GROUP ASSIGNMENT PART 2 (CODING IMPLEMENTATION)

INTAKE CODE: UCDF2307ICT(SE)
MODULE CODE: AAPP015-4-1-PWP
MODULE NAME: PROGRAMMING WITH PYTHON
GROUP TUTORIAL: LAB-13 / LAB-14
GROUP: GROUP 8
SUBMISSION DATE: 4 AUGUST 2024
LECTURER NAME: DR. MASRINA AKMAL BINTI SALLEH

GROUP MEMBERS NAME	MATRIX NO
EJJAZ HAKIMI BIN MOHAMAD AZAN	TP073318
ELIANNA CATRINA HERRERA	TP073631
LAKESH A/L MANIMARAN	TP075855
SUCHITRA NAMBIAR A/P MAHANDRAN	TP074762
NG VIN EE	TP073088

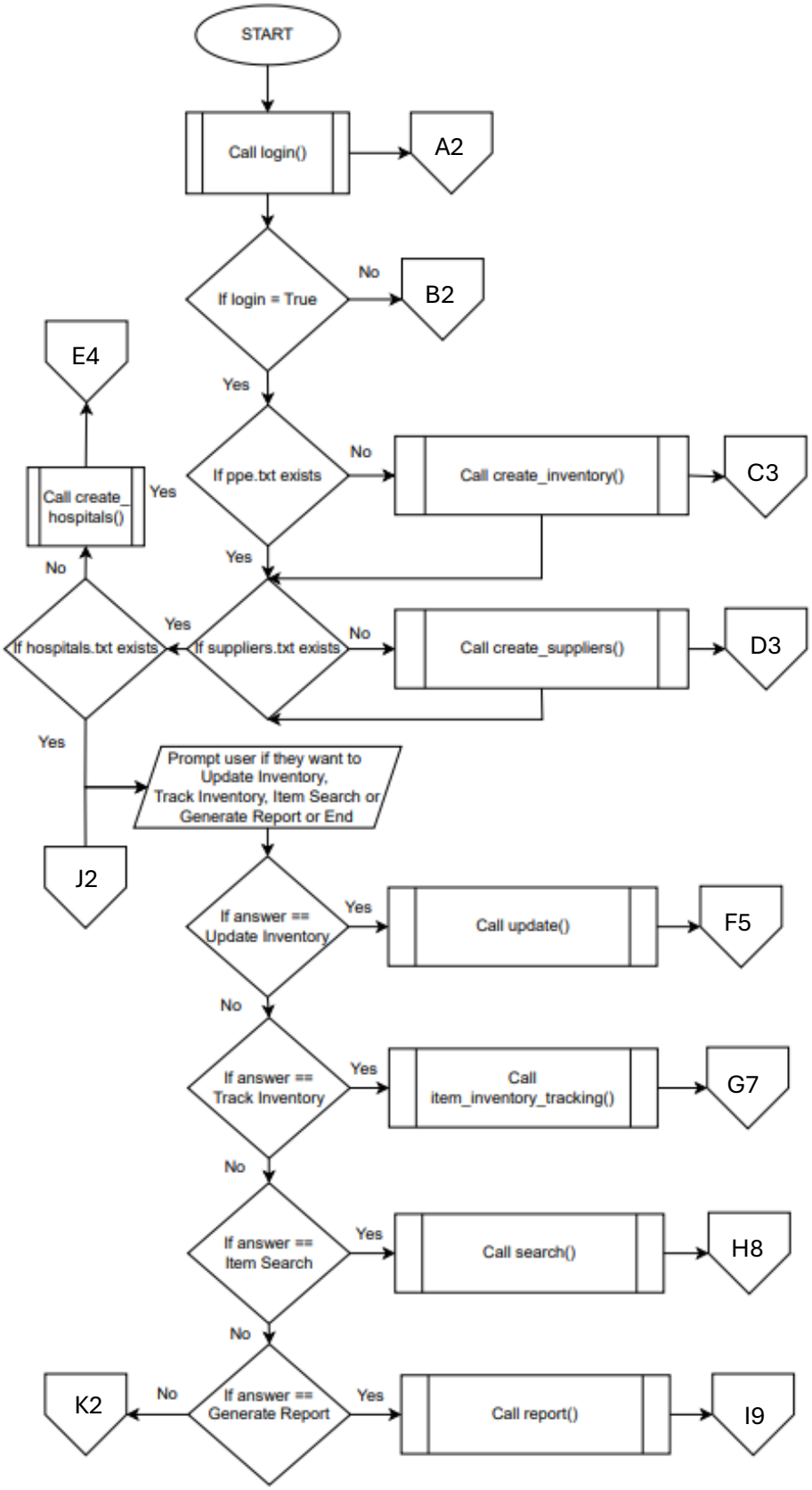
Table of Contents

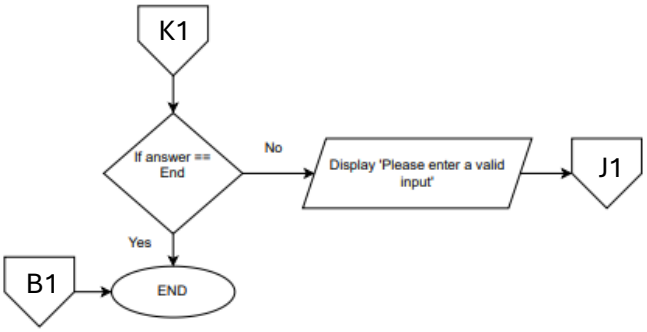
Workload Matrix

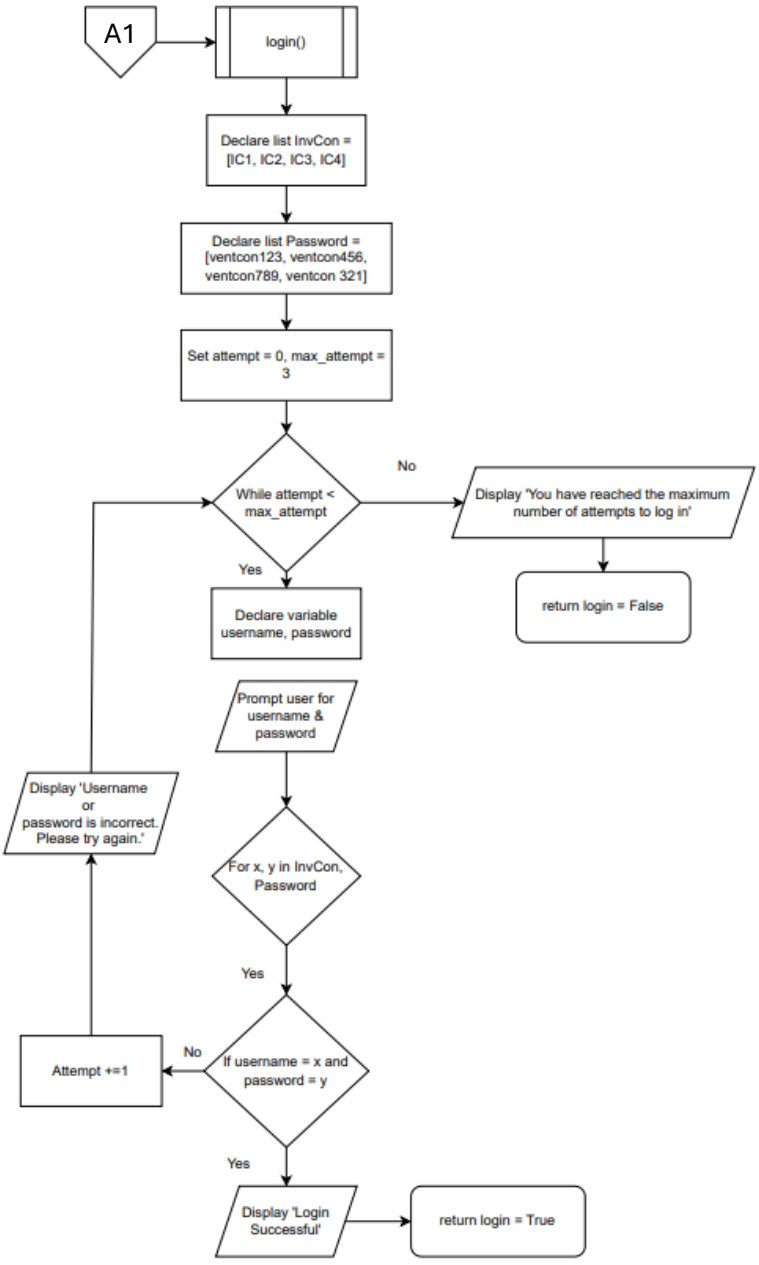
	Task Distribution	Percentage per member (%)					Total Percentage	Signature				
		S1	**S2**	**S3**	**S4**	**S5**						
1	Python Script	25	25	10	20	20	100%					
2	Pseudocode	0	20	20	30	30	100%					
3	Flow Chart	100	0	0	0	0	100%					
4	Testing	20	20	20	20	20	100%					
5	Documentation	30	20	10	20	20	100%					
6	Presentation	20	20	20	20	20	100%					

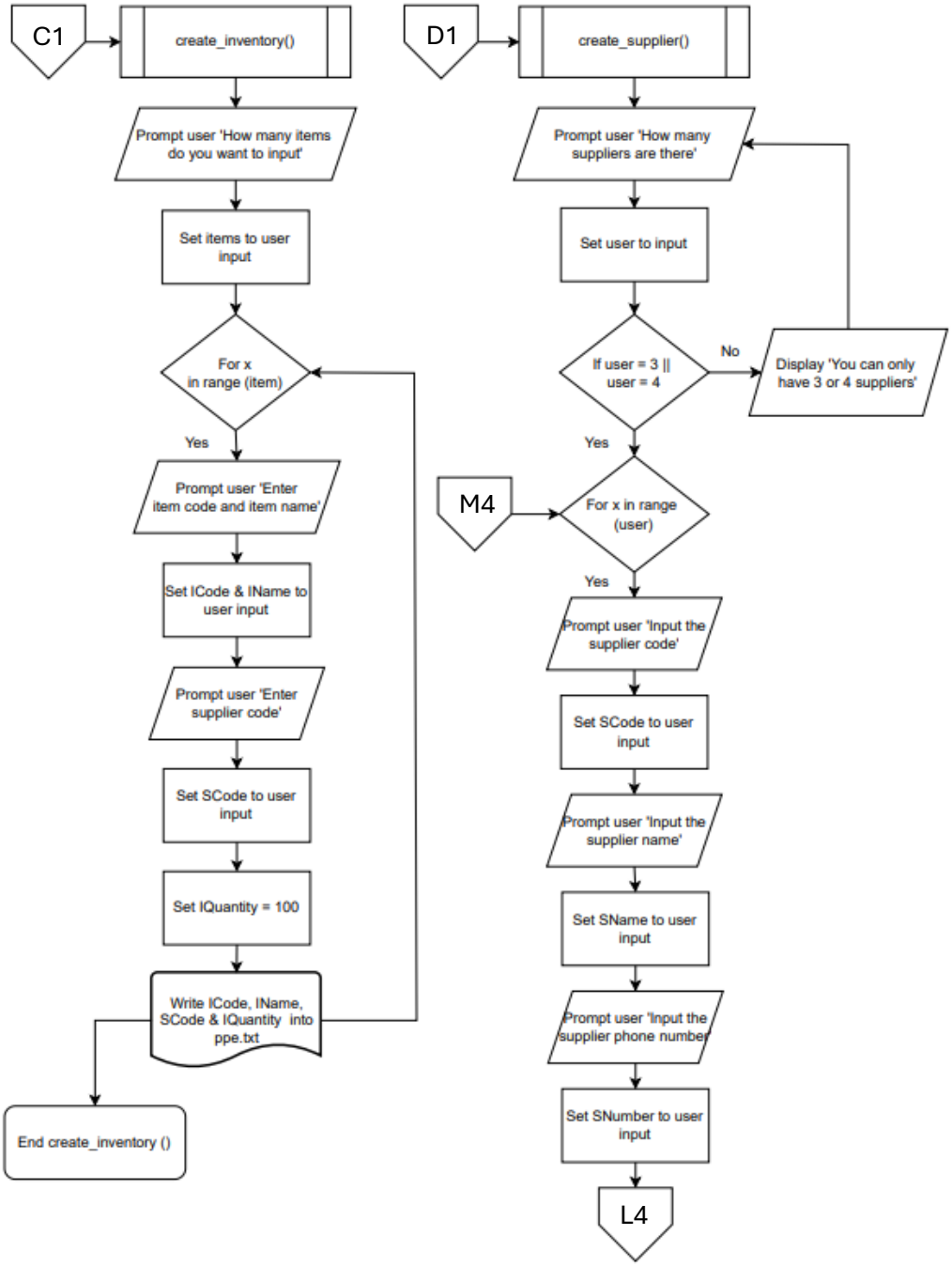
S1	EJJAZ HAKIMI BIN MOHAMAD AZAN
S2	ELIANNA CATRINA HERRERA
S3	LAKESH A/L MANIMARAN
S4	SUCHITRA NAMBIAR A/P MAHANDRAN
S5	NG VIN EE

