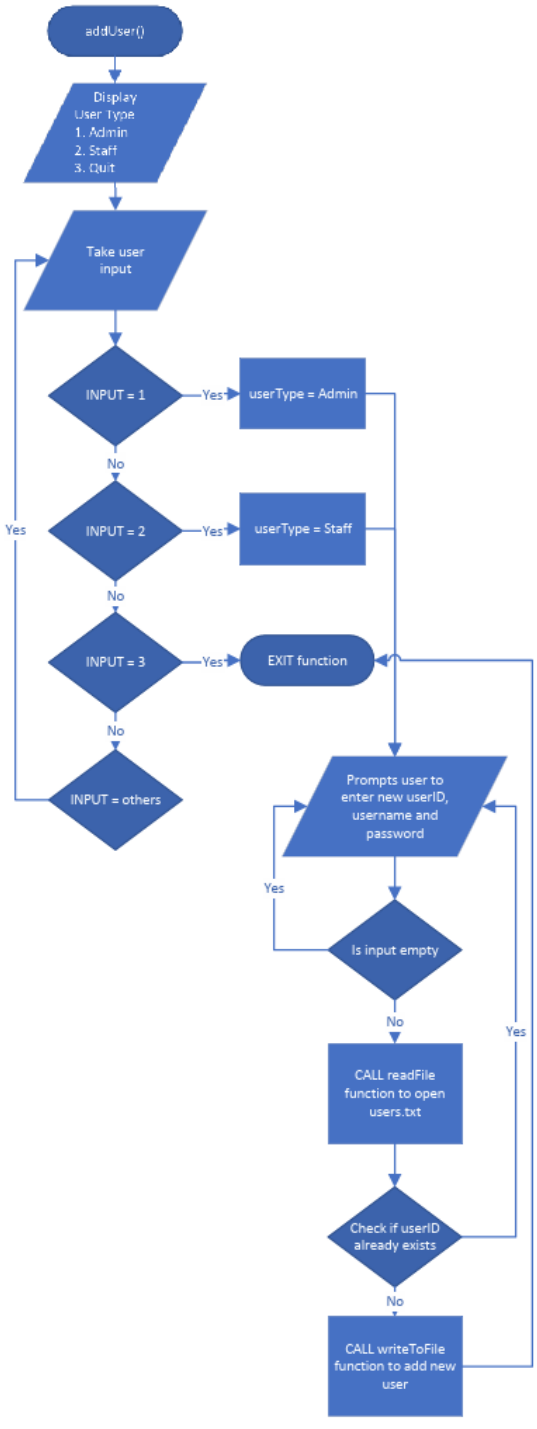
9. addUser

This function allows the admin to create a new admin or staff. It will call the writeToFile function to save the new user in users.txt

Pseudocode

```
addUser function

    ENTER LOOP
        DISPLAY(User Type)
        DISPLAY(1. Admin)
        DISPLAY(2. Staff)
        DISPLAY(3. Quit)

        PROMPTS user to INPUT their hospitalChoice
            INPUT = 1
                userType = "Admin
                EXIT LOOP
            INPUT = 2
                userType = "Staff"
                EXIT LOOP
            INPUT = 3
                EXIT FUNCTION
            INPUT = OTHERS
                DISPLAY(Choice entered not valid, pls try again)

        ENTER LOOP:
            PROMPTS user to input new UserID
            PROMPTS user to input new Name
            PROMPTS user to input new password

            IF newUserID or newName or newPwd is empty
                DISPLAY(Please fill in all the details)
                RESTART LOOP

            CALL readFile function to open users.txt

            Check if userID already existed
                IF False
                    CALL writeToFile function to add new user
                    DISPLAY(Added new User)
                IF True
                    RESTART LOOP
    EXIT FUNCTION
```

Flowchart



Python Code

```
#function for adding a new user as an admin
def addUser():

    userType = None

    while True:
        print("\nUser Type")
        print("1. Admin")
        print("2. Staff")
        print("3. Quit")
        choice = input("Select one: ")
        match choice:
            case "1":
                userType = "Admin"
            case "2":
                userType = "Staff"
            case "3":
                break
            case _:
                print("Choice entered not valid, pls try again")
                continue

        while True:
            newUserID = input("Please enter your userID: ")
            newName = input("Please enter your name: ")
            newPwd = input("Please enter your password: ")

            if newUserID == "" or newName == "" or newPwd ==
"":            print("Please fill in all the details.\n")
                continue

            users = [newUserID, newName, userType, newPwd]
            original = None
            duplicateUserDetected = False

            original = readFile("users.txt")

            for user in original:
                if user[0] == newUserID:
                    print("\nThis userID already exists.\n")
                    duplicateUserDetected = True

            if not duplicateUserDetected:
                original.append(users)
                writeToFile('users.txt',original)
                print("\nAdded New User")
                break
```

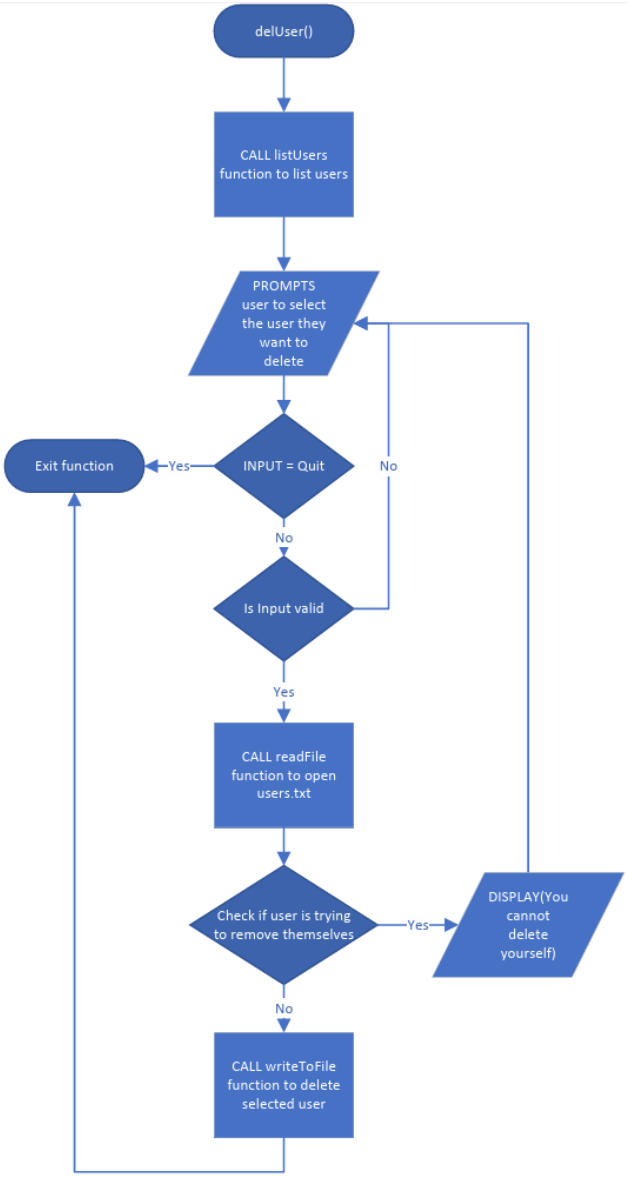
10. delUser

This function allows an admin to remove other users from users.txt. However, the user cannot delete the account they are currently using.

Pseudocode

```
delUser fucntion
    ENTER LOOP
        CALL listUsers function to print out user list
        PROMPTS user to select the user they want to delete

        IF INPUT == Quit
            EXIT LOOP

        IF INPUT is not a number or out of range
            DISPLAY(Error, pls try again)
            RESTART LOOP

        CALL readFile function to open users.txt
        CHECK if user is trying to delete themselves
        IF TRUE
            DISPLAY(You cannot delete yourself)
            RESTART LOOP
        IF FALSE
            CALL writeToFile function to remove user
            DISPLAY(User deleted)
    EXIT FUNCTION
```

Flowchart



Python Code

```python
#function for deleting a user as an admin
def delUser(loginInfo):
    while True:
        print("\nSelect the user you want to delete(Type \"Quit\" to quit): ")
        listUsers()

        delete = input()

        if delete == "Quit":
            break

        try:
            int(delete)
        except:
            print("Value entered is not a valid integer, pls try again")
        else:
            users = readFile('users.txt')

            try:
                users[int(delete)  - 1][0]
            except IndexError:
                print("User doesn\'t exist")

            else:
                if users[int(delete)  - 1][0] == loginInfo[1]:
                    print("You cannot delete yourself.")
                    continue
                else:
                    users.pop(int(delete) - 1)

                writeToFile("users.txt",users)

                print("User deleted\n")
                break
```

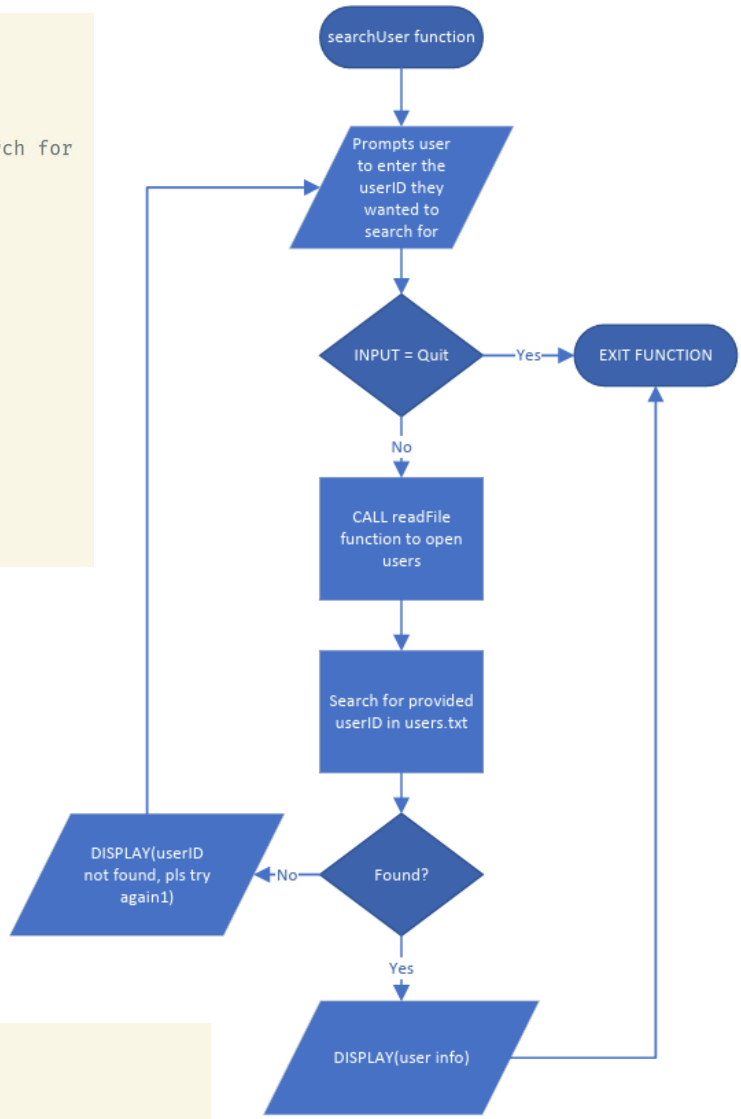
11. searchUser

This function allows an admin to search for another user based on their userID. If the user is found, their user info will be showed or else it will indicate that the user doesn't exists

Pseudocode

```
searchUser function
    ENTER LOOP
        PROMPTS user to enter the userID they want to search for
        if INPUT == Quit
            EXIT LOOP

        CALL readFile function to open users.txt
            SEARCH for user in users.txt
            IF FOUND
                DISPLAY(User Info)
                RESTART LOOP

            IF NOT FOUND
                DISPLAY(User not found)
                RESTART LOOP
    EXIT FUNCTIONA
```

Flowchart



Python Code

```python
#function for searching for a user according to their userID
def searchUser():
    while True:
        userFound = False
        searchTerm = input("\nSearch user by their userID(Type \"Quit\" to quit):")
        if searchTerm == "Quit":
            break

        users = readFile("users.txt")
        for user in users:
            if user[0] == searchTerm:
                print(f"Found user: {user[0]}\nName: {user[1]}\nType: {user[2]}")
                userFound = True

        if not userFound:
            print(f"User code {searchTerm} not found")
```

12. modifyUser

This function allows an admin to change a user from staff to admin or vice versa. The admin is also able to modify the password of a selected user.