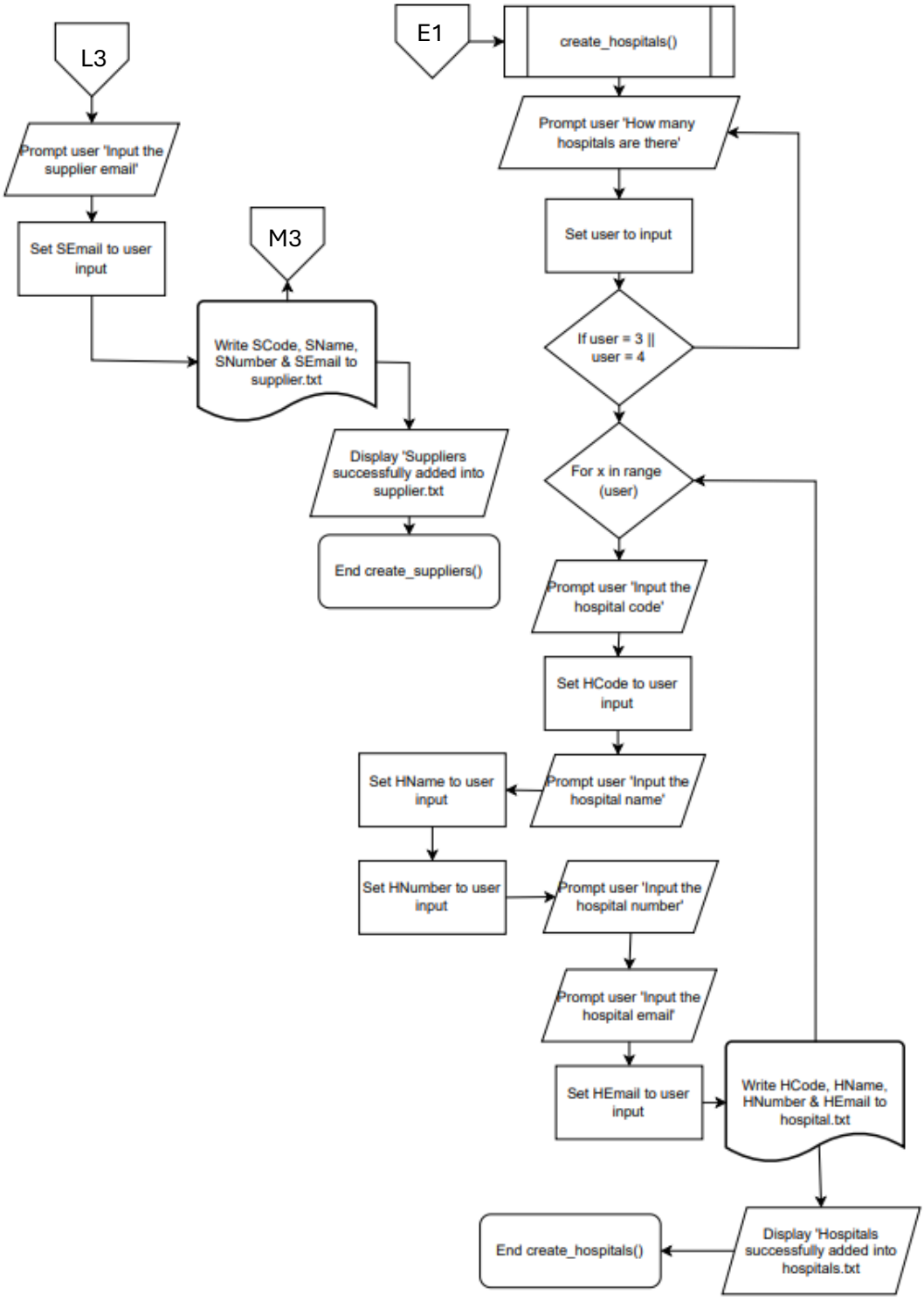
Updated Flow Chart

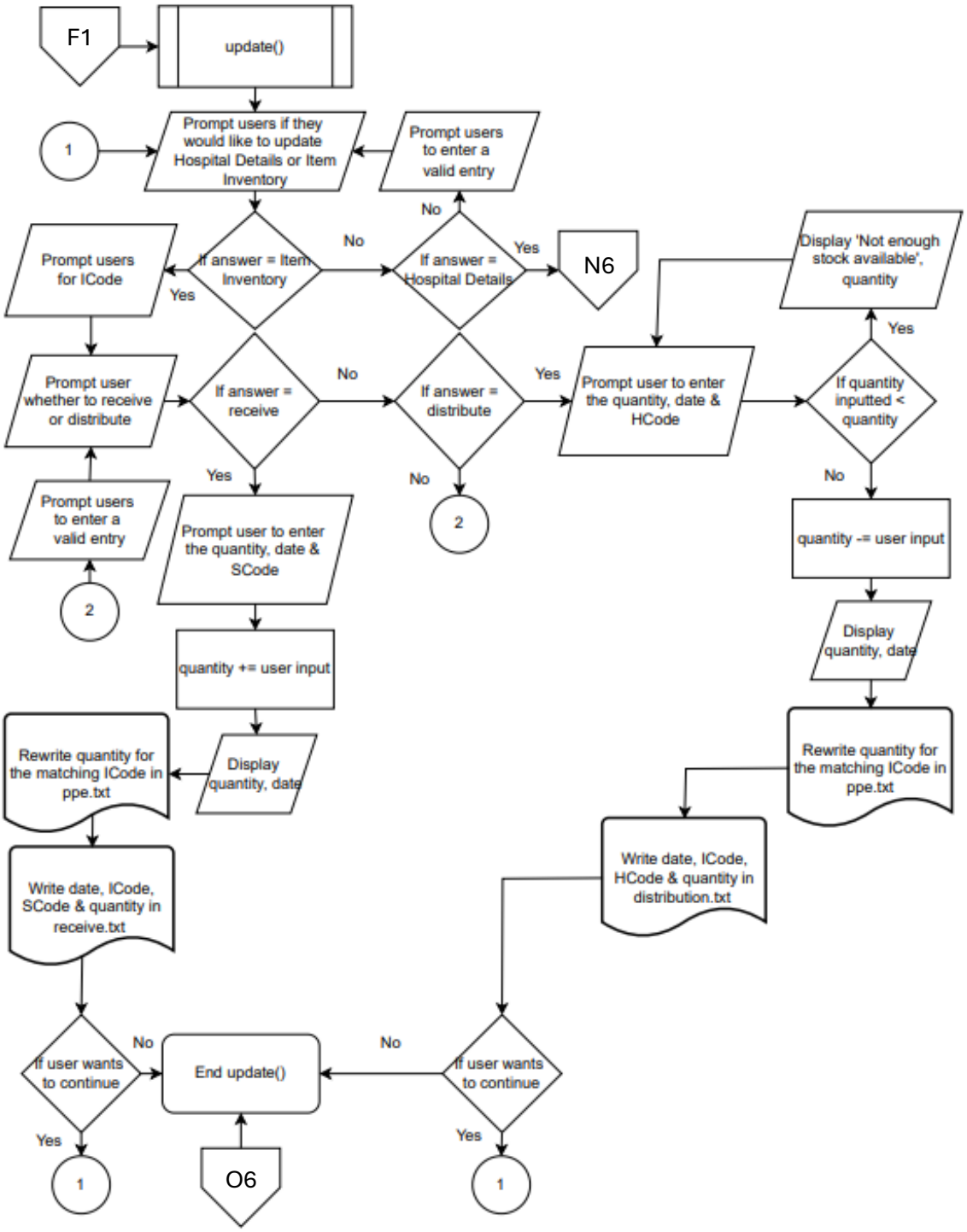


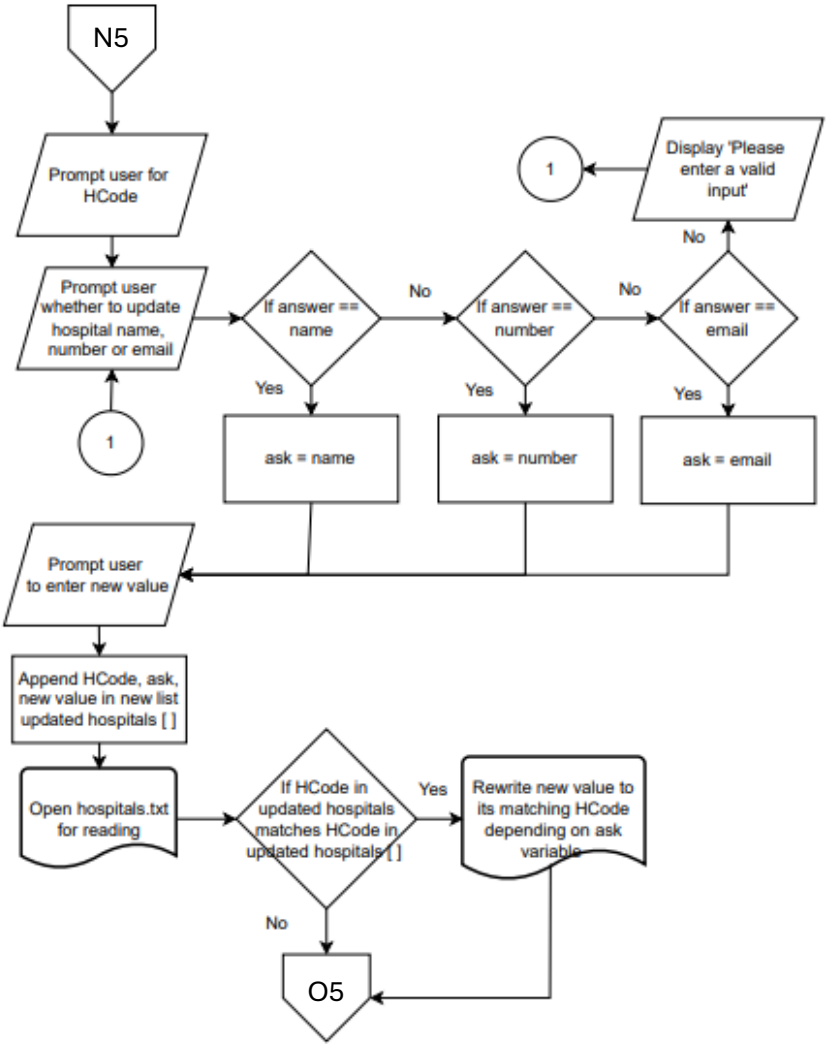

K1
If answer == End
No
Display 'Please enter a valid input'
J1
Yes
B1
END

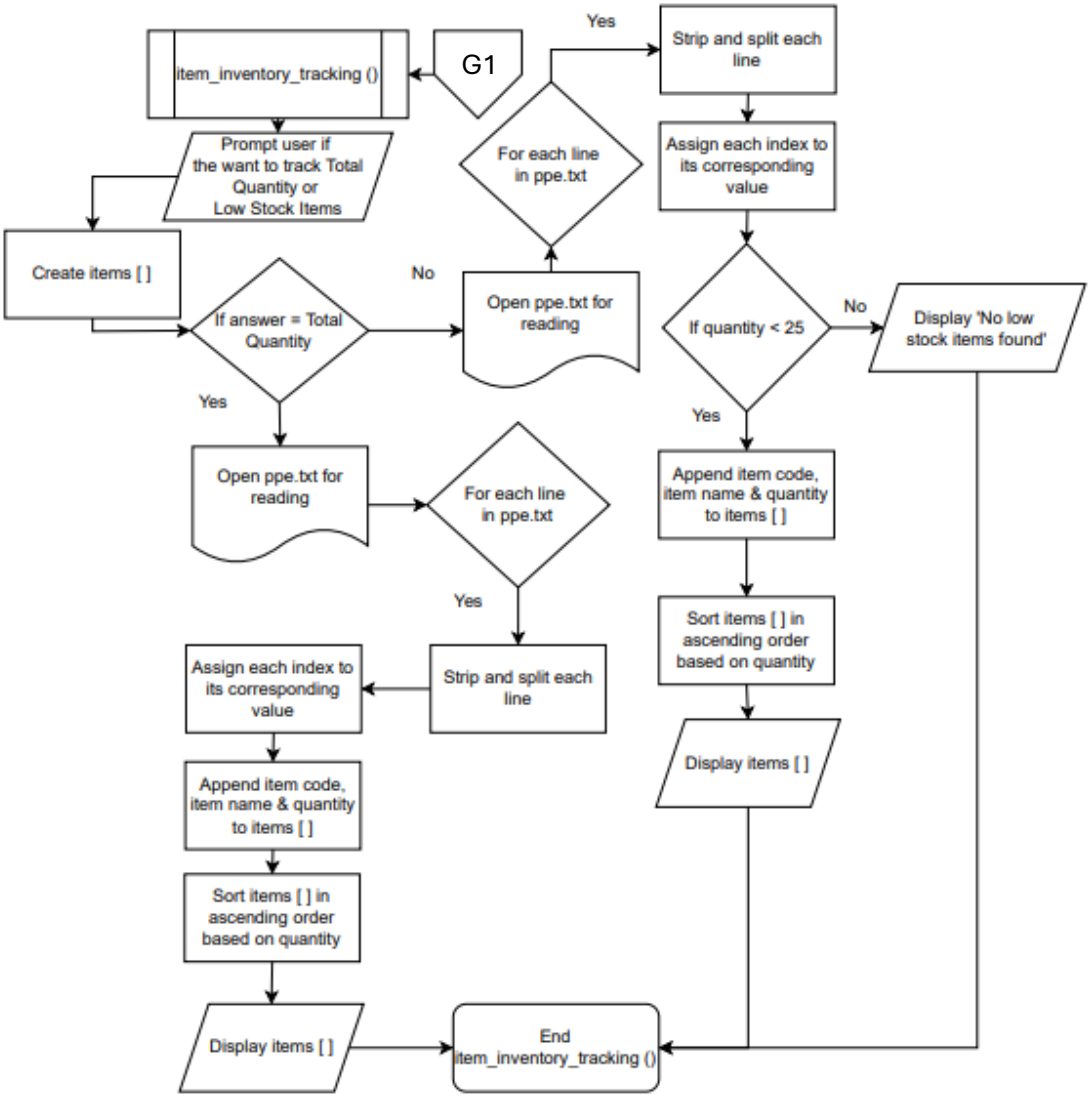

A1
login()
Declare list InvCon = [IC1, IC2, IC3, IC4]
Declare list Password = [ventcon123, ventcon456, ventcon789, ventcon 321]
Set attempt = 0, max_attempt = 3
While attempt < max_attempt
No
Display 'You have reached the maximum number of attempts to log in'
return login = False
Yes
Declare variable username, password
Prompt user for username & password
Display 'Username or password is incorrect. Please try again.'
For x, y in InvCon, Password
Yes
Attempt +=1
No
If username = x and password = y
Yes
Display 'Login Successful'
return login = True

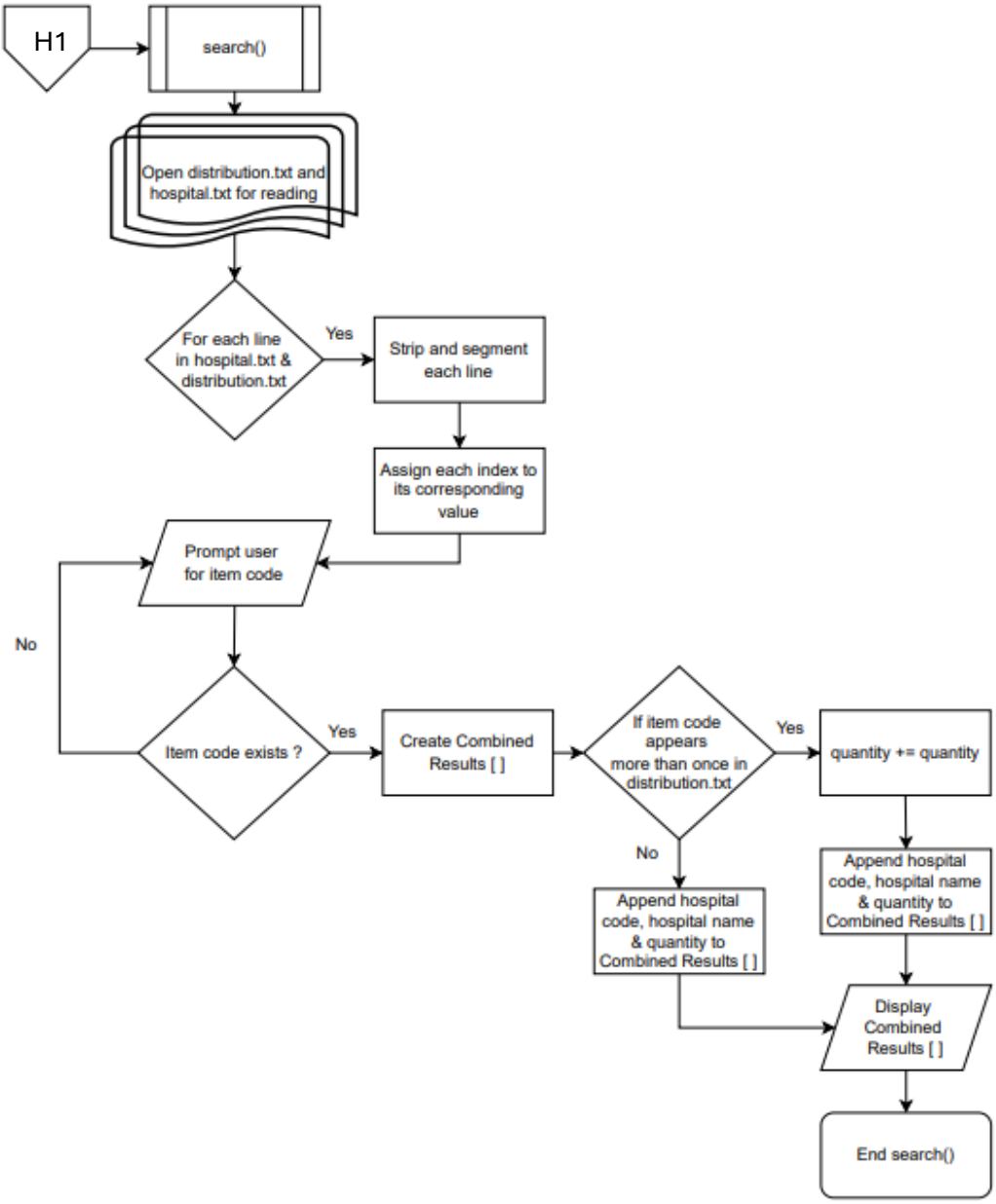

C1
create_inventory()
Prompt user 'How many items do you want to input'
Set items to user input
For x in range (item)
Yes
Prompt user 'Enter item code and item name'
Set ICode & IName to user input
Prompt user 'Enter supplier code'
Set SCode to user input
Set IQuantity = 100
Write ICode, IName, SCode & IQuantity into ppe.txt
End create_inventory ()
D1
create_supplier()
Prompt user 'How many suppliers are there'
Set user to input
If user = 3 || user = 4
No
Display 'You can only have 3 or 4 suppliers'
Yes
M4
For x in range (user)
Yes
Prompt user 'Input the supplier code'
Set SCode to user input
Prompt user 'Input the supplier name'
Set SName to user input
Prompt user 'Input the supplier phone number'
Set SNumber to user input
L4

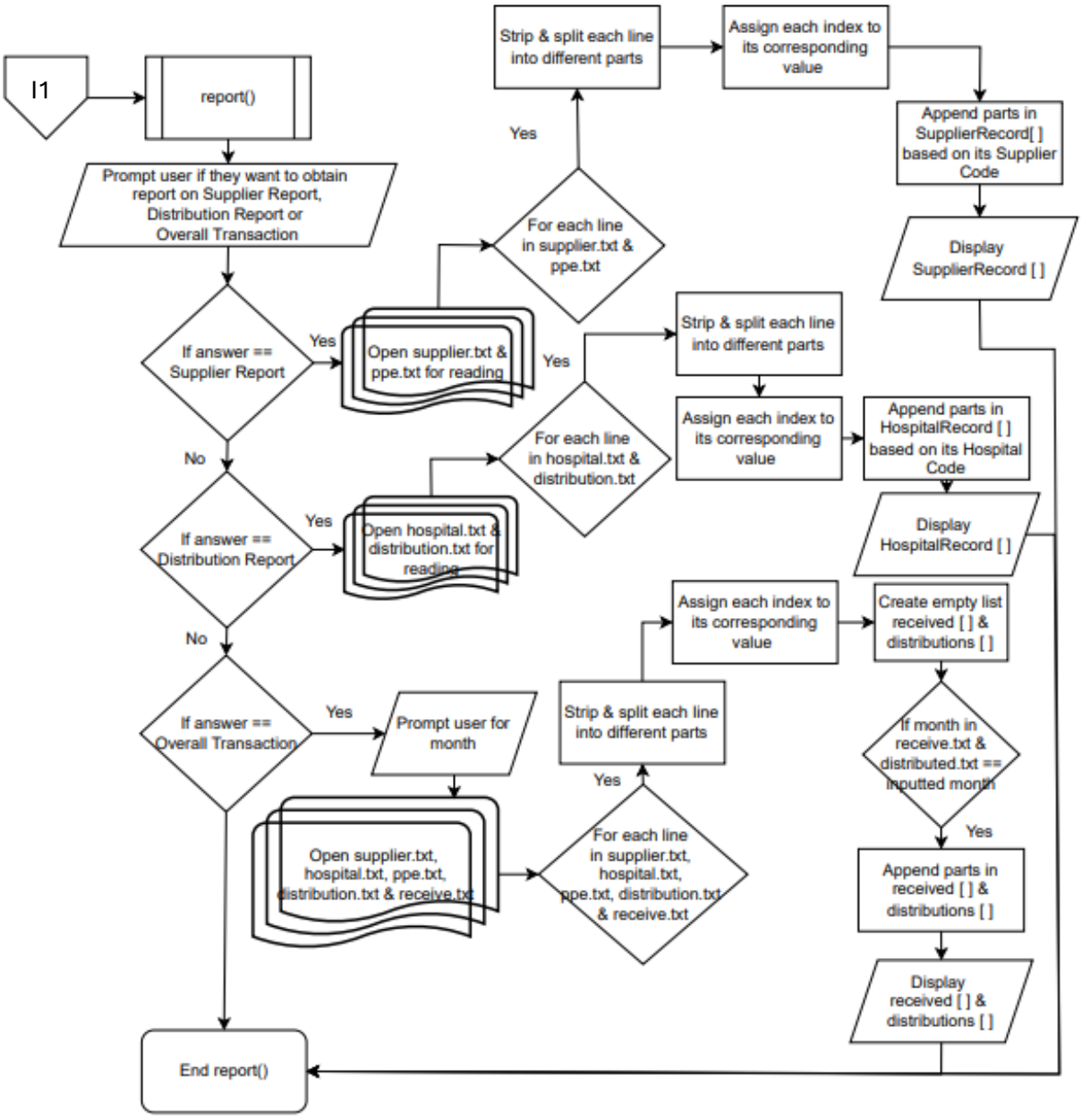

L3
Prompt user 'Input the supplier email'
Set SEmail to user input
M3
Write SCode, SName, SNumber & SEmail to supplier.txt
Display 'Suppliers successfully added into supplier.txt
End create_suppliers()
E1
create_hospitals()
Prompt user 'How many hospitals are there'
Set user to input
If user = 3 || user = 4
For x in range (user)
Prompt user 'Input the hospital code'
Set HCode to user input
Prompt user 'Input the hospital name'
Set HName to user input
Prompt user 'Input the hospital number'
Set HNumber to user input
Prompt user 'Input the hospital email'
Set HEmail to user input
Write HCode, HName, HNumber & HEmail to hospital.txt
Display 'Hospitals successfully added into hospitals.txt
End create_hospitals()


F1
update()
1
Prompt users if they would like to update Hospital Details or Item Inventory
Prompt users to enter a valid entry
No
If answer = Item Inventory
No
If answer = Hospital Details
Yes
N6
Prompt users for ICode
Yes
Prompt user whether to receive or distribute
If answer = receive
No
If answer = distribute
Yes
Prompt user to enter the quantity, date & HCode
Display 'Not enough stock available', quantity
Yes
If quantity inputted < quantity
Prompt users to enter a valid entry
2
Yes
Prompt user to enter the quantity, date & SCode
No
2
No
quantity -= user input
quantity += user input
Display quantity, date
Rewrite quantity for the matching ICode in ppe.txt
Display quantity, date
Rewrite quantity for the matching ICode in ppe.txt
Write date, ICode, SCode & quantity in receive.txt
Write date, ICode, HCode & quantity in distribution.txt
If user wants to continue
No
End update()
No
If user wants to continue
Yes
1
O6
Yes
1


N5
Prompt user for HCode
Prompt user whether to update hospital name, number or email
1
If answer == name
No
If answer == number
No
If answer == email
No
Display 'Please enter a valid input'
1
Yes
Yes
Yes
ask = name
ask = number
ask = email
Prompt user to enter new value
Append HCode, ask, new value in new list updated hospitals [ ]
Open hospitals.txt for reading
If HCode in updated hospitals matches HCode in updated hospitals [ ]
Yes
Rewrite new value to its matching HCode depending on ask variable
No
O5

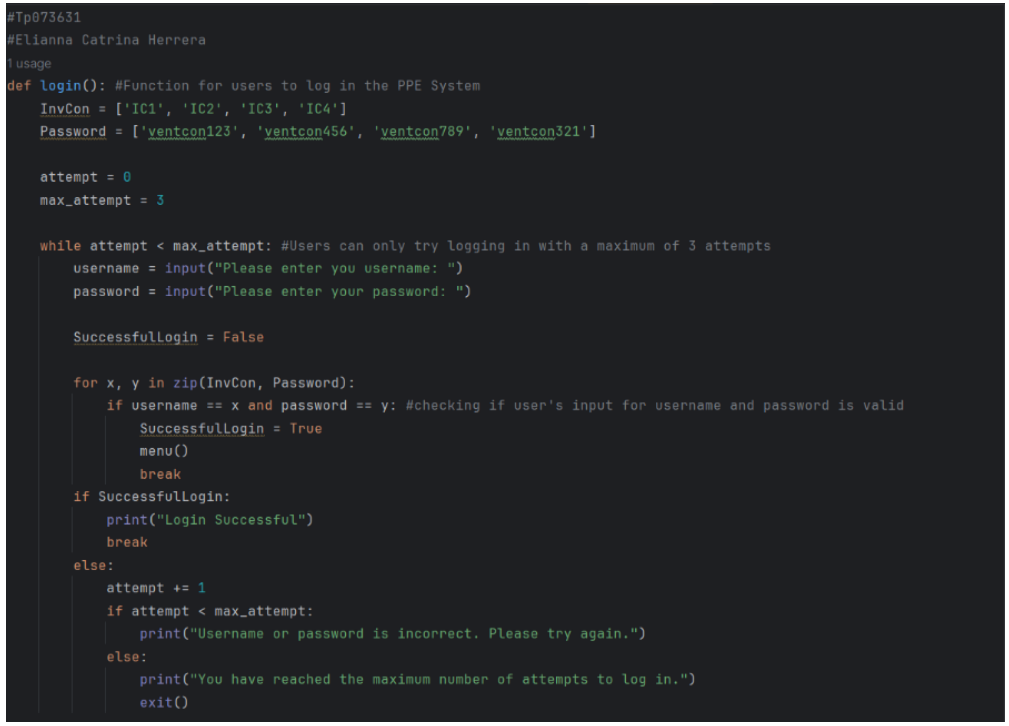

item_inventory_tracking ()
G1
Prompt user if the want to track Total Quantity or Low Stock Items
Create items [ ]
If answer = Total Quantity
No
Open ppe.txt for reading
For each line in ppe.txt
Yes
Strip and split each line
Assign each index to its corresponding value
If quantity < 25
No
Display 'No low stock items found'
Yes
Append item code, item name & quantity to items [ ]
Sort items [ ] in ascending order based on quantity
Display items [ ]
Yes
Open ppe.txt for reading
For each line in ppe.txt
Yes
Strip and split each line
Assign each index to its corresponding value
Append item code, item name & quantity to items [ ]
Sort items [ ] in ascending order based on quantity
Display items [ ]
End item_inventory_tracking ()