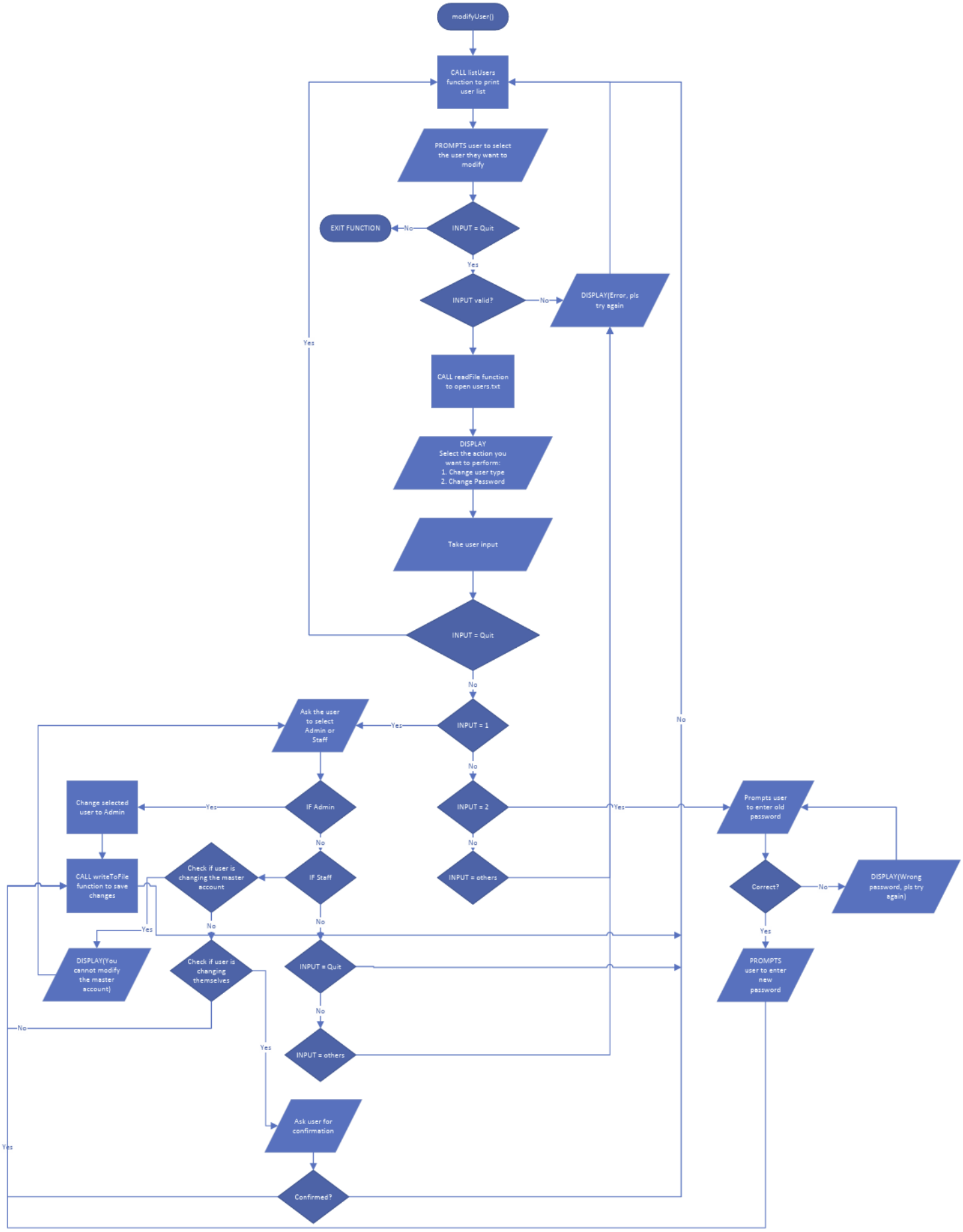
Pseudocode

```
modifyUser function
    ENTER LOOP
        DISPLAY(Select the user you want to modify: )
        CALL listUsers function

        PROMPTS user to select the user they want to modify
        IF INPUT == Quit
            EXIT LOOP
        IF INPUT is not a number or out of range
            DISPLAY(Error, pls try again)
            RESTART LOOP
        CALL readFile function to open users.txt
        DISPLAY(Select the action to perform)
        DISPLAY(1. Change user type)
        DISPLAY(2. Change password)
        PROMPTS user to input their choice
        IF INPUT = Quit
            EXIT LOOP
        IF INPUT = others
            DISPLAY(Error, pls try again)
            RESTART LOOP
        IF INPUT = 1
            ASK the user to select Admin or Staff
                IF Admin
                    Change selected user to Admin
                    CALL writeToFile function to save changes
                IF Staff
                    Change selected user to Staff
                    Check if user is changing the master account
                    IF TRUE
                        DISPLAY(You cannot modify the master account)
                        RESTART LOOP
                    IF FALSE
                        Check if user is changing themselves
                        IF FALSE
                            CALL writeToFile function to save changes
                        IF True
                            ASK user for confirmation
                            IF True
                                CALL writeToFile function to save changes
                                RETURN to mainMENU function
                            IF False
                                RESTART LOOP
        IF INPUT = 2
            ENTER LOOP
                PROMPTS user to enter old password
                CALL readFile function to check if old password is correct
                IF TRUE
                    PROMPTS user to enter new password
                    CALL writeToFile function to save changes
                    EXIT LOOP
                IF FALSE
                    DISPLAY(Old password incorrect)
                    RESTART LOOP
    EXIT FUNCTION
```

Python Code

```python
#function for changing a user from admin to staff or vice versa, or to change password for an account
def modifyUser(loginInfo):
    flag = True
    while flag:
        print("\nSelect the user you want to modify(Type \"Quit\" to quit):")
        listUsers()

        mod = input()

        if mod == "Quit":
            break

        try:
            mod = int(mod)
        except:
            print("Value entered is not a valid integer, pls try again")
        else:
            users = readFile('users.txt')

            try:
                users[mod  - 1][0]
            except IndexError:
                print("User doesn't exist")

            else:
                while flag:
                    print("\nSelect the action to perform(Type \"Quit\" to quit):")
                    print("1. Change user type")
                    print("2. Change password")

                    choice = input()

                    try:
                        if choice == "Quit":
                            break
                        elif choice == "1":
                            while flag:
                                print("\nSelect one(Admin, Staff ; Type \"Quit\" to quit):")
                                changeType = input()
                                match changeType:
                                    case "Admin":
                                        users[mod - 1][2] = "Admin"
                                        writeToFile("users.txt",users)
                                        break
                                    case "Staff":
                                        #the account which is created in initialization cannot be
deleted because at least one admin is required
                                        if mod == 1:
                                            print("This account is the master account, you cannot
change the type of it")
                                            continue
                                        #if user changed themselves from admin to staff, they will be
forced to return to mainMenu and will not be able to access the manage users panel again
                                        elif users[mod - 1][0] == loginInfo[1] and users[mod - 1][2]
== "Admin":
                                            specialChoice = input("You are changing yourself from
Admin to Staff, proceed?(Yes/No): ")
                                            match specialChoice:
                                                case "Yes":
                                                    users[mod - 1][2] = "Staff"
                                                    writeToFile("users.txt",users)
                                                    flag = False
                                                    loginInfo[3] = 'Staff'
                                                    break

                                                case "No":
                                                    continue

                                                case _:
                                                    print("Choice not valid, please try again")
                                                    continue

                                        else:
                                            users[mod - 1][2] = "Staff"
                                            writeToFile("users.txt",users)
                                            break
                                    case "Quit":
                                        break
                                    case _:
                                        print("Error, please enter only \"Admin\" or \"Staff\"")
                        elif choice == "2":
                            while True:
                                oldPass = input("\nType the old password:")
                                newPass = input("Type the new password:")

                                if oldPass == users[mod - 1][3]:
                                    users[mod - 1][3] = newPass

                                    writeToFile("users.txt", users)
                                    break

                                else:
                                    print("Old password isn't correct please try again")

                    except Exception as e:
                        print(e)
    return False
```

Flowchart


modifyUser()
CALL listUsers function to print user list
PROMPTS user to select the user they want to modify
INPUT = Quit
EXIT FUNCTION
No
Yes
INPUT valid?
No
DISPLAY(Error, pls try again
Yes
CALL readFile function to open users.txt
DISPLAY
Select the action you want to perform:
1. Change user type
2. Change Password
Take user input
INPUT = Quit
No
INPUT = 1
Yes
Ask the user to select Admin or Staff
No
INPUT = 2
Yes
Prompts user to enter old password
IF Admin
Yes
Change selected user to Admin
No
IF Staff
INPUT = others
Correct?
No
DISPLAY(Wrong password, pls try again)
Yes
PROMPTS user to enter new password
CALL writeToFile function to save changes
Check if user is changing the master account
Yes
DISPLAY(You cannot modify the master account)
No
Check if user is changing themselves
No
INPUT = Quit
No
INPUT = others
Yes
Ask user for confirmation
Confirmed?
Yes

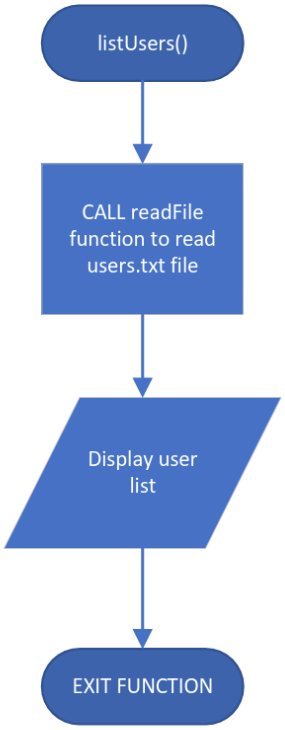
13. listUsers

This function prints out the list of current existing user with all their details in users.txt.

Pseudocode

```
listUsers function
    CALL readFile function to open users.txt file
    READ userinfo from each line
    Print user list
    EXIT FUNCTION
```

Flowchart



Python Code

```
#function to list out all the users info in the users.txt file
def listUsers():
    users = readFile("users.txt")
    print(f"{'No.' : <5}{'User ID' : ^15}{'User Name' : ^15}{'User Type' : ^15}")

    for k,v in enumerate(users):
        print(f"{k+1 : <5}{v[0] : ^15}{v[1] : ^15}{v[2] : ^15}")
```

14. mainMenu

This function presents the main menu and shows the options in PPE Inventory Management System. Users will be able to access different part of the program from here such as inventory, suppliers, hospitals and users.

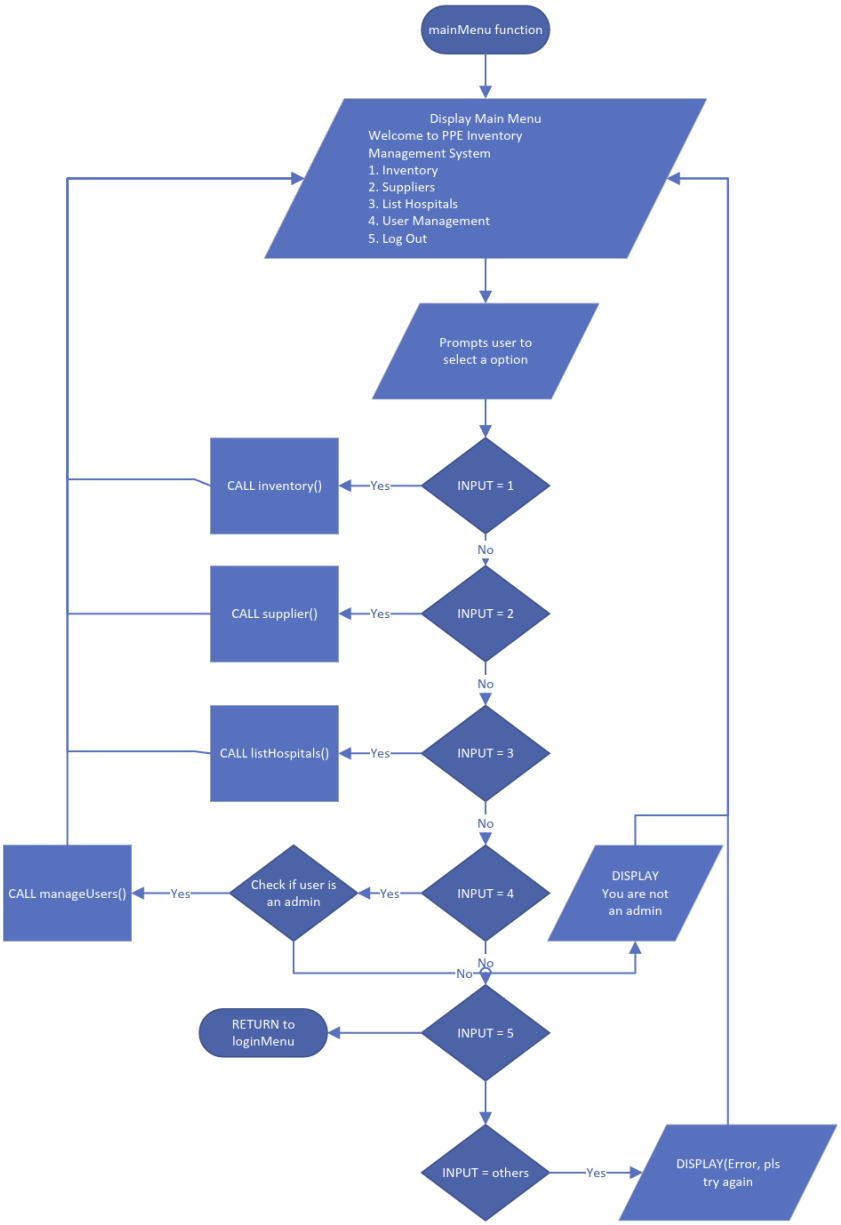
Pseudocode

```
mainMenu function
    ENTER LOOP
        DISPLAY(Welcome to PPE Inventory Management System)
        DISPLAY(1. Inventory)
        DISPLAY(2. Suppliers)
        DISPLAY(3. List Hospitals)
        DISPLAY(4. User Management)
        DISPLAY(5. Log Out)

        PROMPTS user to select an option
        IF INPUT is not an integer or out of range
            DISPLAY(Error, pls try again)
            RESTART LOOP
        IF INPUT == 1
            CALL inventory function
        IF INPUT == 2
            CALL supplier function
        IF INPUT == 3
            CALL listHospitals function
        IF INPUT == 4
            IF user is an Admin
                CALL manageUsers function
            else
                DISPLAY(You are not an admin)
                RESTART LOOP
        IF choice == 5
            EXIT LOOP
            RETURN TO loginMenu
```

Python Code

```
#function for printing out mainMenu and redirecting to each function
def mainMenu(loginInfo):
    while True:
        print("\nWelcome to the PPE Inventory Management System")
        print("1. Inventory")
        print("2. Suppliers")
        print("3. List Hospitals")
        print("4. User Management")
        print("5. Log Out")

        choice = input("Select one: ")

        try:
            int(choice)
        except:
            print("Value entered not a valid integer, pls try again")
        else:
            match int(choice):
                case 1:
                    inventory()
                case 2:
                    supplier()
                case 3:
                    listHospitals()
                case 4:
                    if loginInfo[3] == "Admin":
                        manageUsers(loginInfo)
                    else:
                        print("You are not an admin")
                case 5:
                    print("\n")
                    break

                case _:
                    print("Value entered not a valid choice, pls try again")
```

Flowchart


mainMenu function
Display Main Menu
Welcome to PPE Inventory
Management System
1. Inventory
2. Suppliers
3. List Hospitals
4. User Management
5. Log Out
Prompts user to
select a option
INPUT = 1
Yes
CALL inventory()
No
INPUT = 2
Yes
CALL supplier()
No
INPUT = 3
Yes
CALL listHospitals()
No
INPUT = 4
Yes
Check if user is
an admin
Yes
CALL manageUsers()
DISPLAY
You are not
an admin
No
No
INPUT = 5
RETURN to
loginMenu
INPUT = others
Yes
DISPLAY(Error, pls
try again

15. loginMenu

This function is the main page where the user login. The program will return to this page once the “Log Out” function is selected in mainMenu. Different users can use the program without having to restart.