H1
search()
Open distribution.txt and hospital.txt for reading
For each line in hospital.txt & distribution.txt
Yes
Strip and segment each line
Assign each index to its corresponding value
Prompt user for item code
No
Item code exists ?
Yes
Create Combined Results [ ]
If item code appears more than once in distribution.txt
Yes
quantity += quantity
Append hospital code, hospital name & quantity to Combined Results [ ]
No
Append hospital code, hospital name & quantity to Combined Results [ ]
Display Combined Results [ ]
End search()

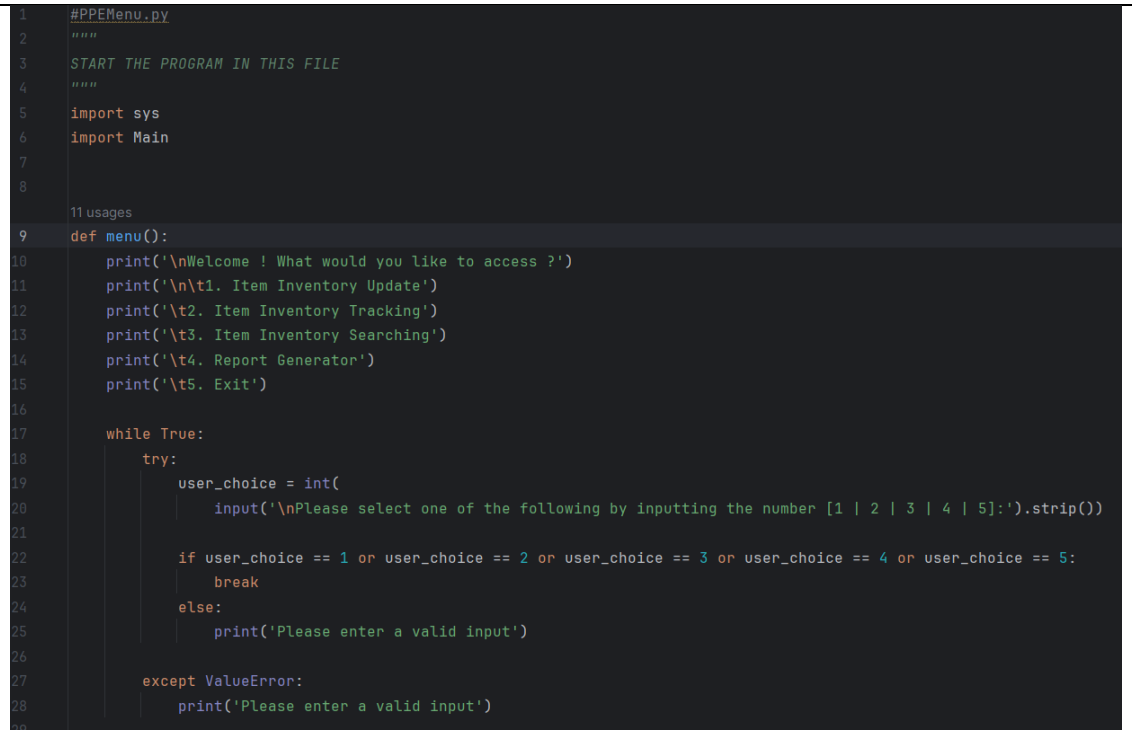
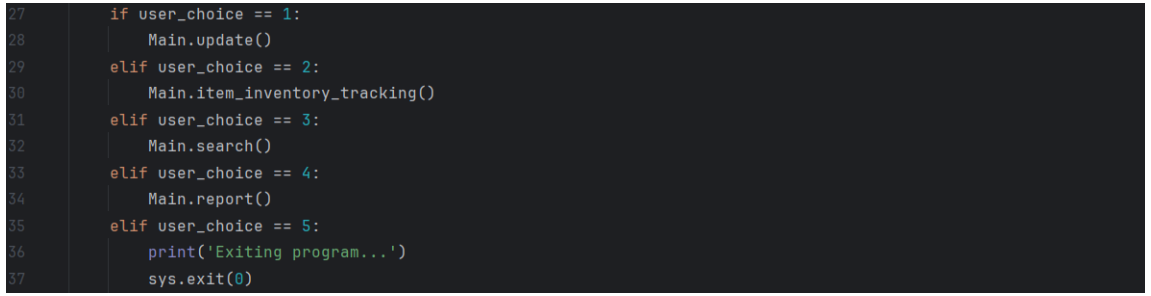

I1
report()
Prompt user if they want to obtain report on Supplier Report, Distribution Report or Overall Transaction
If answer == Supplier Report
Yes
Open supplier.txt & ppe.txt for reading
For each line in supplier.txt & ppe.txt
Yes
Strip & split each line into different parts
Assign each index to its corresponding value
Append parts in SupplierRecord[ ] based on its Supplier Code
Display SupplierRecord [ ]
No
If answer == Distribution Report
Yes
Open hospital.txt & distribution.txt for reading
For each line in hospital.txt & distribution.txt
Yes
Strip & split each line into different parts
Assign each index to its corresponding value
Append parts in HospitalRecord [ ] based on its Hospital Code
Display HospitalRecord [ ]
No
If answer == Overall Transaction
Yes
Prompt user for month
Open supplier.txt, hospital.txt, ppe.txt, distribution.txt & receive.txt
For each line in supplier.txt, hospital.txt, ppe.txt, distribution.txt & receive.txt
Yes
Strip & split each line into different parts
Assign each index to its corresponding value
Create empty list received [ ] & distributions [ ]
If month in receive.txt & distributed.txt == inputted month
Yes
Append parts in received [ ] & distributions [ ]
Display received [ ] & distributions [ ]
End report()

1.0 Program Source Code

Below are the source codes alongside with brief explanations. For the program to work, please ensure **both PPEMenu.py & Main.py** is downloaded. It is important to note that the program should **only be launched in the PPEMenu.py file** and **not** the **Main.py** file. A full setup guide can be found in **Appendix A**

Name	Source Code	Explanation
Login		This code begins by prompting users to enter their Username & Password. Once done, the code will check if both the Username & Password entered are matching and valid. If true, the user is granted access; if not, users are given a maximum of 3 attempts before the program terminates

```
#Tp073631
#Elianna Catrina Herrera
1 usage
def login(): #Function for users to log in the PPE System
    InvCon = ['IC1', 'IC2', 'IC3', 'IC4']
    Password = ['ventcon123', 'ventcon456', 'ventcon789', 'ventcon321']

    attempt = 0
    max_attempt = 3

    while attempt < max_attempt: #Users can only try logging in with a maximum of 3 attempts
        username = input("Please enter you username: ")
        password = input("Please enter your password: ")

        SuccessfulLogin = False

        for x, y in zip(InvCon, Password):
            if username == x and password == y: #checking if user's input for username and password is valid
                SuccessfulLogin = True
                menu()
                break
        if SuccessfulLogin:
            print("Login Successful")
            break
        else:
            attempt += 1
            if attempt < max_attempt:
                print("Username or password is incorrect. Please try again.")
            else:
                print("You have reached the maximum number of attempts to log in.")
                exit()
```

Menu	 	In the menu code users can select what function would they like to access by selecting the corresponding number

```python
#PPEMenu.py
"""
START THE PROGRAM IN THIS FILE
"""
import sys
import Main


11 usages
def menu():
    print('\nWelcome ! What would you like to access ?')
    print('\n\t1. Item Inventory Update')
    print('\t2. Item Inventory Tracking')
    print('\t3. Item Inventory Searching')
    print('\t4. Report Generator')
    print('\t5. Exit')

    while True:
        try:
            user_choice = int(
                input('\nPlease select one of the following by inputting the number [1 | 2 | 3 | 4 | 5]:').strip())

            if user_choice == 1 or user_choice == 2 or user_choice == 3 or user_choice == 4 or user_choice == 5:
                break
            else:
                print('Please enter a valid input')

        except ValueError:
            print('Please enter a valid input')
```

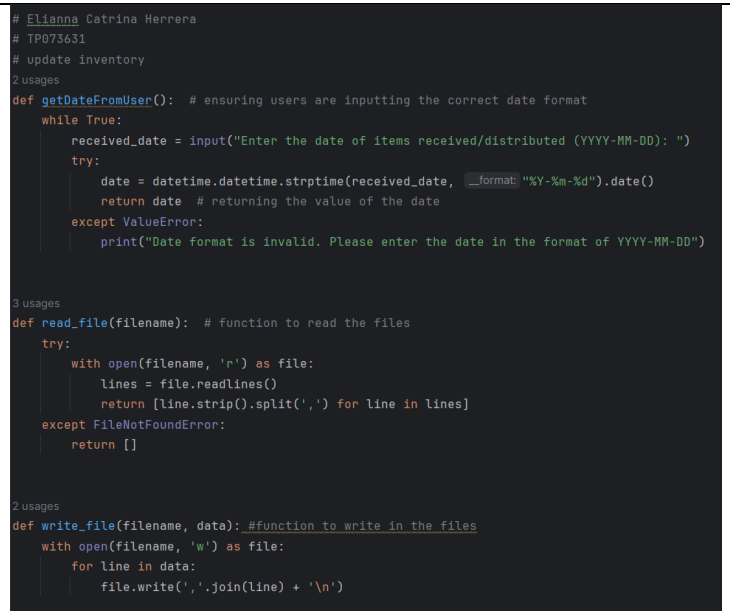
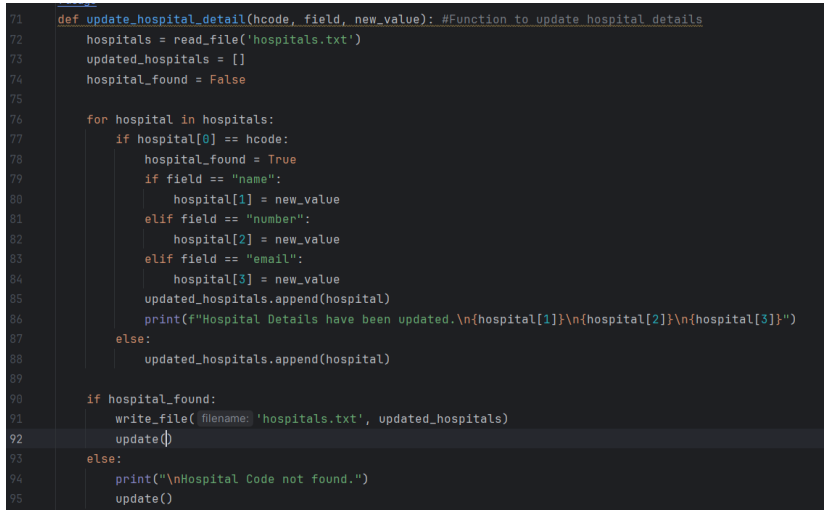
```python
    if user_choice == 1:
        Main.update()
    elif user_choice == 2:
        Main.item_inventory_tracking()
    elif user_choice == 3:
        Main.search()
    elif user_choice == 4:
        Main.report()
    elif user_choice == 5:
        print('Exiting program...')
        sys.exit(0)
```

Item Inventory Creation		The code begins by checking the existence of these three files (ppe.txt, supplier.txt, hospitals.txt). If all or either one of the files do not exist, the code will then create said file and prompt the user to fill up the file with the required details

Item Inventory Creation

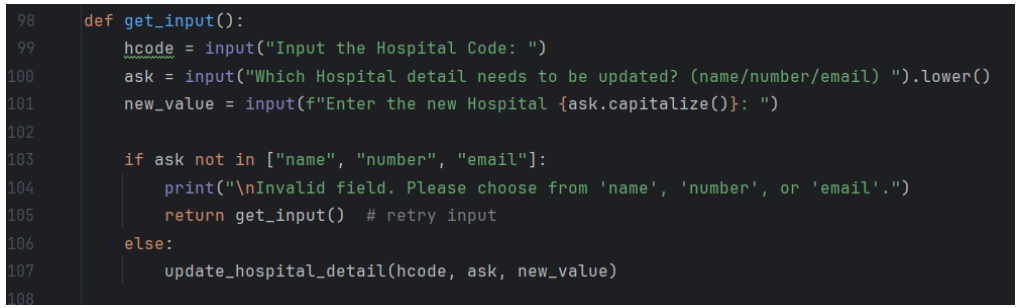
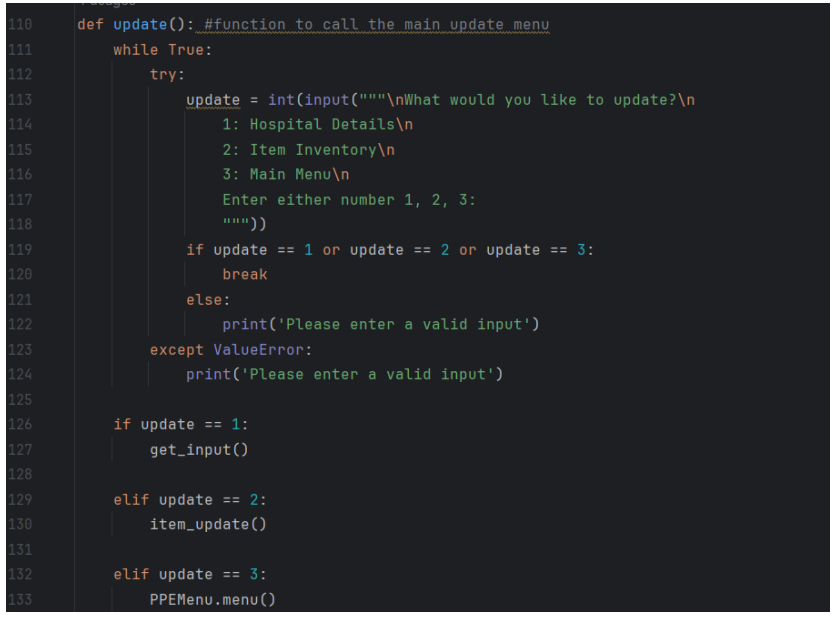
```python
def create_supplier():
    while True:
        try:
            user = int(input("How many suppliers are there: "))
            if user in [3, 4]:
                with open("supplier.txt", "w") as file:
                    for _ in range(user):
                        SCode = input("Input the Supplier Code: ")
                        SName = input("Input the Supplier Name: ")
                        SNumber = input("Input the Supplier Number: ")
                        SEmail = input("Input the Supplier Email: ")
                        file.write(f"{SCode},{SName},{SNumber},{SEmail}\n")
                print("Suppliers successfully added to supplier.txt.")
                break
            else:
                print("You can only have 3 or 4 suppliers")
        except ValueError:
            print("Invalid input. Please enter a valid number (3 or 4)")
```

```python
def create_hospitals():
    while True:
        try:
            user = int(input("How many hospitals are there: "))
            if user in [3, 4]:
                with open("hospitals.txt", "w") as file:
                    for _ in range(user):
                        HCode = input("Input the Hospital Code: ")
                        HName = input("Input the Hospital Name: ")
                        HNumber = input("Input the Hospital Number: ")
                        HEmail = input("Input the Hospital Email: ")
                        file.write(f"{HCode},{HName},{HNumber},{HEmail}\n")
                print("Hospitals successfully added to hospitals.txt.")
                break
            else:
                print("You can only have 3 or 4 hospitals")
        except ValueError:
            print("Invalid input. Please enter a valid number (3 or 4)")
```

Item Inventory Update	 	For Inventory Update, the code begins by defining three functions which will be used by subsequent functions. The first is to get a valid date, the second is to read and segment files, the third is to write in the file When updating hospital details, the code will prompt the user to input the hospital code and what detail would they like to update, if the hospital code is found the code will rewrite the text file to mirror the user input

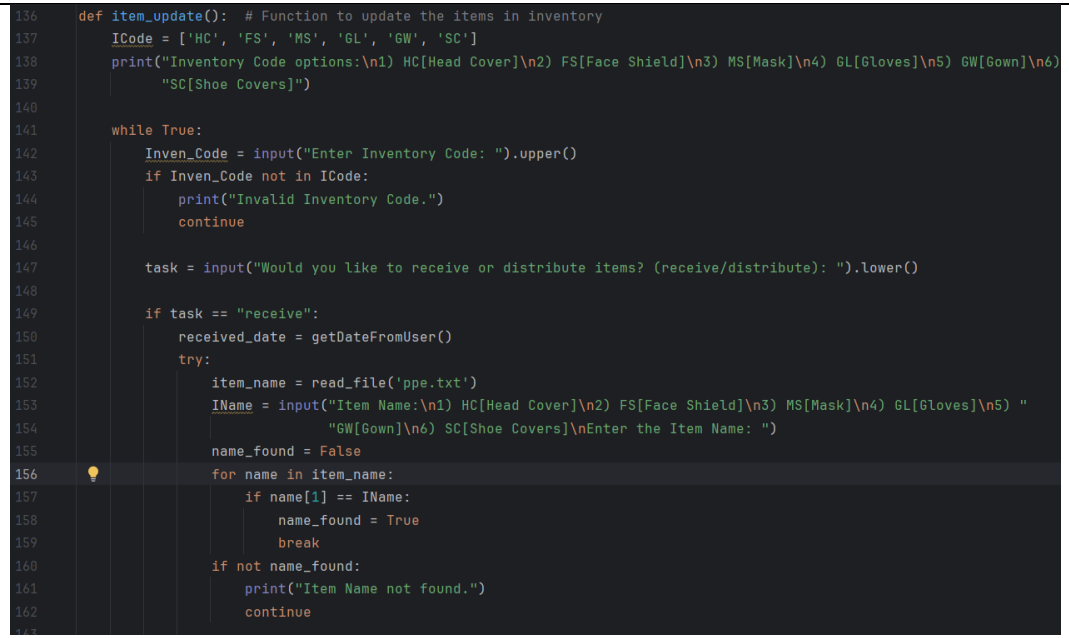
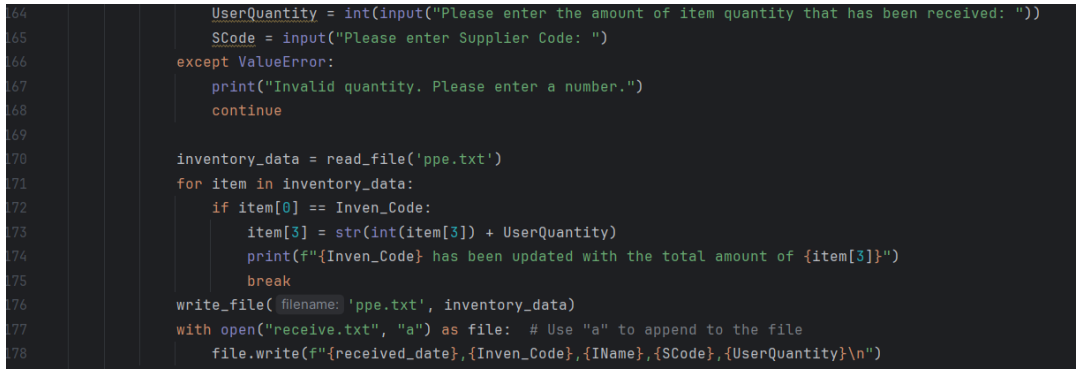
```
# Elianna Catrina Herrera
# TP073631
# update inventory
2 usages
def getDateFromUser():  # ensuring users are inputting the correct date format
    while True:
        received_date = input("Enter the date of items received/distributed (YYYY-MM-DD): ")
        try:
            date = datetime.datetime.strptime(received_date, __format: "%Y-%m-%d").date()
            return date  # returning the value of the date
        except ValueError:
            print("Date format is invalid. Please enter the date in the format of YYYY-MM-DD")


3 usages
def read_file(filename):  # function to read the files
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()
            return [line.strip().split(',') for line in lines]
    except FileNotFoundError:
        return []


2 usages
def write_file(filename, data): #function to write in the files
    with open(filename, 'w') as file:
        for line in data:
            file.write(','.join(line) + '\n')
```

```
71  def update_hospital_detail(hcode, field, new_value): #Function to update hospital details
72      hospitals = read_file('hospitals.txt')
73      updated_hospitals = []
74      hospital_found = False
75
76      for hospital in hospitals:
77          if hospital[0] == hcode:
78              hospital_found = True
79              if field == "name":
80                  hospital[1] = new_value
81              elif field == "number":
82                  hospital[2] = new_value
83              elif field == "email":
84                  hospital[3] = new_value
85              updated_hospitals.append(hospital)
86              print(f"Hospital Details have been updated.\n{hospital[1]}\n{hospital[2]}\n{hospital[3]}")
87          else:
88              updated_hospitals.append(hospital)
89
90      if hospital_found:
91          write_file( filename: 'hospitals.txt', updated_hospitals)
92          update()
93      else:
94          print("\nHospital Code not found.")
95          update()
```

Item Inventory Update	 	This is the function used to obtain user input when updating hospital details This function is used to determine the users desired action. Once the user has their desired action, the code will call the appropriate function

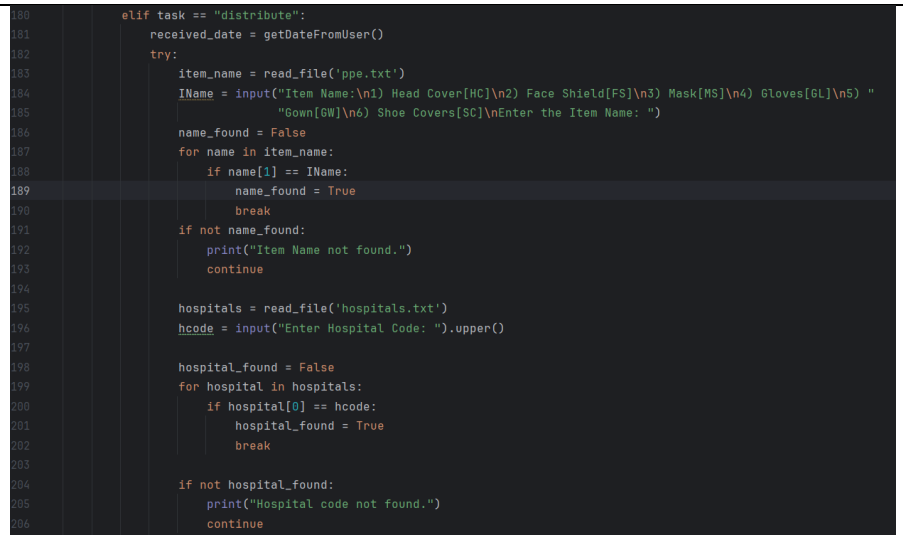
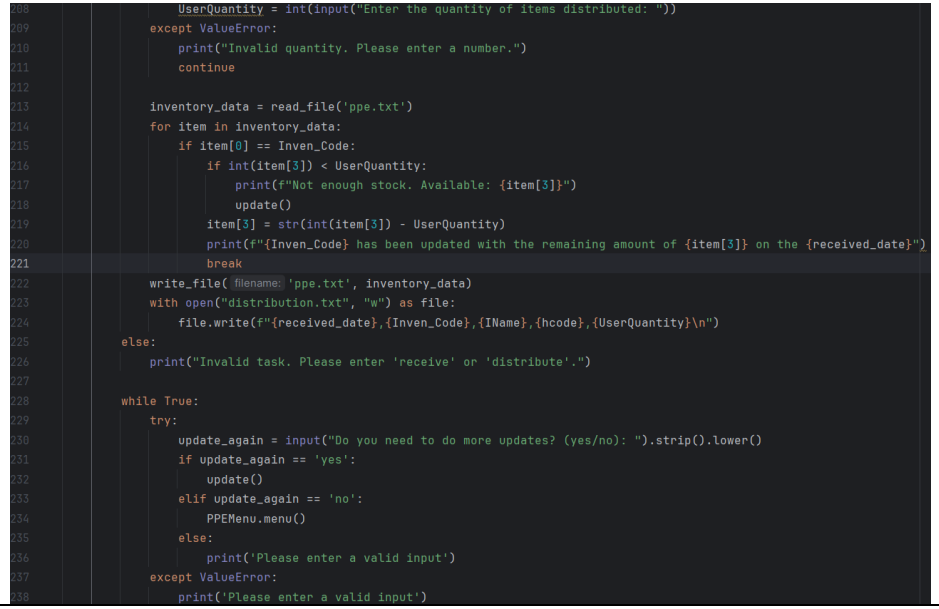
```python
def get_input():
    hcode = input("Input the Hospital Code: ")
    ask = input("Which Hospital detail needs to be updated? (name/number/email) ").lower()
    new_value = input(f"Enter the new Hospital {ask.capitalize()}: ")

    if ask not in ["name", "number", "email"]:
        print("\nInvalid field. Please choose from 'name', 'number', or 'email'.")
        return get_input()  # retry input
    else:
        update_hospital_detail(hcode, ask, new_value)
```

```python
def update(): #function to call the main update menu
    while True:
        try:
            update = int(input("""\nWhat would you like to update?\n
                1: Hospital Details\n
                2: Item Inventory\n
                3: Main Menu\n
                Enter either number 1, 2, 3:
                """))
            if update == 1 or update == 2 or update == 3:
                break
            else:
                print('Please enter a valid input')
        except ValueError:
            print('Please enter a valid input')

    if update == 1:
        get_input()

    elif update == 2:
        item_update()

    elif update == 3:
        PPEMenu.menu()
```

Item Inventory Update	 	If the user wishes to receive items, they will then be prompted to fill up the required details such as the Supplier Code & Quantity. Once all required details are filled up, the code will then rewrite the new values in both ppe.txt & receive.txt

```
def item_update():  # Function to update the items in inventory
    ICode = ['HC', 'FS', 'MS', 'GL', 'GW', 'SC']
    print("Inventory Code options:\n1) HC[Head Cover]\n2) FS[Face Shield]\n3) MS[Mask]\n4) GL[Gloves]\n5) GW[Gown]\n6)
          "SC[Shoe Covers]")

    while True:
        Inven_Code = input("Enter Inventory Code: ").upper()
        if Inven_Code not in ICode:
            print("Invalid Inventory Code.")
            continue

        task = input("Would you like to receive or distribute items? (receive/distribute): ").lower()

        if task == "receive":
            received_date = getDateFromUser()
            try:
                item_name = read_file('ppe.txt')
                IName = input("Item Name:\n1) HC[Head Cover]\n2) FS[Face Shield]\n3) MS[Mask]\n4) GL[Gloves]\n5) "
                              "GW[Gown]\n6) SC[Shoe Covers]\nEnter the Item Name: ")
                name_found = False
                for name in item_name:
                    if name[1] == IName:
                        name_found = True
                        break
                if not name_found:
                    print("Item Name not found.")
                    continue
```