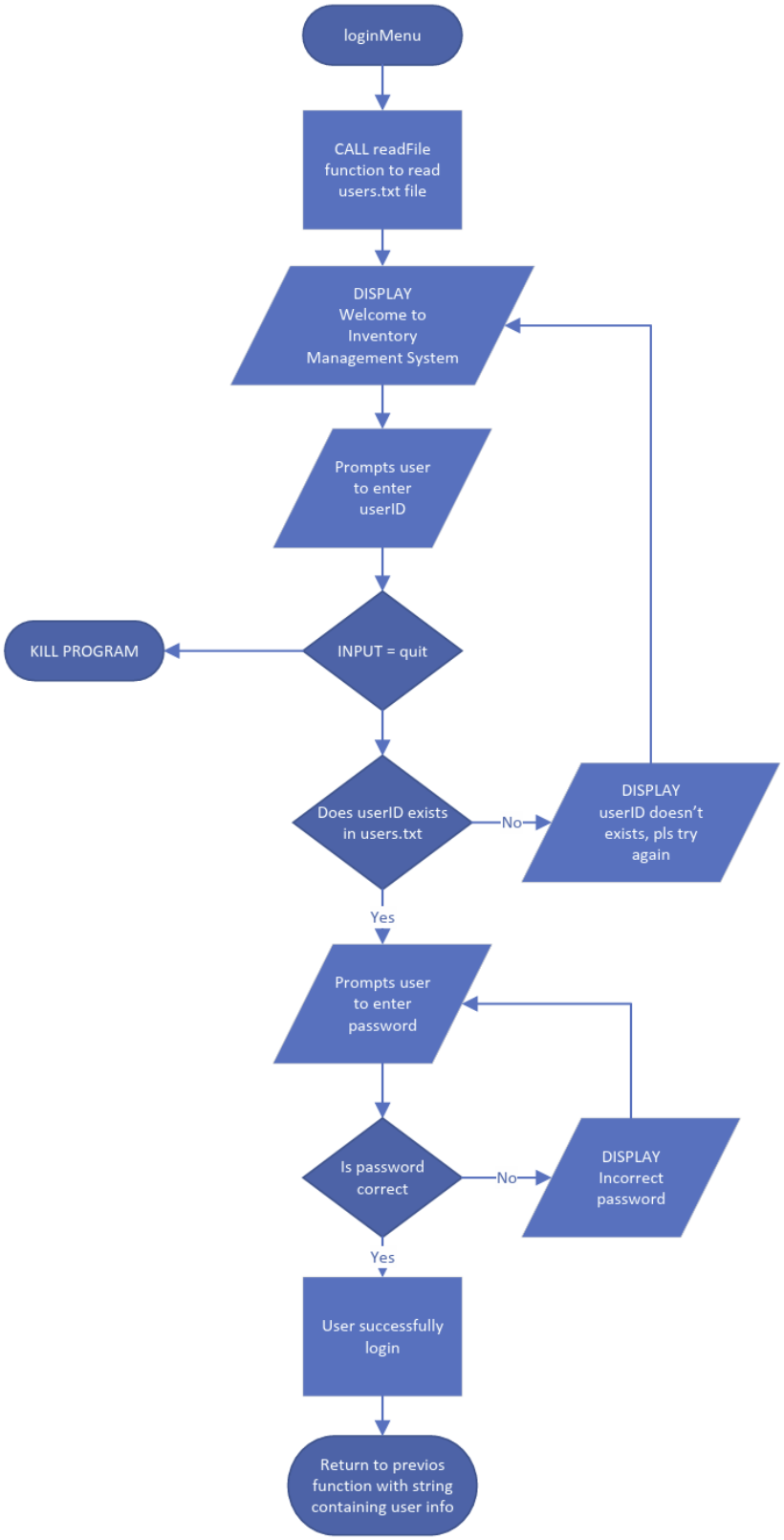
Pseudocode

```
loginMenu function
    CALL readFile function to read users.txt file
    DISPLAY(Welcome to Inventory Management System)
    DISPLAY(Type "quit" to quit the program)
    ENTER LOOP
        PROMPTS user to enter userID
        IF userID exists
            ENTER LOOP
                PROMPTS user to enter password
                IF input matches password in users.txt file
                    user successfully login
                ELSE
                    DISPLAY(Wrong password, pls try again)
                    RESTART LOOP
        IF userID = "quit"
            EXIT program
        IF userID doesn't exist
            DISPLAY(UserID doesn't exist, pls try again)
            RESTART LOOP
```

Python Code

```python
#function for different users to login, selecting "Log Out" in mainMenu will return to this page
def loginMenu():
    users = readFile('users.txt')
    print("\nWelcome to PPE Inventory Management System")
    print("Type \"quit\" to quit the program\n")
    while True:
        userID = input("Please enter your userID: ")
        for k,v in enumerate(users):
            if userID == v[0]:
                while True:
                    password = input("Please enter your password: ")
                    if password == v[3]:
                        return [True, userID,users[k][1], users[k][2]]
                    else:
                        print("Wrong password, pls try again\n")
            elif userID == "quit":
                quit()
        print("User doesn\'t exist, pls try again.\n")
```

Flowchart


loginMenu
CALL readFile function to read users.txt file
DISPLAY Welcome to Inventory Management System
Prompts user to enter userID
INPUT = quit
KILL PROGRAM
Does userID exists in users.txt
No
DISPLAY userID doesn't exists, pls try again
Yes
Prompts user to enter password
Is password correct
No
DISPLAY Incorrect password
Yes
User successfully login
Return to previos function with string containing user info

16. inventoryInit

This function checks for the existence of ppe.txt. If not available, the user will be told to enter the information for 6 PPE items which is item code, item name and supplier of each item. The information will then be saved as strings inside ppe.txt. Before checking the existence of ppe.txt, this function will call supplierInitialize and hospitalInitilize to check for the existence of suppliers.txt and hospitals.txt.

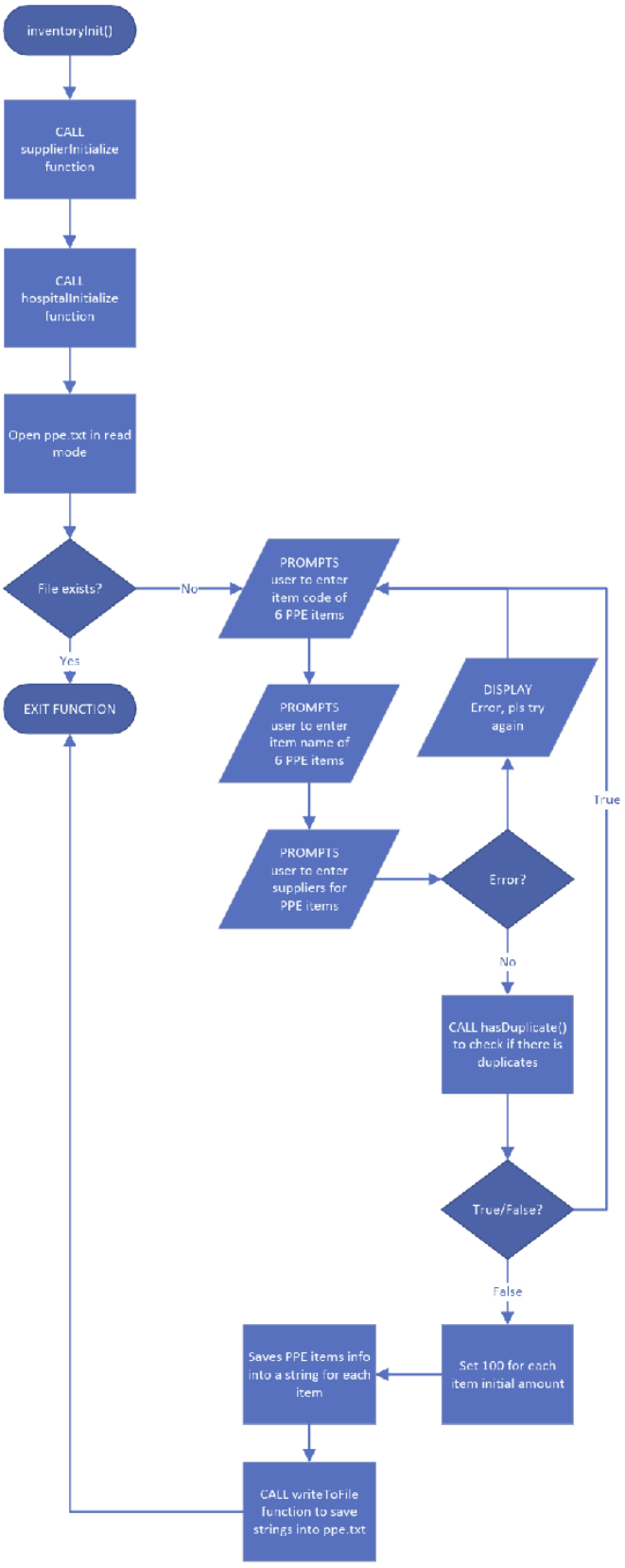
Pseudocode

```
inventoryInit function
    CALL supplierInitialize function to check if suppliers.txt exist
    CALL hospitalInitialize fucntion to check if hospitals.txt exist
    TRY to OPEN ppe.txt file in read mode
        IF file does not exist
            ENTER LOOP
                PROMPTS user to enter 6 item code for PPE item
                    CALL hasDuplicate function to check if item code is duplicated
                        IF True
                            RESTART LOOP
                        IF False
                            CONTINUE
                PROMPTS user to enter 6 item name for PPE item
                    CALL hasDuplicate function to check if item name is duplicated
                        IF True
                            RESTART LOOP
                        IF False
                            CONTINUE
                PROMPTS user to enter 6 supplier for PPE item
                CALL doesItemExists to check if supplier exists in suppliers.txt
                    IF True
                        CONTINUE
                    IF False
                        RESTART LOOP

                ENTER LOOP from 0 to 6
                    SAVE PPE item info into a string as(ppe[i],ppeName[i],ppeSupplier[i],100)

                CALL writeToFile function to save string into ppe.txt
                EXIT LOOP
            IF ERROR exists
                PRINT OUT ERROR
    EXIT FUNCTION
```

Python Code

```
#function for initializing inventory and adding PPE items
def inventoryInit():
    supplierInitialize() #check if suppliers.txt exists
    hospitalInitialize() #check if hospitals.txt exists
    try:
        open("ppe.txt", "r")
    except FileNotFoundError:
        while True:
            ppes = []

            try:
                print("\nPlease enter the details of 6 PPE Items.")
                print("Example: AA,BB,CC,DD,EE,FF\n")

                while True:
                    ppe = list(input("Please enter the all the item code with comma in between:
").strip().split(','))
                    #item code can be "HC,FS,MS,GL,GW,SC"
                    ppeError = hasDuplicates(ppe)
                    if not ppeError:
                        break

                while True:
                    ppeName = list(input("Please enter the all the item name with comma in between:
").strip().split(','))
                    #item name can be "Head Cover,Face Shield,Mask,Gloves,Gown,Shoe Covers"
                    ppeError = hasDuplicates(ppeName)
                    if not ppeError:
                        break

                supplierList = readFile("suppliers.txt")

                while True:
                    supplierError = False
                    ppeSupplier = list(input("Please enter the all the supplier code for each item
with comma in between: ").strip().split(','))
                    #item supplier can be "JJ,Ab,Pf,GSK,JJ,Ab"

                    for sup in ppeSupplier:
                        if not doesItemExists(sup, supplierList):
                            print(f"{sup} does not exist, pls try again")
                            supplierError = True
                            break

                    if not supplierError:
                        break

                for i in range(0,6): #initial quantity for each items is 100, received from suppliers
                    ppes.append([ppe[i], ppeName[i], ppeSupplier[i], "100"])
                    addTranscation(ppe[i], ppeName[i], ppeSupplier[i], "100", "receive")

                writeToFile("ppe.txt", sorted(ppes))
                break
            except IndexError:
                print("Error in input, please try again.\n")

            except Exception as e:
                print(e)
```

Flowchart


inventoryInit()
CALL supplierInitialize function
CALL hospitalInitialize function
Open ppe.txt in read mode
File exists?
No
Yes
EXIT FUNCTION
PROMPTS user to enter item code of 6 PPE items
PROMPTS user to enter item name of 6 PPE items
PROMPTS user to enter suppliers for PPE items
DISPLAY Error, pls try again
Error?
True
No
CALL hasDuplicate() to check if there is duplicates
True/False?
False
Set 100 for each item initial amount
Saves PPE items info into a string for each item
CALL writeToFile function to save strings into ppe.txt

17. inventory

This function is submenu for “Inventory” in mainMenu. Once selected, it will print out the Inventory Menu which consists of different functions for managing inventory such as stock checking, distribution and receiving of items and checking transactions history of items.