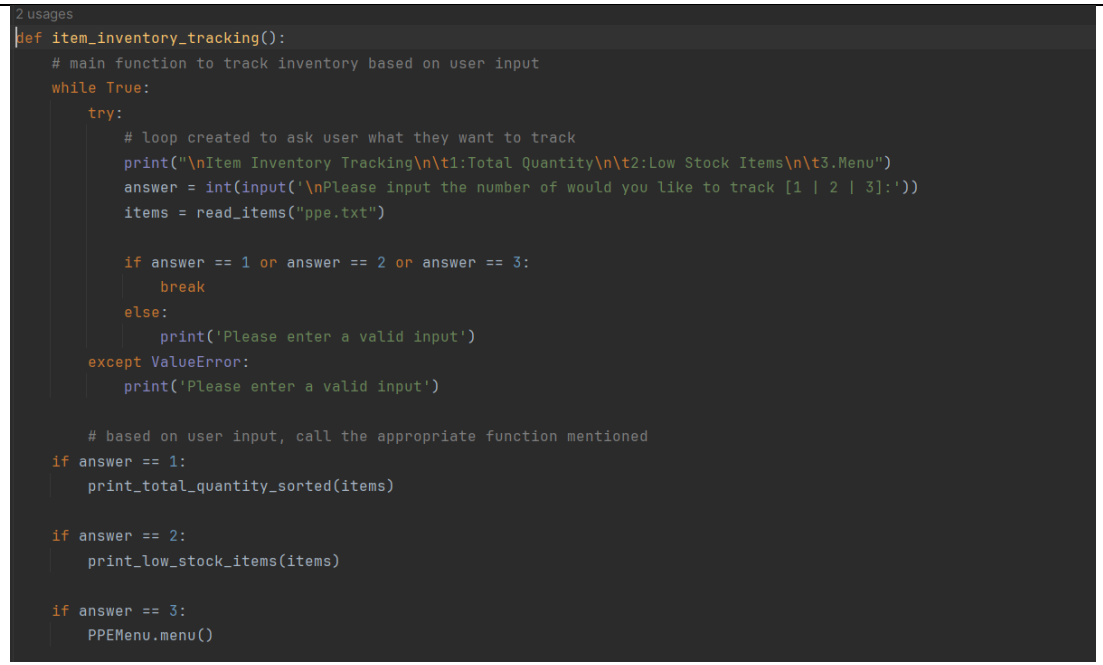
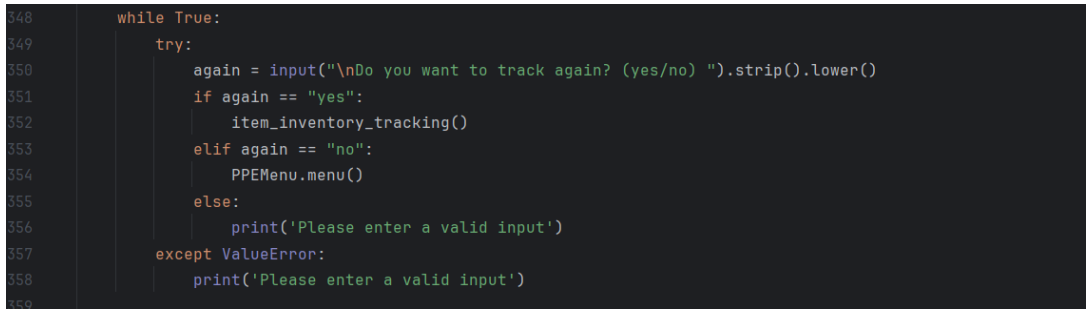
```
                UserQuantity = int(input("Please enter the amount of item quantity that has been received: "))
                SCode = input("Please enter Supplier Code: ")
            except ValueError:
                print("Invalid quantity. Please enter a number.")
                continue

            inventory_data = read_file('ppe.txt')
            for item in inventory_data:
                if item[0] == Inven_Code:
                    item[3] = str(int(item[3]) + UserQuantity)
                    print(f"{Inven_Code} has been updated with the total amount of {item[3]}")
                    break
            write_file( filename: 'ppe.txt', inventory_data)
            with open("receive.txt", "a") as file:  # Use "a" to append to the file
                file.write(f"{received_date},{Inven_Code},{IName},{SCode},{UserQuantity}\n")
```

Item Inventory Update	 	If the user wishes to distribute items however, they will then be prompted to fill up the required details such as the Hospital Code & Quantity. Once all required details are filled up, the code will then rewrite the new values in both ppe.txt & distribution.txt only if the quantity is sufficient

Item Inventory Tracking	 	The code begins by defining a function to read and segment the txt file as well as assigning each corresponding index with its value Next, the code has the ability to print the total quantity of all items in ascending order. Beyond that, the code also has the ability to print items with a quantity that is less than 25 if the user so desires

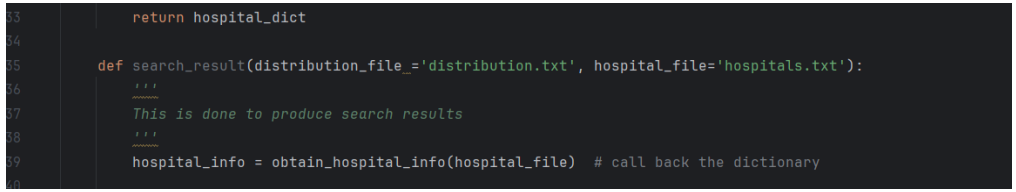
```
# item tracking
# Ng Vin Ee
# TP073088
1 usage
def read_items(file_name):
    items = []
    try:
        with open(file_name, 'r') as file:
            for line in file:
                # to skip empty lines
                if not line.strip():
                    continue
                # split each line with a comma
                parts = line.strip().split(',')
                # ensure each line has at least 4 parts
                if len(parts) >= 4:
                    item_code = parts[0]
                    item_name = parts[1]
                    # assuming size is parts[2], skip to parts[3] for quantity
                    try:
                        quantity = int(parts[3])
                        # append dictionary with the item details to the items
                        items.append({'item_code': item_code, 'item_name': item_name, 'quantity': quantity})
                    except ValueError:
                        print(f"Error: Quantity '{parts[3]}' is not a valid integer.")
                else:
                    print(f"Error: Line '{line.strip()}' does not have at least four parts.")
    except FileNotFoundError:
        print(f"Error: The file '{file_name}' was not found.")
    return items
```

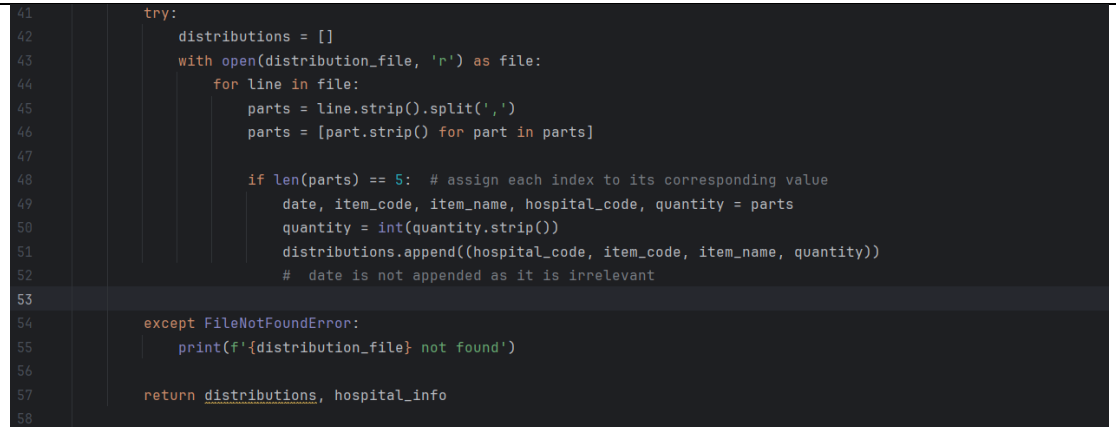
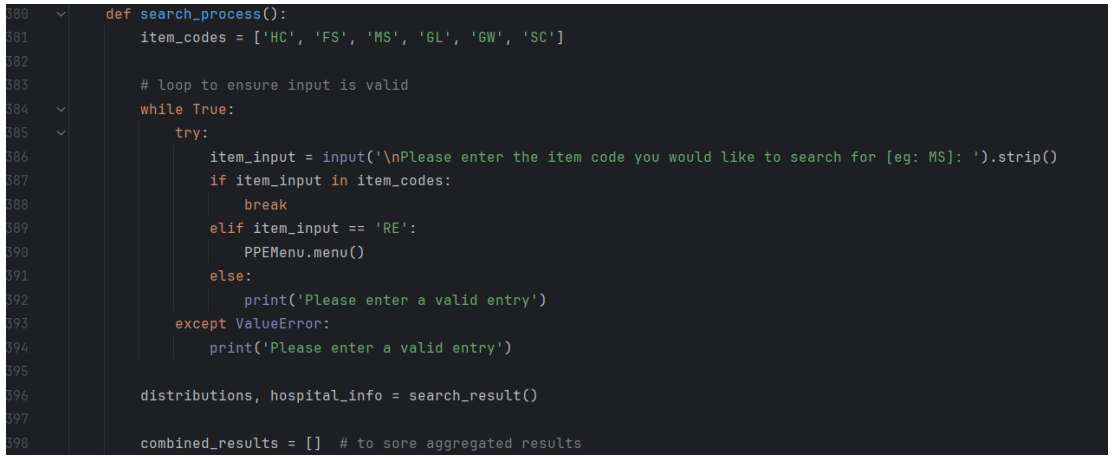
```
303     def print_total_quantity_sorted(items):
304         # a function to print items sorted by their quantity
305         items_sorted = sorted(items, key=lambda x: x['quantity'])
306         for item in items_sorted:
307             print(f"{item['item_code']}, {item['item_name']}, {item['quantity']}")
308
309
        1 usage
310     def print_low_stock_items(items):
311         # another function to print items with quantity that is less than 25
312         low_stock_found = False
313         for item in items:
314             if item['quantity'] < 25:
315                 print(f"{item['item_code']}, {item['item_name']}, {item['quantity']}")
316                 low_stock_found = True
317         if not low_stock_found:
318             print("No low stock items found.")
```

Item Inventory Tracking	 	Finally, this function is used to determine the users desired action. Once the user has their desired action, the code will call the appropriate function

```
2 usages
def item_inventory_tracking():
    # main function to track inventory based on user input
    while True:
        try:
            # loop created to ask user what they want to track
            print("\nItem Inventory Tracking\n\t1:Total Quantity\n\t2:Low Stock Items\n\t3.Menu")
            answer = int(input('\nPlease input the number of would you like to track [1 | 2 | 3]:'))
            items = read_items("ppe.txt")

            if answer == 1 or answer == 2 or answer == 3:
                break
            else:
                print('Please enter a valid input')
        except ValueError:
            print('Please enter a valid input')

        # based on user input, call the appropriate function mentioned
    if answer == 1:
        print_total_quantity_sorted(items)

    if answer == 2:
        print_low_stock_items(items)

    if answer == 3:
        PPEMenu.menu()
```

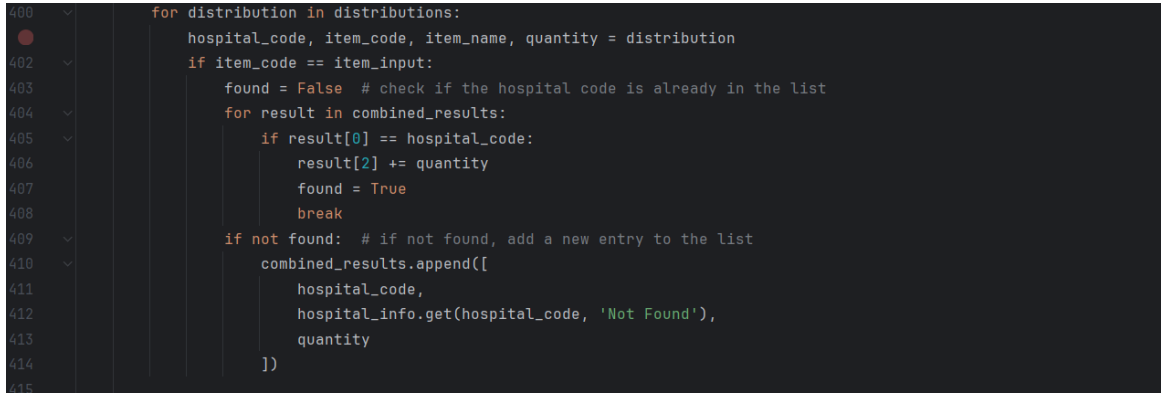
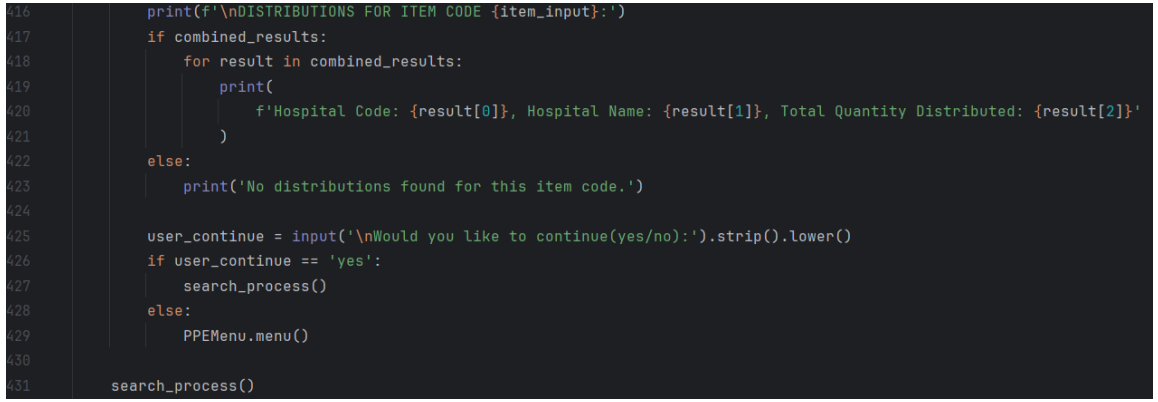
```
348         while True:
349             try:
350                 again = input("\nDo you want to track again? (yes/no) ").strip().lower()
351                 if again == "yes":
352                     item_inventory_tracking()
353                 elif again == "no":
354                     PPEMenu.menu()
355                 else:
356                     print('Please enter a valid input')
357             except ValueError:
358                 print('Please enter a valid input')
```

Searching Functionality	 	When the code is launched, users will be given a list of all items code for reference. Beyond that, the code will open hospital.txt to obtain the hospital code & name for future use

Searching Functionality	 	Once this is done, each index is assigned to its corresponding value. Users will then need to input a valid item code they wish to search for. Users can also input 'RE' to return to main menu

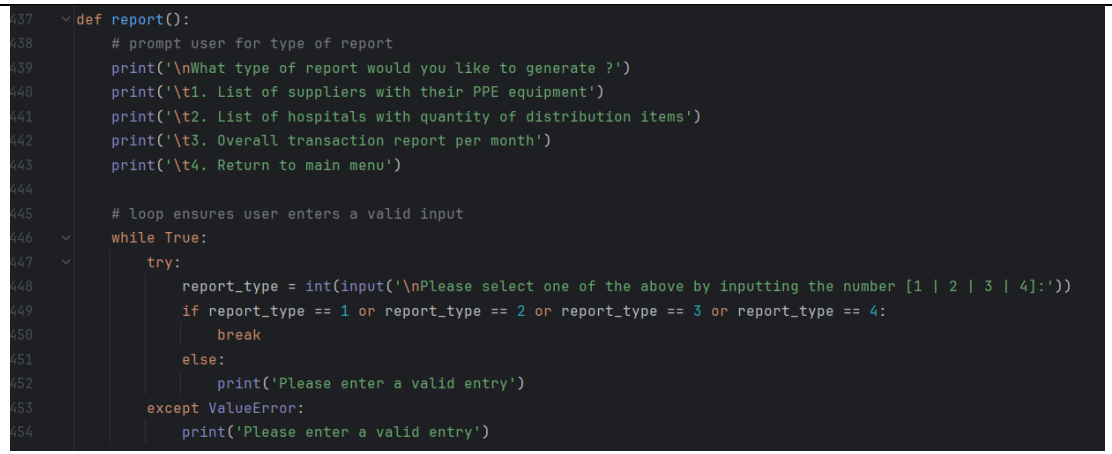
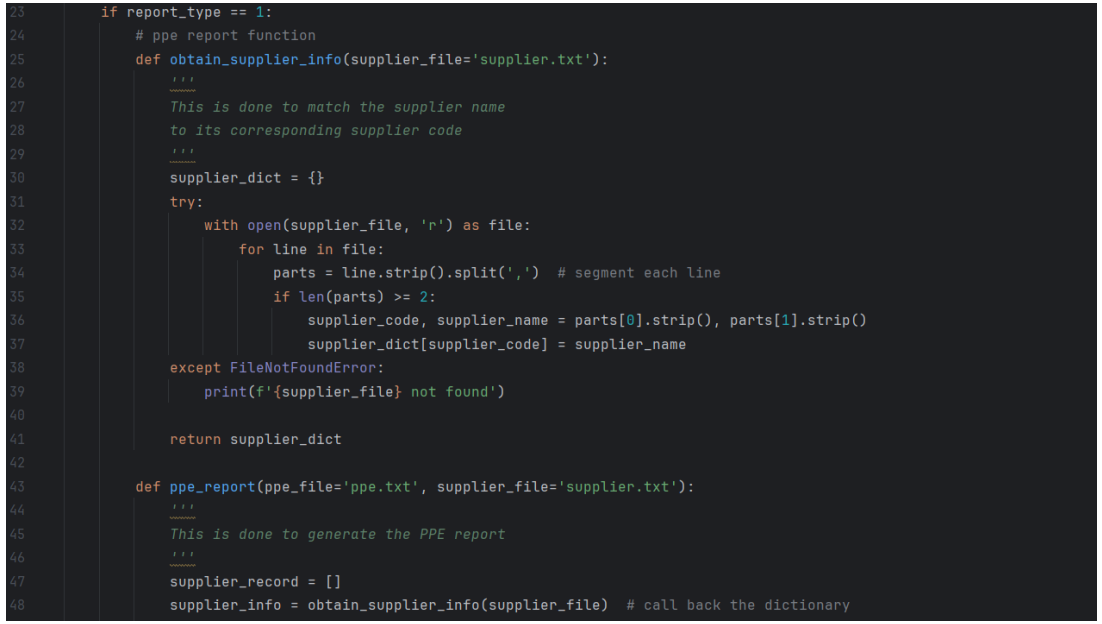
```
    try:
        distributions = []
        with open(distribution_file, 'r') as file:
            for line in file:
                parts = line.strip().split(',')
                parts = [part.strip() for part in parts]

                if len(parts) == 5:  # assign each index to its corresponding value
                    date, item_code, item_name, hospital_code, quantity = parts
                    quantity = int(quantity.strip())
                    distributions.append((hospital_code, item_code, item_name, quantity))
                    #  date is not appended as it is irrelevant

    except FileNotFoundError:
        print(f'{distribution_file} not found')

    return distributions, hospital_info

```

```
def search_process():
    item_codes = ['HC', 'FS', 'MS', 'GL', 'GW', 'SC']

    # loop to ensure input is valid
    while True:
        try:
            item_input = input('\nPlease enter the item code you would like to search for [eg: MS]: ').strip()
            if item_input in item_codes:
                break
            elif item_input == 'RE':
                PPEMenu.menu()
            else:
                print('Please enter a valid entry')
        except ValueError:
            print('Please enter a valid entry')

    distributions, hospital_info = search_result()

    combined_results = []  # to sore aggregated results
```

Searching Functionality	 	After this, the code will then search for distributions made under that item code. If multiple distributions were made under the same item, their quantity are summed up. The search results are then printed

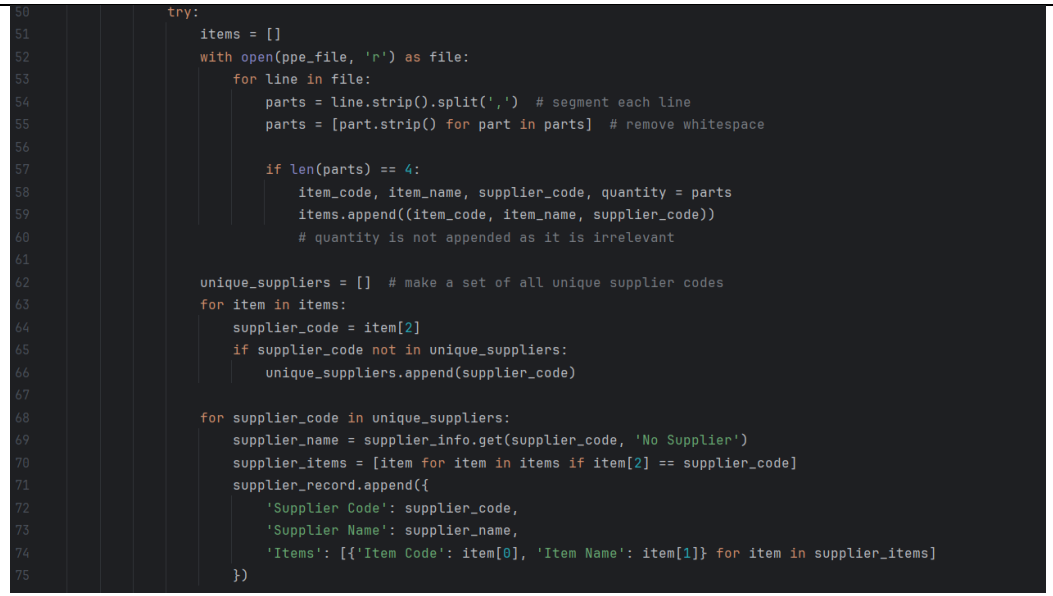
```
        for distribution in distributions:
            hospital_code, item_code, item_name, quantity = distribution
            if item_code == item_input:
                found = False  # check if the hospital code is already in the list
                for result in combined_results:
                    if result[0] == hospital_code:
                        result[2] += quantity
                        found = True
                        break
                if not found:  # if not found, add a new entry to the list
                    combined_results.append([
                        hospital_code,
                        hospital_info.get(hospital_code, 'Not Found'),
                        quantity
                    ])
```

```
        print(f'\nDISTRIBUTIONS FOR ITEM CODE {item_input}:')
        if combined_results:
            for result in combined_results:
                print(
                    f'Hospital Code: {result[0]}, Hospital Name: {result[1]}, Total Quantity Distributed: {result[2]}'
                )
        else:
            print('No distributions found for this item code.')

        user_continue = input('\nWould you like to continue(yes/no):').strip().lower()
        if user_continue == 'yes':
            search_process()
        else:
            PPEMenu.menu()

    search_process()
```

Report Functionality	 	When the code is launched, users will be prompted to select the type of report they desire. They do so by selecting 1 | 2 | 3 | 4 with each corresponding to a different report. Selecting 1 will generate the Suppliers Report. The code begins by opening the supplier.txt and ppe.txt files and segmenting each line as well as removing all the whitespace.

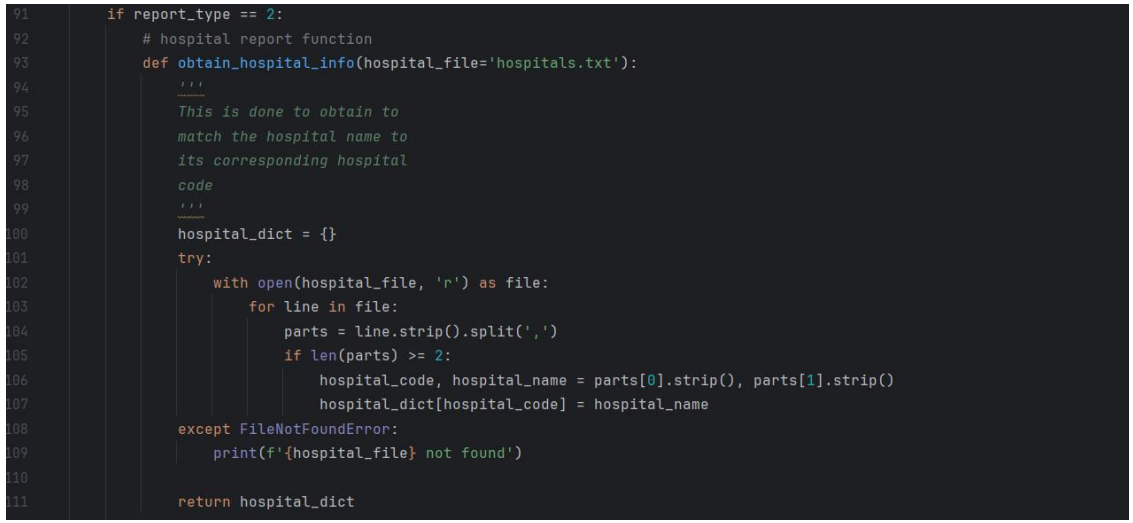
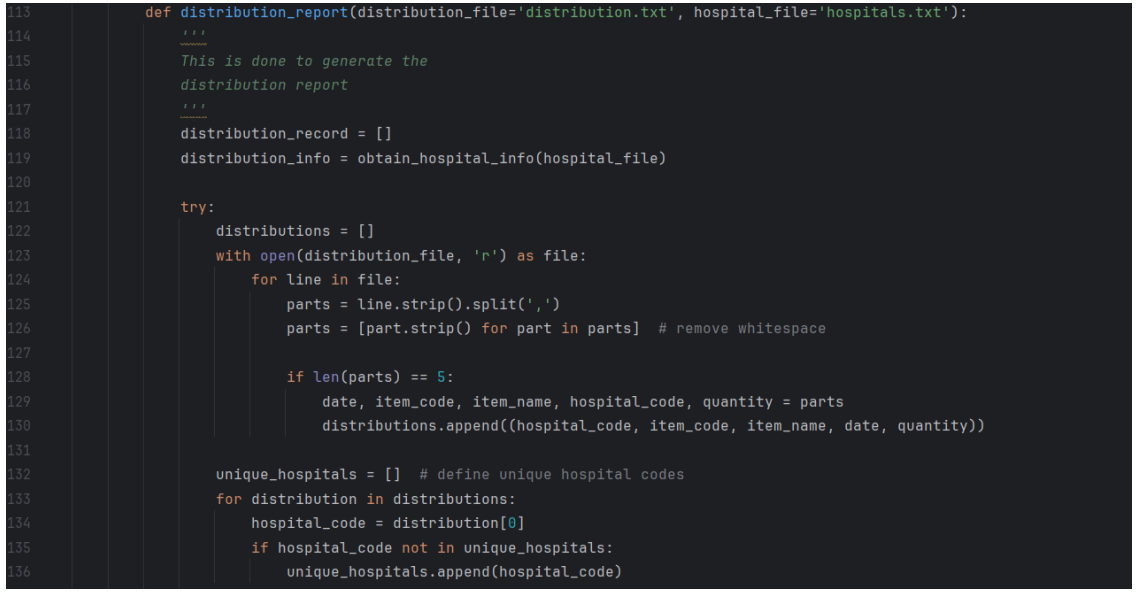
```
def report():
    # prompt user for type of report
    print('\nWhat type of report would you like to generate ?')
    print('\t1. List of suppliers with their PPE equipment')
    print('\t2. List of hospitals with quantity of distribution items')
    print('\t3. Overall transaction report per month')
    print('\t4. Return to main menu')

    # loop ensures user enters a valid input
    while True:
        try:
            report_type = int(input('\nPlease select one of the above by inputting the number [1 | 2 | 3 | 4]:'))
            if report_type == 1 or report_type == 2 or report_type == 3 or report_type == 4:
                break
            else:
                print('Please enter a valid entry')
        except ValueError:
            print('Please enter a valid entry')
```

```
    if report_type == 1:
        # ppe report function
        def obtain_supplier_info(supplier_file='supplier.txt'):
            '''
            This is done to match the supplier name
            to its corresponding supplier code
            '''
            supplier_dict = {}
            try:
                with open(supplier_file, 'r') as file:
                    for line in file:
                        parts = line.strip().split(',')  # segment each line
                        if len(parts) >= 2:
                            supplier_code, supplier_name = parts[0].strip(), parts[1].strip()
                            supplier_dict[supplier_code] = supplier_name
            except FileNotFoundError:
                print(f'{supplier_file} not found')

            return supplier_dict

        def ppe_report(ppe_file='ppe.txt', supplier_file='supplier.txt'):
            '''
            This is done to generate the PPE report
            '''
            supplier_record = []
            supplier_info = obtain_supplier_info(supplier_file)  # call back the dictionary
```