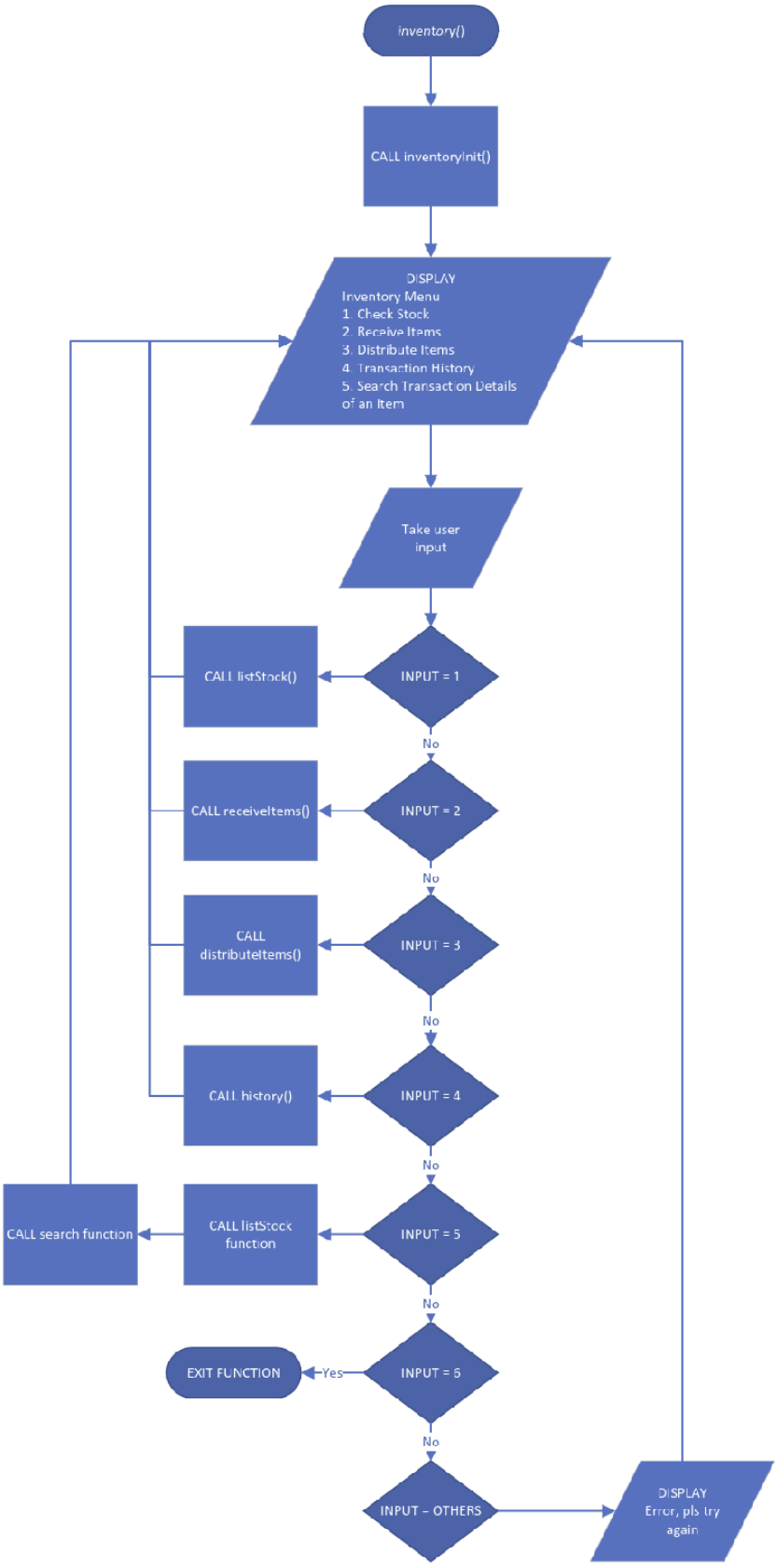
Pseudocode

```
inventory function
    CALL inventoryInit function to check if ppe.txt file is created

    ENTER LOOP
        DISPLAY(Inventory Menu)
        DISPLAY(1. Check Stock)
        DISPLAY(2. Receive Items)
        DISPLAY(3. Distribute Items)
        DISPLAY(4. Transaction History)
        DISPLAY(5. Search Transaction Detail of an Item)
        DISPLAY(6. Quit)

        CALL lessThan25 function to check if any item is less than 25

        TAKE user INPUT
            INPUT = 1
                CALL listStock function
            INPUT = 2
                CALL receiveItems function
            INPUT = 3
                CALL distributeItems function
            INPUT = 4
                CALL history function
            INPUT = 5
                CALL listStock function
                CALL search function
            INPUT = 6
                EXIT FUNCTION
            INPUT = OTHERS
                DISPLAY("Choice entered not valid, pls try again)
```

Python Code

```python
#function for displaying Inventory Menu
def inventory():
    inventoryInit() #checks if ppe.txt have been initialized or not

    while True:
        print("\nInventory")
        print("1. Check Stock")
        print("2. Receive Items")
        print("3. Distribute Items")
        print("4. Transaction History")
        print("5. Search Transaction Detail of an Item")
        print("6. Quit")

        lessThan25()

        choice = input("\nSelect one: ")
        match choice:
            case "1":
                listStock()
            case "2":
                receiveItems()
            case "3":
                distributeItems()
            case "4":
                history()
            case "5":
                listStock()
                search()
            case "6":
                break
            case _:
                print("Choice entered not valid, pls try again")
```

Flowchart


inventory()
CALL inventoryInit()
DISPLAY
Inventory Menu
1. Check Stock
2. Receive Items
3. Distribute Items
4. Transaction History
5. Search Transact on Details of an Item
Take user input
INPUT = 1
CALL listStock()
No
INPUT = 2
CALL receiveItems()
No
INPUT = 3
CALL distributeItems()
No
INPUT = 4
CALL history()
No
INPUT = 5
CALL listStock function
CALL search function
No
INPUT = 6
Yes
EXIT FUNCTION
No
INPUT – OTHERS
DISPLAY
Error, pls try again

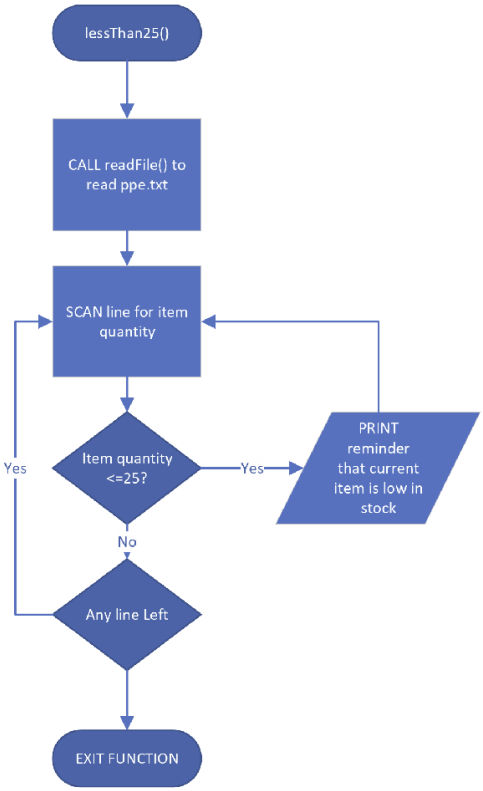
18. lessThan25

This function is called to check if any PPE items quantity is below or equal to 25 in inventory. If the item is low is stock, a reminder will be printed in the inventory menu to remind the user.

Pseudocode

```
lessThan25 function
    CALL readFile function to open ppe.txt in readmode

    ENTER LOOP
        SCAN each line for item quantity
            IF current line quanity ⩽25
                PRINT a reminder that the item is running low in stock
            IF NO line left
                EXIT FUNCTION
```

Flowchart



Python Code

```
#notify user if any item is less than 25 (low in stock)
def lessThan25():
    ppes = readFile("ppe.txt")
    itemLessThan25 = []

    for ppe in ppes:
        if int(ppe[3]) < 25:
            itemLessThan25.append(ppe[1])

    if not itemLessThan25 == []:
        print("Reminder: Items less than 25 boxes")
        for i in itemLessThan25:
            print(i, end=' ')
```

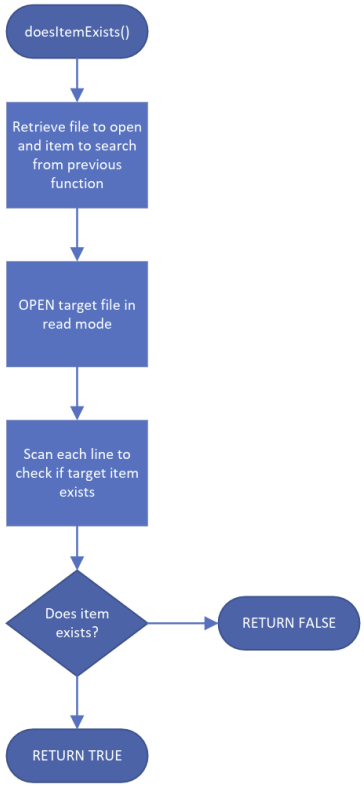
19. doesItemExists

This function checks if the input of the user exists within a specific text file. The function will go through each line and looks for any phrase that is identical to the input. Based on the situation, this function will return either True or False to the previous function.

Pseudocode

```
doesItemExists function with PARAMETER(file,item)
    OPEN file in read mode
    Check if item exist
        IF TRUE
            RETURN TRUE
        IF FALSE
            RETURN FALSE
```

Flowchart



Python Code

```
#checks if the user is looking for an existing item in the txt files
def doesItemExists(element, li):
    for i in li:
        if i[0] == element:
            return True

    return False
```

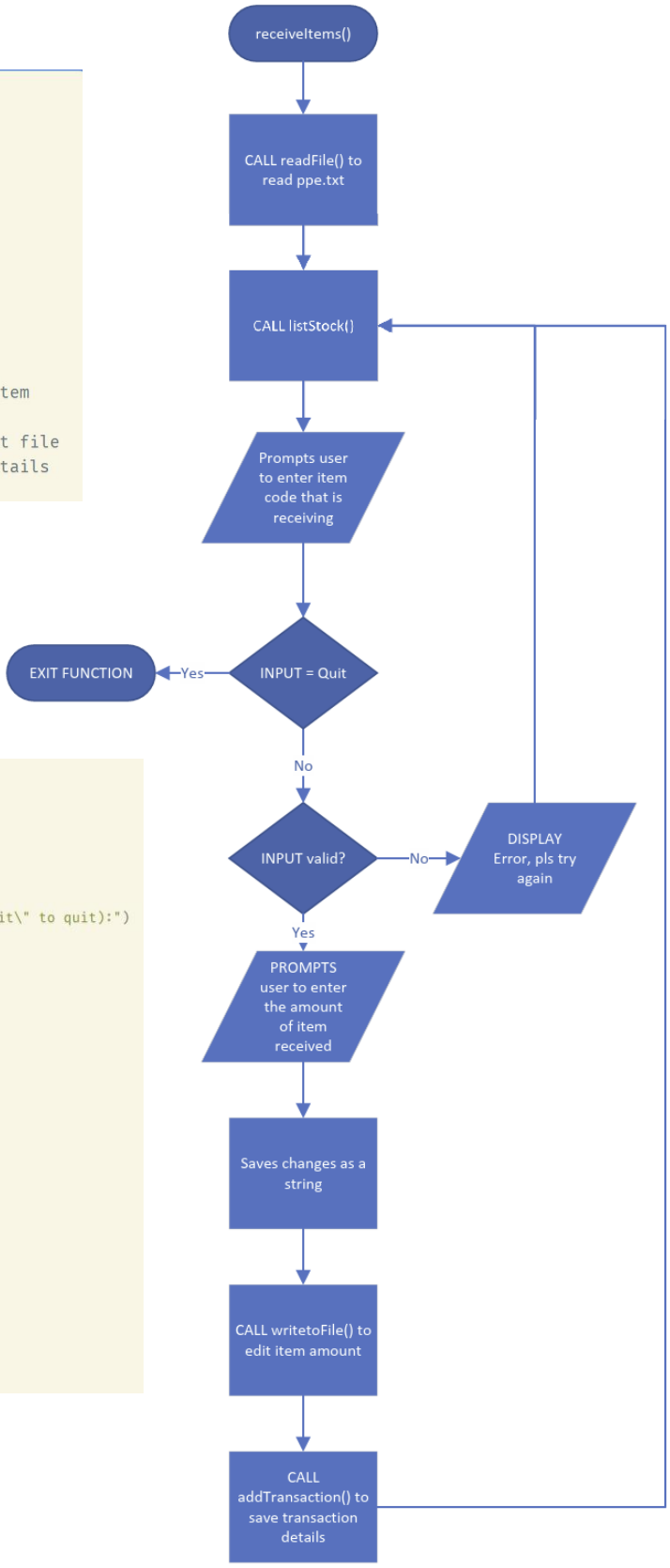
20. receiveItems

This function is selected when the inventory receives stocks from the supplier. Users can select the items that is being restocked and the quantity of item received. The function will call the addTransaction function to save the transaction details with the date and time and also save changes of item quantity in ppe.txt

Pseudocode

```
receiveItems function
    CALL readFile fucntion to read ppe.txt file
    CALL listStock function
    ENTER LOOP
        PROMPTS user to enter item code that is receiving
            IF input = Quit
                EXIT LOOP
            IF input doesn't exist
                DISPLAY(Item doesn't exist, pls try again)
                RESTART LOOP
        PROMPTS user to input the amount received for the item
        SAVE changes to item amount as string
        CALL writeToFile function to make changes to ppe.txt file
        CALL addTranscation function to save transaction Details
```

Flowchart



Python Code

```python
#function for receiving items from suppliers
def receiveItems():
    ppes = readFile("ppe.txt")
    listStock()
    while True:
        choice = input("\nSelect the item receiving(Item Code, Type \"Quit\" to quit):")

        if choice == 'Quit':
            break

        if not doesItemExists(choice, ppes):
            print("Item doesn't exits please try again")
            continue

        amount = input("Input the amount received:")

        try:
            for k,v in enumerate(ppes):
                if v[0] == choice:
                    ppes[k][3] = int(ppes[k][3])
                    ppes[k][3] += int(amount)
                    ppes[k][3] = str(ppes[k][3])

                    writeToFile("ppe.txt", ppes)
                    addTranscation(v[0],v[1],v[2],amount,"receive")
                    break
        except Exception as e:
            print(e)
```

21. distributeItems

Similar to receiveItems, this function is used when items in inventory is distributed to hospitals. A user can select which item to distribute and its amount and select which hospital to distribute it to. This function will also check if there is sufficient amount of selected items to be distributed. Once done, the distribution details with the date and time will be added to transaction.txt and distribution.txt. The quantity of items in stock will also be deducted.

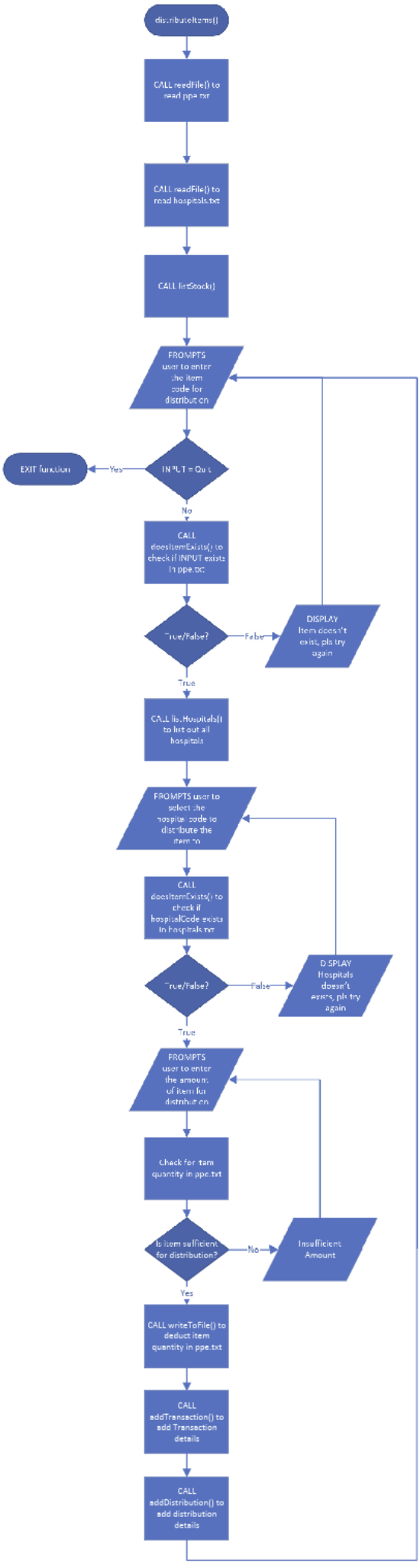
Pseudocode

```
distributeItems function
    CALL readFile function to read ppe.txt file
    CALL readFile function to read hospitals.txt file
    CALL listStock function
    ENTER LOOP
        PROMPTS user to enter the item for distribute by typing item code

        IF INPUT = Quit
            EXIT LOOP
        CALL doesItemExists function to check if INPUT exists in file
            IF FALSE
                DISPLAY(Item doesn't exist, pls try again)
                RESTART LOOP

            ENTER LOOP
                CALL listHospitals function to list out hospitals
                PROMPTS user to select the hospital to distribute the item to by typing hospital code
                CALL doesItemExists to check if INPUT exists in file
                    IF False
                        DISPLAY(The hospital doesn't exists, pls try again)
                        RESTART LOOP
                    IF True
                        ENTER LOOP
                            PROMPTS user to enter the amount for distributing
                            CHECK if quantity of item is sufficient for distributing
                                IF FALSE
                                    DISPLAY(Insufficient Amount)
                                    RESTART LOOP
                                IF TRUE
                                    SAVE changes to quantity in ppe.txt
                                    CALL addTranscaction function to add Transcaction Details
                                    CALL addDistribution function to add distribution details
                                    EXIT LOOP
                EXIT LOOP
    EXIT FUNCTION
```

Python Code

```
#function for distributing items to hospitals, deducts items quantities from inventory
def distributeItems():
    ppes = readFile("ppe.txt")
    hospitals = readFile("hospitals.txt")
    listStock()
    while True:
        choice = input("\nSelect the item distributing(Item Code, Type \"Quit\" to quit):")

        if choice == 'Quit':
            break

        if not doesItemExists(choice, ppes):
            print("Item doesn't exits please try again")
            continue

        while True:
            listHospitals()
            hospitalChoice = input("\nSelect the hospital distributing to(Hospital Code):")

            if not doesItemExists(hospitalChoice, hospitals):
                print("Hospital doesn't exits please try again")
                continue

            while True:
                amount = input("Input the amount distributed:")

                try:
                    for k,v in enumerate(ppes):
                        if v[0] == choice:
                            ppes[k][3] = int(ppes[k][3])

                            # Check for sufficient quantity
                            if ppes[k][3] - int(amount) >= 0:

                                ppes[k][3] -= int(amount)
                                ppes[k][3] = str(ppes[k][3])

                                writeToFile("ppe.txt", ppes)
                                addTranscation(v[0],v[1],hospitalChoice,amount,"distribute")
                                addDistribution(v[0],v[1],hospitalChoice,amount)

                                break

                            else:
                                print(f"Insufficient amount, current stock left is {ppes[k][3]},
please try again")

                            ppes[k][3] = str(ppes[k][3])

                    break
                except Exception as e:
                    print(e)
            break
```

Flowchart


distributeItems()
CALL readFile() to read ppe.txt
CALL readFile() to read hospitals.txt
CALL listStock()
PROMPTS user to enter the item code for distribution
EXIT function
Yes
INPUT = Quit
No
CALL doesItemExists() to check if INPUT exists in ppe.txt
True/False?
False
DISPLAY Item doesn't exist, pls try again
True
CALL lisHospitals() to list out all hospitals
PROMPTS user to select the hospital code to distribute the item to
CALL doesItemExists() to check if hospitalCode exists in hospitals.txt
True/False?
False
DISPLAY Hospitals doesn't exists, pls try again
True
PROMPTS user to enter the amount of item for distribution
Check for item quantity in ppe.txt
Is item sufficient for distribution?
No
Insufficient Amount
Yes
CALL writeToFile() to deduct item quantity in ppe.txt
CALL addTransaction() to add Transaction details
CALL addDistribution() to add distribution details

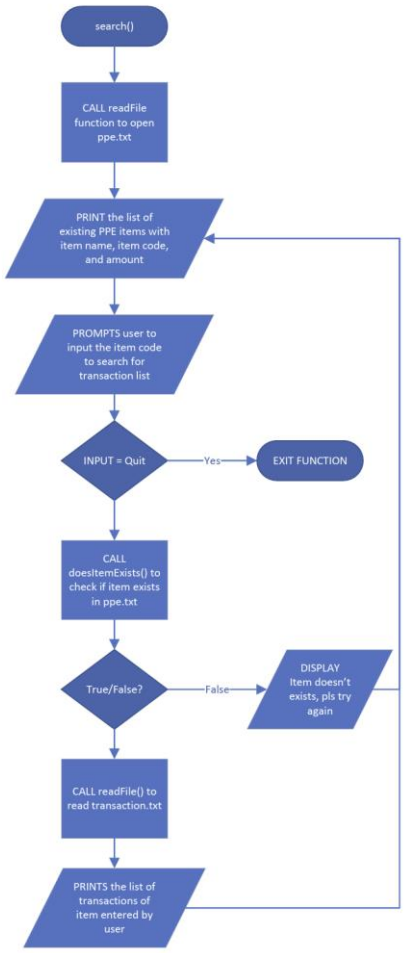
22. search

This function is for searching for a transaction detail of a specific item, no matter received or distributed. The user will be required to key in the item code for searching, and the function will return of the list of transaction for that item with its details if founded.

Pseudocode

```
search function
    CALL readFile function to read ppe.txt file
    ENTER LOOP
        List the existing PPE items with item name, item code and amount
        PROMPTS user to INPUT the item code to search for transaction list
            IF INPUT = quit
                EXIT LOOP
            CALL doesItemExists to check if item exist
            IF False
                DISPLAY(Item doesn't exist, pls try again)
                RESTART LOOP
            IF TRUE
                CALL readFile function to read transaction.txt file
                PRINT the list for tranaction of the item entered by user
    EXIT FUNCTION
```

Flowchart



Python Code

```
#function for searching for the transaction details of an item
def search():
    ppes = readFile("ppe.txt")

    while True:
        item = input("Please enter the item code(Type \"Quit\" to quit): ")

        if item == "Quit":
            break

        if not doesItemExists(item, ppes):
            print("Item doesn't exist, please try agian")
            continue

        transactions = readFile("transaction.txt")
        print(f"\n{'Transaction Time' : <30}{'Item Name' : ^20}{'Item Code' : ^20}{'Item Quantity' : ^15}{'Supplier or Hospital Code' : ^40}{'Status' : ^10}")

        for v in transactions:
            if item == v[2]:
                print(f"{v[0] : <30}{v[1] : ^20}{v[2] : ^20}{v[3] : ^15}{v[4] : ^40}{v[5] : ^10}")

        break
```

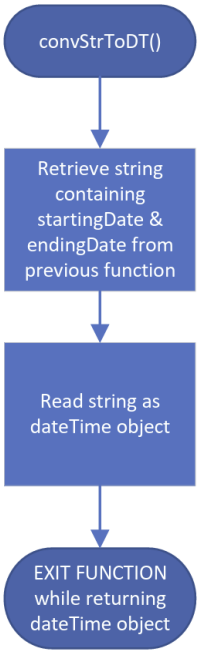
23. convStrToDT

This function is converts a string that is in a date and time format that exists within a text file to a dateTime object which is read by the external datetime library. This function made it possible so that the users can look for the transaction history of a particular item within it's date range.

Pseudocode

```
convStrToDT function
    READ string as dateTime object
    EXIT FUNCTION and RETURN object
```

Flowchart



Python Code

```
#function for converting string that contains date & time info in a file to dateTime format
def convStrToDT(s):
    date = datetime.datetime.strptime(s, "%Y-%m-%d %H:%M:%S.%f").strftime("%d/%m/%Y")
    return datetime.datetime.strptime(date, "%d/%m/%Y")
```

24. history

This function brings out the menu for transaction history from the Inventory submenu. Users can look up the full transaction history of distributing and receiving items. User can also look up for the transaction list and details between a date range. If the starting date and ending date are left blank, the full transaction history will be shown.

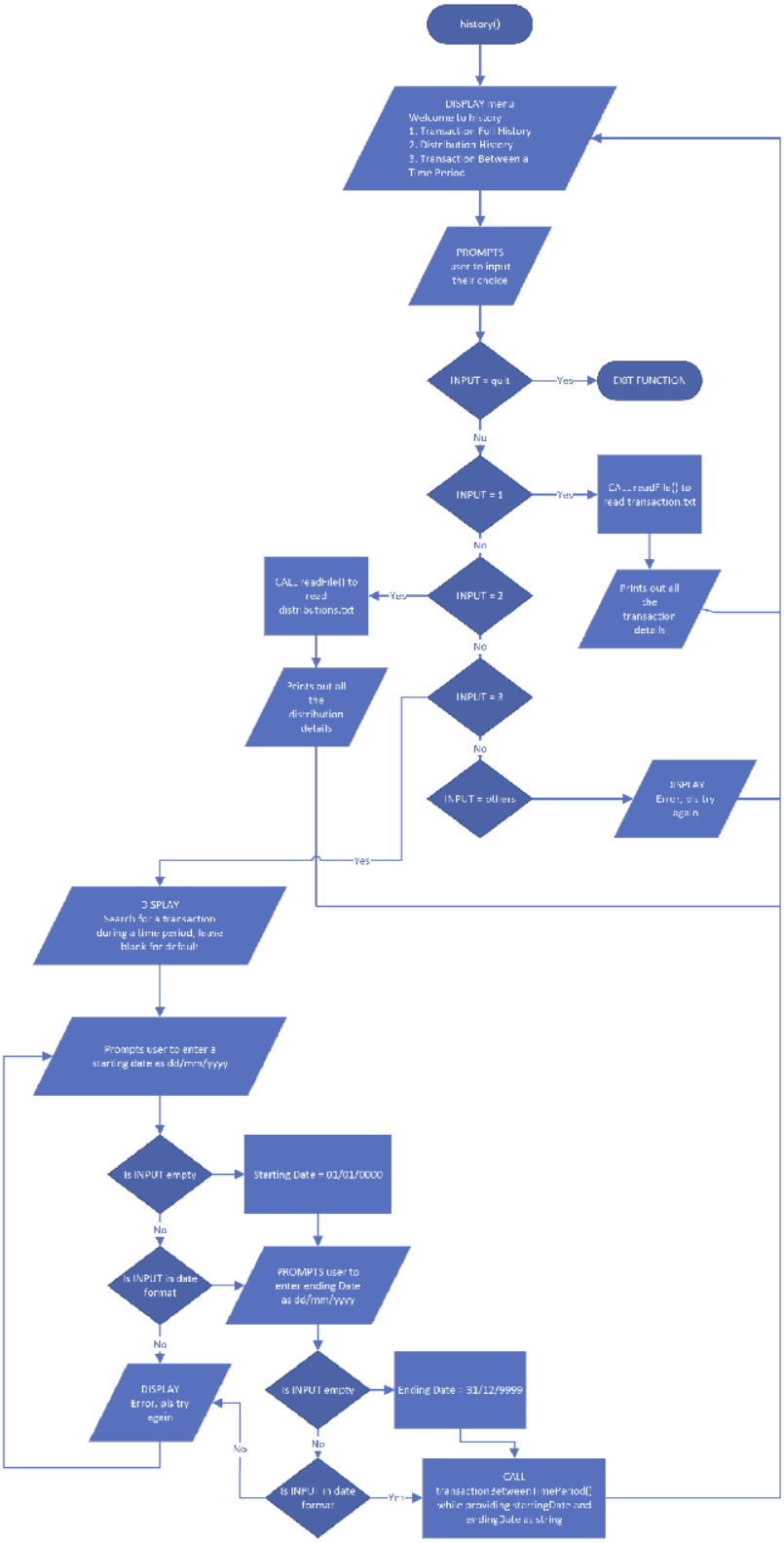
Pseudocode

```
history function
    ENTER LOOP
        DISPLAY(Welcome to history)
        DISPLAY(1. Transaction Full History)
        DISPLAY(2. Distribution History)
        DISPLAY(3. Transaction Between a Time Period)

        Prompts user to enter their choice
        IF INPUT = quit
            EXIT LOOP
        IF INPUT = 1
            CALL readFile function to read tranaction.txt
            PRINT all transaction details
        IF INPUT = 2
            CALL readFile function to read distribution.txt
            Print all distribution details
        IF INPUT = 3
            ENTER LOOP
                DISPLAY(Search for a transactions during a time period,leave blank for default)
                Prompts user to enter the starting date as dd/mm/yyyy
                Prompts user to enter the ending date as dd/mm/yyyy
                    IF starting date and ending date are empty
                        Starting Date = 01/01/0001
                        Ending Date = 31/12/9999
                    IF starting date is empty
                        Starting Date = 01/01/0001
                        ending date = INPUT
                    IF ending Date is empty
                        Ending Date = 31/12/9999
                        Starting Date = INPUT
                Checks if starting date and ending date are valid
                    IF TRUE
                        CALL transactionBetweenTimePeriod function while providing startingDate and
endingDate
                        EXIT LOOP
                    IF FALSE
                        DISPLAY(Error, pls try again)
                        RESTART LOOP
```

Python Code

```python
#function for listing the transaction history of an item
def history():
    while True:
        print("\nWelcome to history")
        print("1. Transaction Full History")
        print("2. Distribution History")
        print("3. Transaction Between a Time Period")

        choice = input("\nPlease select one(Type \"Quit\" to quit): ")

        match choice:
            case "Quit":
                break

            case "1":
                transactions = readFile("transaction.txt")
                print(f"\n{'Transaction Time' : <30}{'Item Name' : ^20}{'Item Code' : ^20}{'Item
Quantity' : ^15}{'Supplier or Hospital Code' : ^40}{'Status' : ^10}")

                for v in transactions:
                    print(f"{v[0] : <30}{v[1] : ^20}{v[2] : ^20}{v[3] : ^15}{v[4] : ^40}{v[5] :
^10}")
            case "2":
                try:
                    open("distribution.txt")
                except FileNotFoundError:
                    print("No distributions have been created yet.")
                else:
                    distributions = readFile("distribution.txt")
                    print(f"\n{'Distribution Time' : <30}{'Item Name' : ^20}{'Item Code' : ^20}{'Item
Quantity' : ^15}{'Hospital Code' : ^40}")

                    for v in distributions:
                        print(f"{v[0] : <30}{v[1] : ^20}{v[2] : ^20}{v[3] : ^15}{v[4] : ^40}")

            case "3":
                while True:
                    #if both are left blank, all history will be shown
                    #if starting date is left blank, all history from the beginning till the ending
date will be shown
                    #if ending date is left blank, all history from starting date till the end will
be shown
                    print("\nSearch for transactions during a time period\n*leave blank for
default*")
                    sDate = input("Please input the starting date(dd/mm/yyyy): ")
                    eDate = input("Please input the ending date(dd/mm/yyyy): ")

                    sDateDT = None
                    eDateDT = None

                    if sDate == '' and eDate == '':
                        sDateDT = datetime.datetime.strptime("01/01/0001", "%d/%m/%Y")
                        eDateDT = datetime.datetime.strptime("31/12/9999", "%d/%m/%Y")

                    elif sDate == '':
                        sDateDT = datetime.datetime.strptime("01/01/0001", "%d/%m/%Y")
                        try:
                            eDateDT = datetime.datetime.strptime(eDate, "%d/%m/%Y")
                        except Exception as e:
                            print(e)
                            continue

                    elif eDate == '':
                        try:
                            sDateDT = datetime.datetime.strptime(sDate, "%d/%m/%Y")
                        except Exception as e:
                            print(e)
                            continue
                        eDateDT = datetime.datetime.strptime("31/12/9999", "%d/%m/%Y")

                    else:
                        try:
                            sDateDT = datetime.datetime.strptime(sDate, "%d/%m/%Y")
                            eDateDT = datetime.datetime.strptime(eDate, "%d/%m/%Y")

                        except Exception as e:
                            print(e)
                            continue

                    transactionBetweenTimePeriod(sDateDT,eDateDT)
                    break
```

Flowchart


history()
DISPLAY menu
Welcome to history
1. Transaction Full History
2. Distribution History
3. Transaction Between a Time Period
PROMPTS user to input their choice
INPUT = quit
Yes
EXIT FUNCTION
No
INPUT = 1
Yes
CALL readFile() to read transaction.txt
Prints out all the transaction details
No
INPUT = 2
Yes
CALL readFile() to read distributions.txt
Prints out all the distribution details
No
INPUT = 3
No
INPUT = others
DISPLAY Error, pls try again
Yes
DISPLAY Search for a transaction during a time period, leave blank for default
Prompts user to enter a starting date as dd/mm/yyyy
Is INPUT empty
Starting Date = 01/01/0000
No
Is INPUT in date format
PROMPTS user to enter ending Date as dd/mm/yyyy
No
DISPLAY Error, pls try again
Is INPUT empty
Ending Date = 31/12/9999
No
Is INPUT in date format
Yes
CALL transactionBetweenTimePeriod() while providing startingDate and endingDate as string

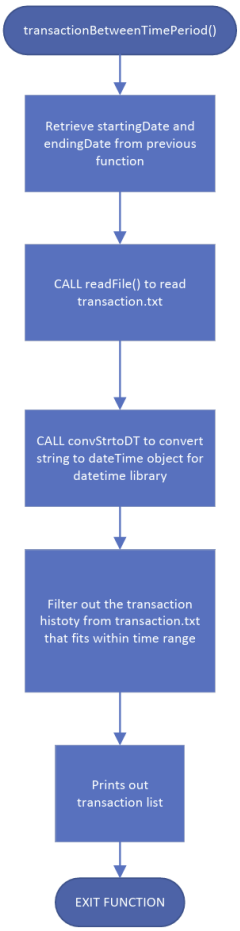
25. transactionBetweenTimePeriod

This function is called when the user intends to search for the transaction history that falls within two date range. The function will first read the transaction history from transaction.txt, then filters out the history that's fits within the given dates. Finally, it will print out the transaction history with its details that falls between the two dates.

Pseudocode

```
transactionBetweenTimePeriod function with parameters(startingDate, endingDate)
    CALL readFile function to read transaction.txt file
    CALL convStrToDT function to convert string from transaction.txt into dateTime object
    Filter out the transaction that fits within time range
    Print out transaction list
```

Flowchart



Python Code

```
#function for searching for the transaction history between two dates
def transactionBetweenTimePeriod(startDate, endDate):
    transactions = readFile("transaction.txt")
    transactionDates = []

    for transaction in transactions:
        transactionDates.append(convStrToDT(transaction[0]))

    print(f"\n{'Transaction Date' : ^26}{'Item Name' : ^20}{'Item Code' : ^20}{'Quantity' : ^20}
{'Supplier Or Hospital Code' : ^25}{'Status' : ^20}")
    for k, date in enumerate(transactionDates):
        if date ≥ startDate and date ≤ endDate:
            # filteredTransactions.append(k)
            print(f"{transactions[k][0] : <26}{transactions[k][1] : ^20}{transactions[k][2] : ^20}
{transactions[k][3] : ^20}{transactions[k][4] : ^25}{transactions[k][5] : ^20}")
```

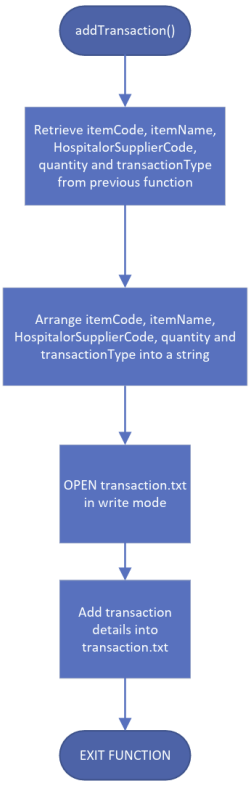
26. addTransaction

This function is called when a transaction is made so that the transaction details with its date and time can be added to transaction.txt. The transaction details is sent to this function which will then be converted into a string. Finally, the strings will be added to the end of transaction.txt. The transaction type, being “distributed” or “received” will also be taken care of.

Pseudocode

```
addTranscaction function with parameters(itemCode, itemName, code, quantity, transactionType)
    OPEN transaction.txt
    IF transactionType = receive
        Add details of transaction into transaction.txt file as a string
    IF transactionType = distribute
        Add details of transaction into transaction.txt file as a string
```

Flowchart



Python Code

```
#function for adding a transaction and its details to transaction.txt
def addTranscation(itemCode, itemName, supplierOrHospitalCode, quantity, transactionType):
    with open("transaction.txt","a") as f:
        if transactionType == "receive":
            f.write(f'{datetime.datetime.now()},{itemName},{itemCode},{quantity},
{supplierOrHospitalCode},received')
        elif transactionType == "distribute":
            f.write(f'{datetime.datetime.now()},{itemName},{itemCode},{quantity},
{supplierOrHospitalCode},distributed')
        f.write('\n')
```

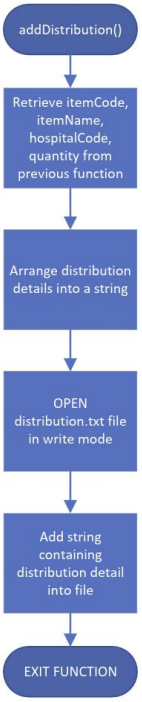
27. addDistribution

Similar to addTransaction, this function adds the details of the distribution of a particular item which includes, item distributed, the quantity and the name of hospital that the item is distributed to into distribution.txt.

Pseudocode

```
addDistribution function with parameters(itemCode, itemName, hospitalCode, quantity)
    OPEN distribution.txt file
        Add parameters to file
```

Flowchart



Python Code

```
#function for adding a distribution and its details to distribution.txt
def addDistribution(itemCode, itemName, hospitalCode, quantity):
    with open("distribution.txt","a") as f:
        f.write(f'{datetime.datetime.now()},{itemName},{itemCode},{quantity},{hospitalCode}')
        f.write('\n')
```

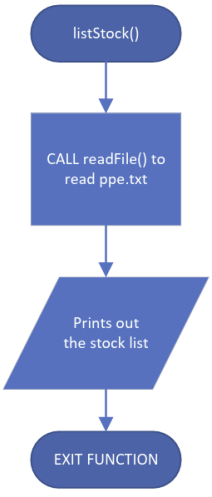
28. listStock

This function reads ppe.txt and prints out the item code, item name and its current quantity.

Pseudocode

```
listStock function
    CALL readFile function to open ppe.txt
    Print out the stock list
```

Flowchart



Python Code

```
#function for listing the remaining stock
def listStock():
    ppe = readFile("ppe.txt")
    print(f"\n{'Item Code' : <10}{'Item Name' : ^20}{'Item Quantity' : ^15}")

    for v in ppe:
        print(f"{v[0] : <10}{v[1] : ^20}{v[3] : ^15}")
```

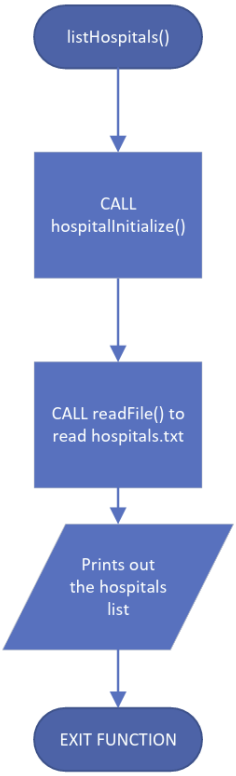
29. listHospitals

This function list all the hospitals with its details when “List Hospitals” is selected in the mainMenu. Before proceeding, it will call hospitalInitialize function to check for the existence of hospitals.txt than contains the crucial information of hospitals.

Pseudocode

```
listHospitals function
    CALL hospitalInitialize function to check if hospitals.txt exists
    CALL readFile function to read hospitals.txt
    Print out the hospitals list
```

Flowchart



Python Code

```
#function for listing the hospitals
def listHospitals():
    hospitalInitialize()
    hospitals = readFile("hospitals.txt")
    print(f"\n{'Hospital Code' : <15}{'Hospital Name' : ^40}")

    for v in hospitals:
        print(f"{v[0] : <15}{v[1] : ^40}")
```

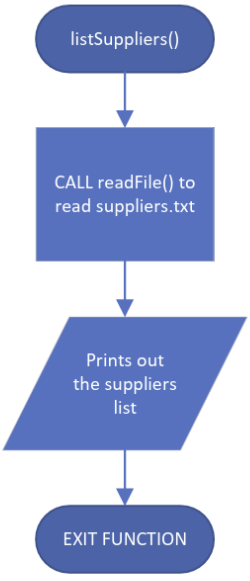
30. listSuppliers

Similarly, this function prints out the suppliers details from suppliers.txt.

Pseudocode

```
listSuppliers function
    CALL readFile function to read suppliers.txt
    Print out the suppliers list
```

Flowchart



Python Code

```
#function for listing suppliers
def listSuppliers():
    suppliers = readFile("suppliers.txt")
    print(f"\n{'Supplier Code' : <15}{'Supplier Name' : ^25}{'Supplier Contact' : ^20}")

    for v in suppliers:
        print(f"{v[0] : <15}{v[1] : ^25}{v[2] : ^20}")
```

31. suppliers

This function prints out the submenu for “Suppliers” from the mainMenu, users can choose from different options such as listing supplier details, changing suppliers’ name or contact number. The action of changing suppliers’ name and contact number are done within the function.

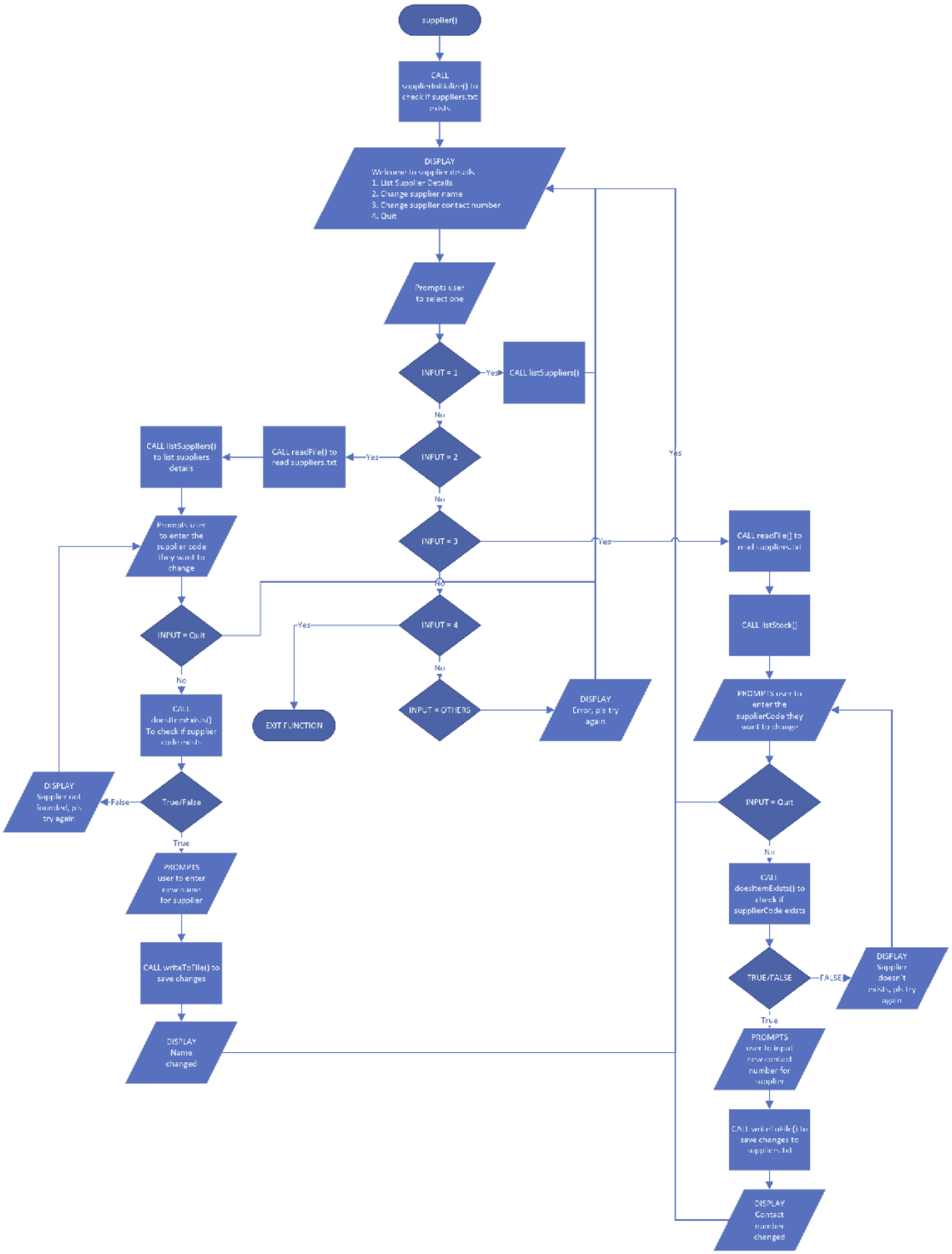
Pseudocode

```
supplier function
    call supplierInitialize function to check if suppliers.txt exists
    Enter LOOP
        DISPLAY(Welcome to supplier details)
        DISPLAY(1. List Supplier Details)
        DISPLAY(2. Change supplier name)
        DISPLAY(3. Change supplier contact number)
        DISPLAY(4. Quit)
        Prompts user to select one
            IF INPUT = 4
                EXIT LOOP
            IF INPUT = 1
                CALL listSuppliers function
                CONTINUE LOOP
            IF INPUT = 2
                CALL readFile function to read suppliers.txt
                CALL listSuppliers function
                ENTER LOOP
                    PROMPTS user to enter the supplier code they want to change
                        IF INPUT = Quit
                            EXIT LOOP
                        CALL doesItemExists to check if supplier code exists
                            IF FALSE
                                DISPLAY(Error, pls try again)
                                RESTART LOOP
                            ELSE
                                PROMPTS user to enter the new name for supplier
                                CALL writeToFile function to save changes
                                DISPLAY(Name changed)
                                EXIT LOOP
            IF INPUT = 3
                CALL readFile function to read suppliers.txt
                CALL listStock function
                ENTER LOOP
                    PROMPTS user to enter the supplier code they want to change
                        IF INPUT = Quit
                            EXIT LOOP
                        CALL doesItemExists function to check if supplier code
exists                      IF FALSE
                                Supplier doesn't exist, pls try again
                                RESTART LOOP
                            IF TRUE
                                PROMPTS user to enter new contact number for
supplier                        CALL writeToFile function to save changes
                                DISPLAY(Contact number changes)
                                EXIT LOOP
            IF INPUT = others
                DISPLAY(Error, pls try again)
```

Python Code

```python
#function for printing out supplier menu and redirecting to the functions
def supplier():
    supplierInitialize()
    while True:
        print("\nWelcome to supplier details")
        print("1. List Supplier Details")
        print("2. Change Supplier Name")
        print("3. Change Supplier Contact Number")
        print("4. Quit")

        choice = input("Select one: ")

        if choice == "4":
            break

        match choice:
            case "1":
                listSuppliers()
                continue
            case "2":
                suppliers = readFile("suppliers.txt")
                listSuppliers()

                while True:
                    print("Select the supplier you want to change(Supplier Code, Type \"Quit\" to
quit):")
                    supToChange = input()

                    if supToChange == "Quit":
                        break

                    if not doesItemExists(supToChange, suppliers):
                        print("Supplier doesn't exist please try again")
                        continue

                    for k,v in enumerate(suppliers):
                        if v[0] == supToChange:
                            suppliers[k][1] = input("Enter the new name:")
                            writeToFile("suppliers.txt", suppliers)
                            print("New name changed")
                            break
            case "3":
                suppliers = readFile("suppliers.txt")
                listSuppliers()

                while True:
                    print("Select the supplier you want to change(Supplier Code, Type \"Quit\" to
quit):")
                    supToChange = input()

                    if supToChange == "Quit":
                        break

                    if not doesItemExists(supToChange, suppliers):
                        print("Supplier doesn't exist please try again")
                        continue

                    for k,v in enumerate(suppliers):
                        if v[0] == supToChange:
                            suppliers[k][2] = input("Enter the new contact number:")
                            writeToFile("suppliers.txt", suppliers)
                            print("Contact number changed")
                            break

            case _:
                print("Choice entered is not valid, please try again")
```

Flowchart


supplier()
CALL supplierInitialize() to check if suppliers.txt exists
DISPLAY
Welcome to supplier details
1. List Supplier Details
2. Change supplier name
3. Change supplier contact number
4. Quit
Prompts user to select one
INPUT = 1
Yes
CALL listSuppliers()
No
INPUT = 2
Yes
CALL readFile() to read suppliers.txt
CALL listSuppliers() to list suppliers details
No
INPUT = 3
Yes
CALL readFile() to read suppliers.txt
CALL listStock()
PROMPTS user to enter the supplierCode they want to change
INPUT = Quit
No
CALL doesItemExists() To check if supplier code exists
TRUE/FALSE
FALSE
DISPLAY Supplier doesn't exists, pls try again
True
PROMPTS user to input new contact number for supplier
CALL writeToFile() to save changes to suppliers.txt
DISPLAY Contact number changed
No
INPUT = 4
Yes
EXIT FUNCTION
No
INPUT = OTHERS
DISPLAY Error, pls try again
Prompts user to enter the supplier code they want to change
INPUT = Quit
No
CALL doesItemExists() To check if supplier code exists
True/False
False
DISPLAY Supplier not founded, pls try again
True
PROMPTS user to enter new name for supplier
CALL writeToFile() to save changes
DISPLAY Name changed

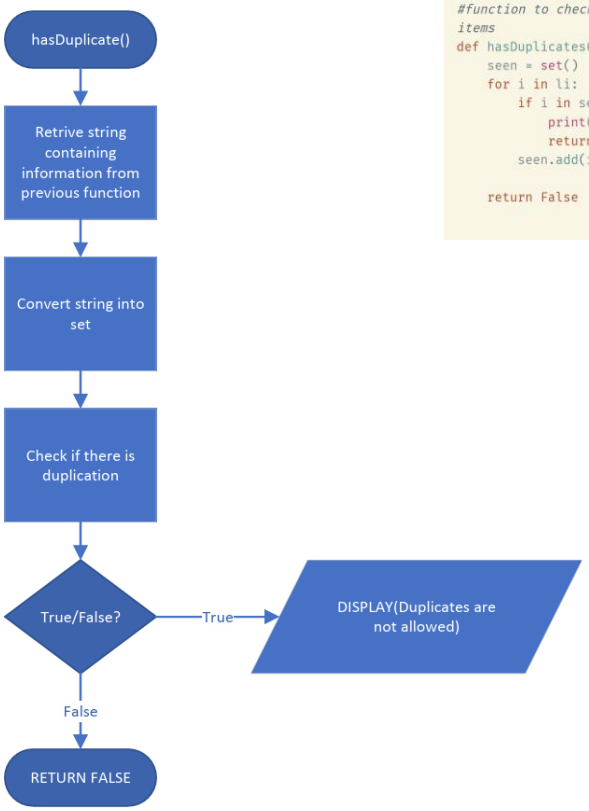
32. hasDuplicate

This function is called to check if there are duplicates among codes, names and contact numbers as these cannot be the same. It converts the strings to sets to check if there is duplicate. If true, it will print out an error message.

Pseudocode

```
hasDuplicate function
    RECEIVE string containing codes/names/contactNumber from previous function
    CONVERT string into set
    Checks for duplicatation
        IF True
            DISPLAY(Duplicates are not allowed)
            Return TRUE
        If False
            Return False
```

Flowchart



Python Code

```python
#function to check if codes and names or contact numbers are the same for suppliers,hospitals and PPE items
def hasDuplicates(li):
    seen = set()
    for i in li:
        if i in seen:
            print("Duplicates are not allowed")
            return True
        seen.add(i)

    return False
```

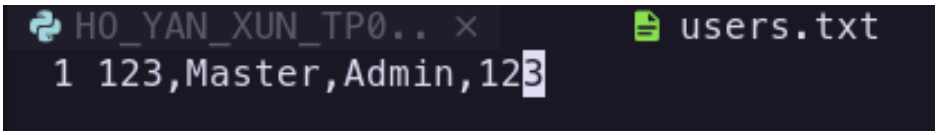
Code Execution & Outcome

1. Init Check / Main Menu

When first executing the program, initCheck() is run to check whether the program is being run for the first time, if the answer is yes, it will prompt the user to enter a user id, user name and password for the creation of the master account.

```
❯ python HO_YAN_XUN_TP073669_YAP_ZHU_SHENG_TP073670.py
Initializing system...

Entering initialization, please enter the userID and password for creating an
admin account
Please enter your userID: 123
Please enter your name: Master
Please enter your password: 123
```

After the user enters the details, users.txt is created and the first account is created.



Next is the creation of the suppliers:

```
Initializing suppliers.txt...

Do you have 3 or 4 suppliers: 3

Please enter the details of 3 suppliers only.
Example: AA,BB,CC

Please enter the all the supplier code with comma in between: AA,BB,CC
Please enter the all the supplier name with comma in between: Alpha, Beta, Gam
ma
```

Next is the creation of hospitals:

```
Initializing system...

Do you have 3 or 4 hospitals: 3

Please enter the details of 3 hospitals only.
Example: AA,BB,CC

Please enter the all the hospital code with comma in between: KKM,KPJ,KBJ
Please enter the all the hospital name with comma in between: Klinik Kesihatan
 Malaysia,Klinik Petaling Jaya,Klinik Bukit Jalil
Initializing complete
Initialization complete.


Welcome to PPE Inventory Management System
Type "quit" to quit the program

Please enter your userID:
```

After all that, the user is presented with the login screen, here after filling in the login details, the user will see a beautiful main menu.

```
Welcome to PPE Inventory Management System
Type "quit" to quit the program

Please enter your userID: 123
Please enter your password: 123
 ______ __       __  ______
|      \  \     /  \/      \
 \▓▓▓▓▓▓ ▓▓\   /  ▓▓▓▓▓▓▓▓\
  | ▓▓ | ▓▓\ \/  ▓▓ | ▓▓▓▓\
  | ▓▓ | ▓▓ \  \ ▓▓ | ▓▓\▓▓\
  | ▓▓ | ▓▓  \▓▓ ▓▓_| ▓▓ \▓▓\
 _| ▓▓_| ▓▓   \▓▓▓|  \▓▓__| ▓▓
|  \ ▓▓ \ ▓▓    \▓▓\  \▓▓▓▓▓▓
 \▓▓▓▓▓▓\▓▓▓▓      \▓▓ \▓▓▓▓▓▓

Welcome Master

Welcome to the PPE Inventory Management System
1. Inventory
2. Suppliers
3. List Hospitals
4. User Management
5. Log Out
Select one:
```

There are many options for the user can choose from. User can enter different menus by typing the number of the option.

2. Inventory

When first entering the inventory, it detects that the inventory has not been created, thus entering the inventory initialization process.

```
Please enter the details of 6 PPE Items.
Example: AA,BB,CC,DD,EE,FF

Please enter the all the item code with comma in between: HC,FS,MS,GL,GW,SC
Please enter the all the item name with comma in between: Head Cover,Face Shield,Mask,Gloves,
Gown,Shoe Covers
Please enter the all the supplier code for each item with comma in between: AA,AA,BB,BB,CC,CC

Inventory
1. Check Stock
2. Receive Items
3. Distribute Items
4. Transaction History
5. Search Transaction Detail of an Item
6. Quit

Select one:
```

This process validates the input before proceeding to creating ppe.txt and record the initial stock input inside trasactions.txt.

HO_YAN_XUN_TP0.. × users.txt × suppliers.txt × transaction.txt ×

```
2023-10-28 18:15:04.685687,Head Cover,HC,100,,received
2023-10-28 18:16:40.902316,Head Cover,HC,100,AA,received
2023-10-28 18:16:40.902370,Face Shield,FS,100,AA,received
2023-10-28 18:16:40.902380,Mask,MS,100,BB,received
2023-10-28 18:16:40.902388,Gloves,GL,100,BB,received
2023-10-28 18:16:40.902396,Gown,GW,100,CC,received
2023-10-28 18:16:40.902404,Shoe Covers,SC,100,CC,received
```

Initial Stock Input is recorded

From the inventory menu, the user can perform various actions such as stock checking, receiving and distributing items, view the transaction and distribution history, and search transactions between a time period.

a. Check Stock

```
Inventory
1. Check Stock
2. Receive Items
3. Distribute Items
4. Transaction History
5. Search Transaction Detail of an Item
6. Quit

Select one: 1

Item Code      Item Name       Item Quantity
FS            Face Shield           100
GL               Gloves             100
GW                Gown              100
HC             Head Cover           100
MS                Mask              100
SC            Shoe Covers           100
```

Displays the stock in a readable and eye appealing format.

b. Receive Items

Select the item user want to receive with item code, then the quantity. Records the transaction in transaction.txt

```
Select one: 2

Item Code      Item Name       Item Quantity
FS            Face Shield           100
GL               Gloves             100
GW                Gown              100
HC             Head Cover           100
MS                Mask              100
SC            Shoe Covers           100

Select the item receiving(Item Code, Type "Quit" to quit):FS
Input the amount received:69

Select the item receiving(Item Code, Type "Quit" to quit):Quit

Inventory
1. Check Stock
2. Receive Items
3. Distribute Items
4. Transaction History
5. Search Transaction Detail of an Item
6. Quit

Select one: 1

Item Code      Item Name       Item Quantity
FS            Face Shield           169
GL               Gloves             100
GW                Gown              100
HC             Head Cover           100
MS                Mask              100
SC            Shoe Covers           100
```

c. Distribute Items

Distribute items by selecting the item code, target hospital and the quantity. Distributions will be recorded in distributions.txt.

```
Select one: 3

Item Code      Item Name       Item Quantity
FS            Face Shield           169
GL               Gloves             100
GW                Gown              100
HC             Head Cover           100
MS                Mask              100
SC            Shoe Covers           100

Select the item distributing(Item Code, Type "Quit" to quit):GL

Hospital Code              Hospital Name
KBJ                      Klinik Bukit Jalil
KKM                  Klinik Kesihatan Malaysia
KPJ                     Klinik Petaling Jaya

Select the hospital distributing to(Hospital Code):KBJ
Input the amount distributed:80

Select the item distributing(Item Code, Type "Quit" to quit):Quit

Inventory
1. Check Stock
2. Receive Items
3. Distribute Items
4. Transaction History
5. Search Transaction Detail of an Item
6. Quit
Reminder: Items less than 25 boxes
Gloves
Select one: 1

Item Code      Item Name       Item Quantity
FS            Face Shield           169
GL               Gloves             20
GW                Gown              100
HC             Head Cover           100
MS                Mask              100
SC            Shoe Covers           100
```

Note the reminder on the bottom that reminds the user which item has less than 25 boxes.

d. Transaction History

The user can choose to view the full transaction history, where the program will display it in a nicely manner:

```
Welcome to history
1. Transaction Full History
2. Distribution History
3. Transaction Between a Time Period

Please select one(Type "Quit" to quit): 1

Transaction Time                 Item Name          Item Code       Item Quantity       Supplier or Hospital Code         Status
2023-10-28 18:16:40.902316        Head Cover             HC              100                        AA                  received
2023-10-28 18:16:40.902370       Face Shield             FS              100                        AA                  received
2023-10-28 18:16:40.902380           Mask                MS              100                        BB                  received
2023-10-28 18:16:40.902388          Gloves               GL              100                        BB                  received
2023-10-28 18:16:40.902396           Gown                GW              100                        CC                  received
2023-10-28 18:16:40.902404       Shoe Covers             SC              100                        CC                  received
2023-10-28 18:29:53.128115       Face Shield             FS              69                         AA                  received
2023-10-28 18:33:10.962661          Gloves               GL              80                        KBJ                 distributed
```

The user can choose to view only the distributions:

```
Welcome to history
1. Transaction Full History
2. Distribution History
3. Transaction Between a Time Period

Please select one(Type "Quit" to quit): 2

Distribution Time                Item Name          Item Code       Item Quantity             Hospital Code
2023-10-28 18:33:10.962765          Gloves               GL              80                        KBJ
```

The user can also only view transactions between a time period (data manually edited for demonstration purposes):

Full history

```
Welcome to history
1. Transaction Full History
2. Distribution History
3. Transaction Between a Time Period

Please select one(Type "Quit" to quit): 1

Transaction Time                 Item Name          Item Code       Item Quantity       Supplier or Hospital Code         Status
2020-10-28 18:16:40.902316        Head Cover             HC              100                        AA                  received
2021-10-28 18:16:40.902370       Face Shield             FS              100                        AA                  received
2022-10-28 18:16:40.902380           Mask                MS              100                        BB                  received
2023-10-28 18:16:40.902388          Gloves               GL              100                        BB                  received
2024-10-28 18:16:40.902396           Gown                GW              100                        CC                  received
2025-10-28 18:16:40.902404       Shoe Covers             SC              100                        CC                  received
2026-10-28 18:29:53.128115       Face Shield             FS              69                         AA                  received
2027-10-28 18:33:10.962661          Gloves               GL              80                        KBJ                 distributed
2028-10-28 18:33:10.962661          Gloves               GL              80                        KBJ                 distributed
2029-10-28 18:16:40.902396           Gown                GW              100                        CC                  received
```

History between time

```
Welcome to history
1. Transaction Full History
2. Distribution History
3. Transaction Between a Time Period

Please select one(Type "Quit" to quit): 3

Search for transactions during a time period
*leave blank for default*
Please input the starting date(dd/mm/yyyy): 01/01/2023
Please input the ending date(dd/mm/yyyy): 01/01/2027

     Transaction Date          Item Name          Item Code           Quantity      Supplier Or Hospital Code       Status
2023-10-28 18:16:40.902388       Gloves              GL                100                    BB                 received
2024-10-28 18:16:40.902396        Gown               GW                100                    CC                 received
2025-10-28 18:16:40.902404    Shoe Covers            SC                100                    CC                 received
2026-10-28 18:29:53.128115    Face Shield            FS                 69                    AA                 received
```

e. Transaction details of a certain item

```
Inventory
1. Check Stock
2. Receive Items
3. Distribute Items
4. Transaction History
5. Search Transaction Detail of an Item
6. Quit
Reminder: Items less than 25 boxes
Gloves
Select one: 5

Item Code      Item Name       Item Quantity
FS            Face Shield           169
GL               Gloves             20
GW                Gown              100
HC             Head Cover           100
MS                Mask              100
SC            Shoe Covers           100
Please enter the item code(Type "Quit" to quit): FS

Transaction Time                  Item Name          Item Code       Item Quantity       Supplier or Hospital Code          Status
2021-10-28 18:16:40.902370       Face Shield            FS              100                      AA                   received
2026-10-28 18:29:53.128115       Face Shield            FS              69                       AA                   received
```

3. Suppliers

Users can list supplier details or change their name and contact number. Below are some examples.

```
Welcome to the PPE Inventory Management System
1. Inventory
2. Suppliers
3. List Hospitals
4. User Management
5. Log Out
Select one: 2

Welcome to supplier details
1. List Supplier Details
2. Change Supplier Name
3. Change Supplier Contact Number
4. Quit
Select one:
```

a. List Supplier Details

```
Welcome to supplier details
1. List Supplier Details
2. Change Supplier Name
3. Change Supplier Contact Number
4. Quit
Select one: 1

Supplier Code        Supplier Name        Supplier Contact
AA                       Alpha                  123
BB                       Beta                   456
CC                       Gamma                  789
```

b. Change Supplier Name

```
Welcome to supplier details
1. List Supplier Details
2. Change Supplier Name
3. Change Supplier Contact Number
4. Quit
Select one: 2

Supplier Code        Supplier Name        Supplier Contact
AA                       Alpha                  123
BB                       Beta                   456
CC                       Gamma                  789
Select the supplier you want to change(Supplier Code, Type "Quit" to quit):
AA
Enter the new name:Cita
New name changed
Select the supplier you want to change(Supplier Code, Type "Quit" to quit):
Quit

Welcome to supplier details
1. List Supplier Details
2. Change Supplier Name
3. Change Supplier Contact Number
4. Quit
Select one: 1

Supplier Code        Supplier Name        Supplier Contact
AA                       Cita                   123
BB                       Beta                   456
CC                       Gamma                  789
```

c. Change Supplier Contact Number

```
Welcome to supplier details
1. List Supplier Details
2. Change Supplier Name
3. Change Supplier Contact Number
4. Quit
Select one: 3

Supplier Code        Supplier Name        Supplier Contact
AA                       Cita                 123
BB                       Beta                 456
CC                       Gamma                789
Select the supplier you want to change(Supplier Code, Type "Quit" to quit):
AA
Enter the new contact number:6969
Contact number changed
Select the supplier you want to change(Supplier Code, Type "Quit" to quit):
Quit

Welcome to supplier details
1. List Supplier Details
2. Change Supplier Name
3. Change Supplier Contact Number
4. Quit
Select one: 1

Supplier Code        Supplier Name        Supplier Contact
AA                       Cita                 6969
BB                       Beta                 456
CC                       Gamma                789
```

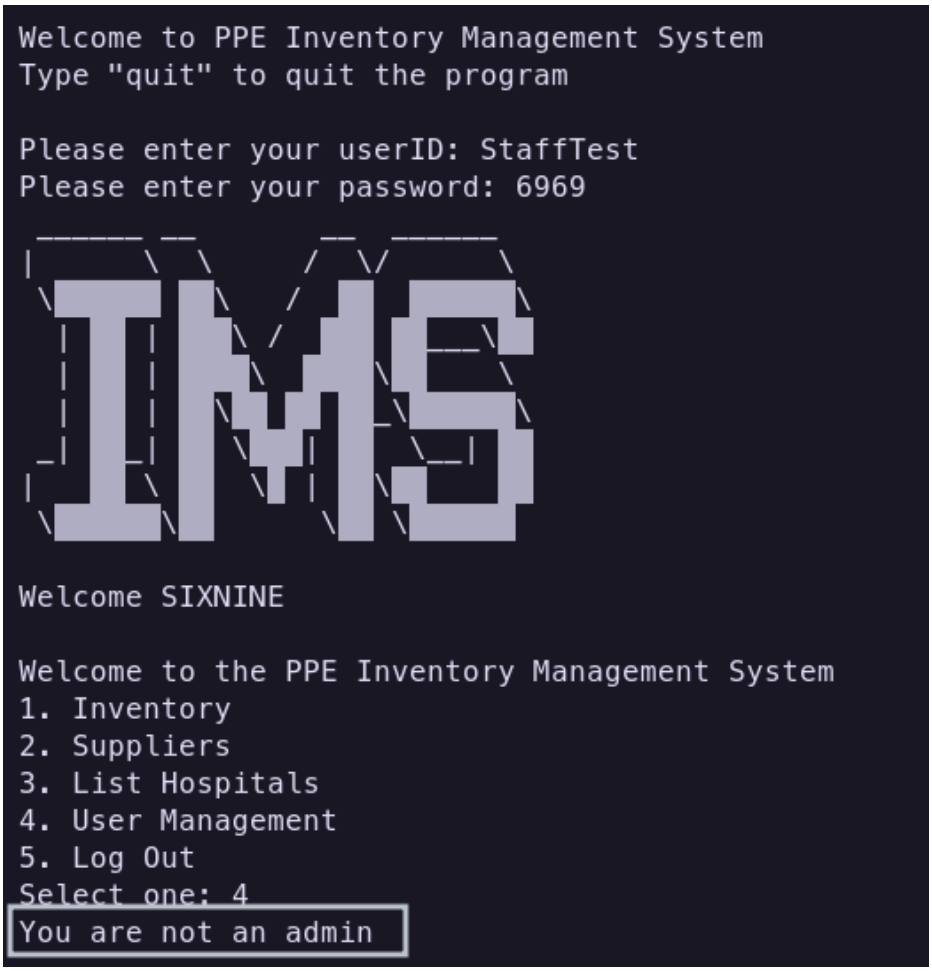
4. List Hospitals

List out hospitals.

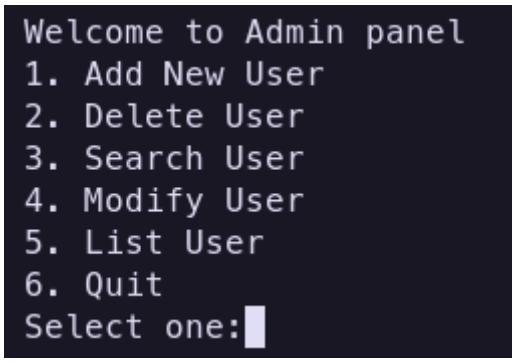
```
Welcome to the PPE Inventory Management System
1. Inventory
2. Suppliers
3. List Hospitals
4. User Management
5. Log Out
Select one: 3

Hospital Code                Hospital Name
KBJ                       Klinik Bukit Jalil
KKM                   Klinik Kesihatan Malaysia
KPJ                      Klinik Petaling Jaya
```

5. User Management

Only if the users that are Admin can access this panel. Access using a normal staff account will be denied.



Once the user is using an Admin account, the user management page should look like this.



a. Add New User

First the user will be asked to select whether to add an Admin or Staff, then the user will need to enter the user id, username and password.

```
Welcome to Admin panel
1. Add New User
2. Delete User
3. Search User
4. Modify User
5. List User
6. Quit
Select one: 1

User Type
1. Admin
2. Staff
3. Quit
Select one: 1
Please enter your userID: 6969
Please enter your name: IDK
Please enter your password: 6969
```

b. Delete User

The program will list out all the users. The user just has to select the one to be deleted. Note that the program will not allow the deletion of the user itself nor the first account (master account)

```
Welcome to Admin panel
1. Add New User
2. Delete User
3. Search User
4. Modify User
5. List User
6. Quit
Select one: 2

Select the user you want to delete(Type "Quit" to quit):
No.      User ID       User Name      User Type
1          123          Master          Admin
2       StaffTest       SIXNINE         Staff
3         6969            IDK           Admin
3
User deleted
```

c. Search User

Simple search function using user code.

```
Welcome to Admin panel
1. Add New User
2. Delete User
3. Search User
4. Modify User
5. List User
6. Quit
Select one: 3

Search user by their userID(Type "Quit" to quit):StaffTest
Found user: StaffTest
Name: SIXNINE
Type: Staff

Search user by their userID(Type "Quit" to quit):fjdkslfjdska
User code fjdkslfjdska not found
```

d. Modify User

Admins can modify the user type and password of the other accounts. All accounts' user type can be changed except for the master account.

```
Welcome to Admin panel
1. Add New User
2. Delete User
3. Search User
4. Modify User
5. List User
6. Quit
Select one: 4

Select the user you want to modify(Type "Quit" to quit):
No.      User ID       User Name      User Type
1          123          Master          Admin
2       StaffTest       SIXNINE         Staff
2

Select the action to perform(Type "Quit" to quit):
1. Change user type
2. Change password
1

Select one(Admin, Staff ; Type "Quit" to quit):
Admin

Select the action to perform(Type "Quit" to quit):
1. Change user type
2. Change password
2

Type the old password:6969
Type the new password:ilove69
```

When demoting the current account's user type from Admin to Staff, the user will be redirected back to the main menu and loses access to the user management page.

```
Select the user you want to modify(Type "Quit" to quit):
No.      User ID       User Name      User Type
1          123          Master          Admin
2       StaffTest       SIXNINE         Admin
2

Select the action to perform(Type "Quit" to quit):
1. Change user type
2. Change password
1

Select one(Admin, Staff ; Type "Quit" to quit):
Staff
You are changing yourself from Admin to Staff, proceed?(Yes/No): Yes

Welcome to the PPE Inventory Management System
1. Inventory
2. Suppliers
3. List Hospitals
4. User Management
5. Log Out
Select one: 4
You are not an admin
```

e. List User

Very straightforward, this function lists all the users.

```
Welcome to Admin panel
1. Add New User
2. Delete User
3. Search User
4. Modify User
5. List User
6. Quit
Select one: 5
No.      User ID       User Name      User Type
1          123          Master          Admin
2       StaffTest       SIXNINE         Staff
```

Conclusion

The Inventory Management System for PPE is a python-based program that runs in the terminal. It is designed to manage PPE stock, suppliers, hospitals and user accounts. The program also comes with different functions the user can utilize such as receiving, distributing PPE stocks, tracking and viewing transaction records, adding, deleting, searching and modifying user accounts, managing suppliers and hospitals etc. When viewing transactions or other data, they are presented in a nice readable format, allowing for easy auditing. The data stored can also be easily parsed to other programming languages, allowing for further functionality expansion.

In conclusion, this is a great piece of software, it is packed with features, and it comes with input validation and error handling that prompts accurate and informative error messages whenever an input is invalid.

End of Report