Report Functionality	 	Once that is done, each segment is assigned to its corresponding value and appended into the supplier record [] Beyond that, the output is printed in ascending order based on the Supplier Code.

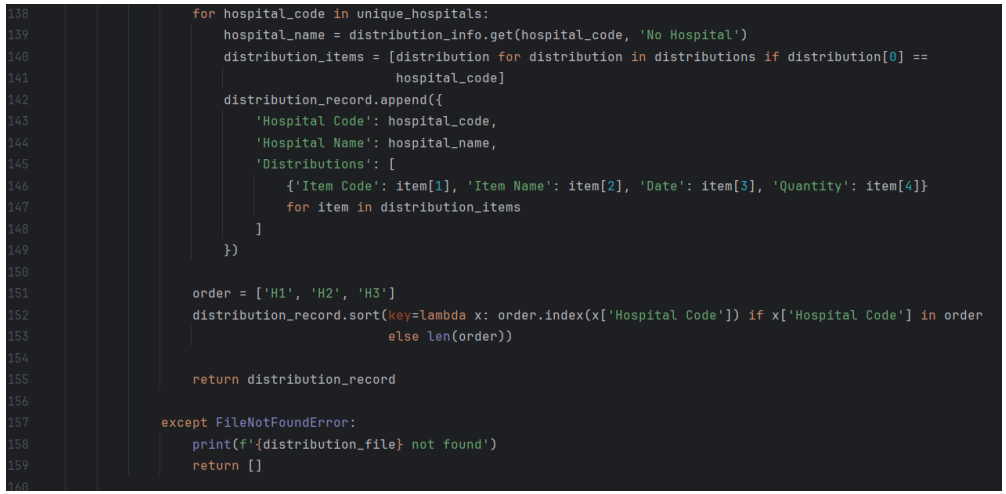
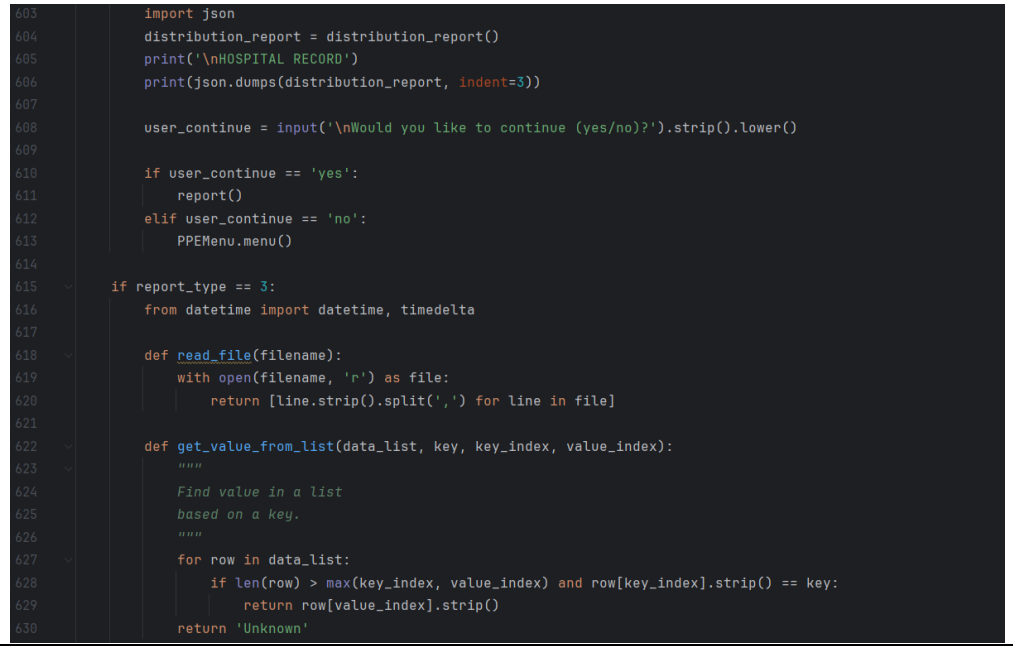
```
        try:
            items = []
            with open(ppe_file, 'r') as file:
                for line in file:
                    parts = line.strip().split(',')  # segment each line
                    parts = [part.strip() for part in parts]  # remove whitespace

                    if len(parts) == 4:
                        item_code, item_name, supplier_code, quantity = parts
                        items.append((item_code, item_name, supplier_code))
                        # quantity is not appended as it is irrelevant

            unique_suppliers = []  # make a set of all unique supplier codes
            for item in items:
                supplier_code = item[2]
                if supplier_code not in unique_suppliers:
                    unique_suppliers.append(supplier_code)

            for supplier_code in unique_suppliers:
                supplier_name = supplier_info.get(supplier_code, 'No Supplier')
                supplier_items = [item for item in items if item[2] == supplier_code]
                supplier_record.append({
                    'Supplier Code': supplier_code,
                    'Supplier Name': supplier_name,
                    'Items': [{'Item Code': item[0], 'Item Name': item[1]} for item in supplier_items]
                })
```

```
            order = ['S1', 'S2', 'S3', 'S4']  # order the output
            supplier_record.sort(
                key=lambda x: order.index('Supplier Code'[0]) if 'Supplier Code'[0] in order else
                len(order))
            return supplier_record

        except FileNotFoundError:
            print(f'{ppe_file} not found')
            return []

    import json
    ppe_report = ppe_report()
    print('\nSUPPLIER RECORD')
    print(json.dumps(ppe_report, indent=3))

    user_continue = input('\nWould you like to continue (yes/no)?').strip().lower()

    if user_continue == 'yes':
        report()
    elif user_continue == 'no':
        PPEMenu.menu()
```

Report Functionality	 	Selecting 2 will generate the Distributions Report. The code begins by opening the hospitals.txt and distribution.txt files and segmenting each line as well as removing all the whitespace. Each segment is assigned a value and appended into distributions record []

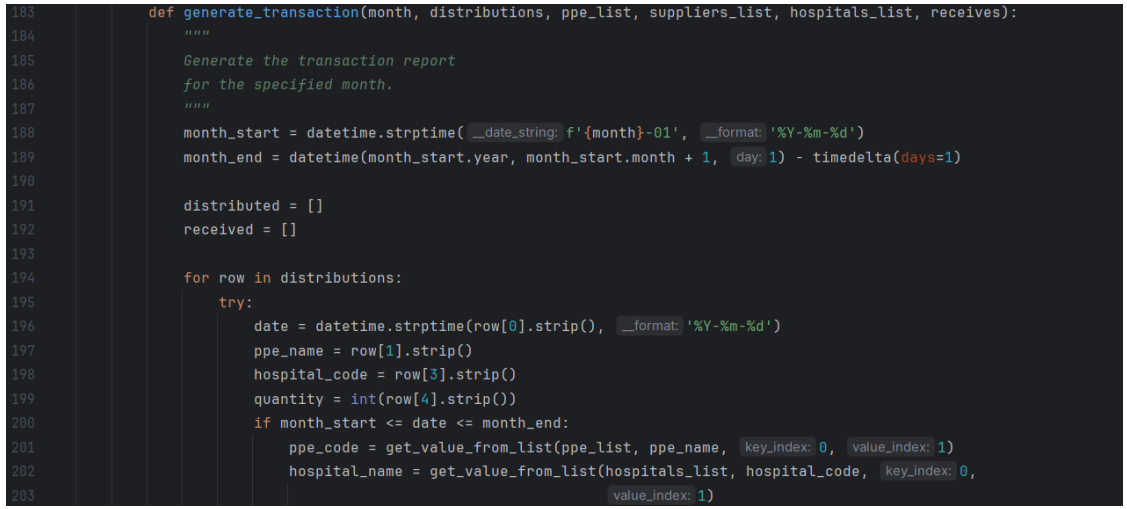
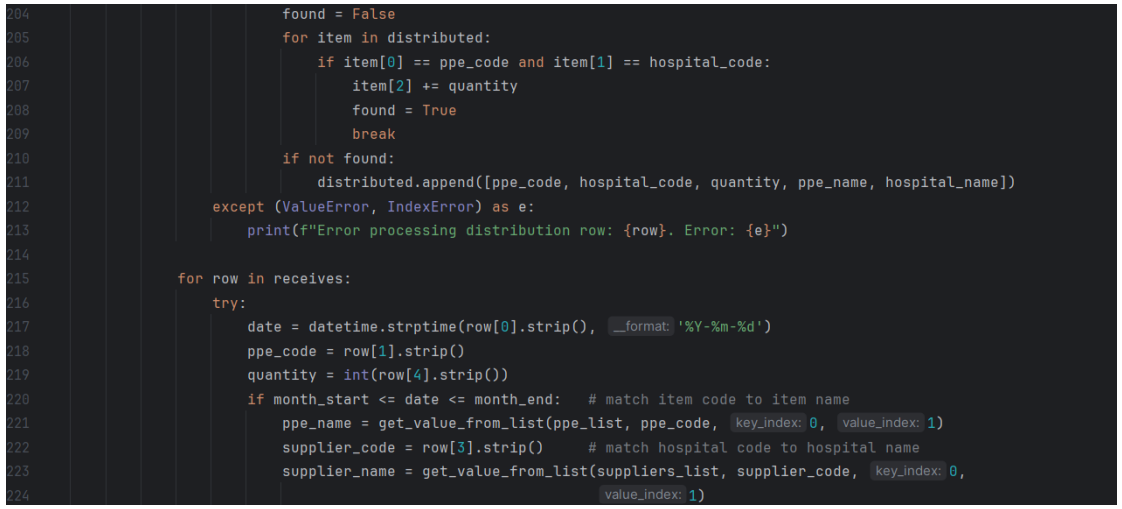
```
if report_type == 2:
    # hospital report function
    def obtain_hospital_info(hospital_file='hospitals.txt'):
        '''
        This is done to obtain to
        match the hospital name to
        its corresponding hospital
        code
        '''
        hospital_dict = {}
        try:
            with open(hospital_file, 'r') as file:
                for line in file:
                    parts = line.strip().split(',')
                    if len(parts) >= 2:
                        hospital_code, hospital_name = parts[0].strip(), parts[1].strip()
                        hospital_dict[hospital_code] = hospital_name
        except FileNotFoundError:
            print(f'{hospital_file} not found')

        return hospital_dict
```

```
    def distribution_report(distribution_file='distribution.txt', hospital_file='hospitals.txt'):
        '''
        This is done to generate the
        distribution report
        '''
        distribution_record = []
        distribution_info = obtain_hospital_info(hospital_file)

        try:
            distributions = []
            with open(distribution_file, 'r') as file:
                for line in file:
                    parts = line.strip().split(',')
                    parts = [part.strip() for part in parts]  # remove whitespace

                    if len(parts) == 5:
                        date, item_code, item_name, hospital_code, quantity = parts
                        distributions.append((hospital_code, item_code, item_name, date, quantity))

            unique_hospitals = []  # define unique hospital codes
            for distribution in distributions:
                hospital_code = distribution[0]
                if hospital_code not in unique_hospitals:
                    unique_hospitals.append(hospital_code)
```

Report Functionality	 	Beyond that, the output is printed in ascending order depending on the Hospital Code. Selecting 3 will generate the Overall Transaction report. First the code defines a function to segment each line in a file. After that, the code assigns a key to each list.

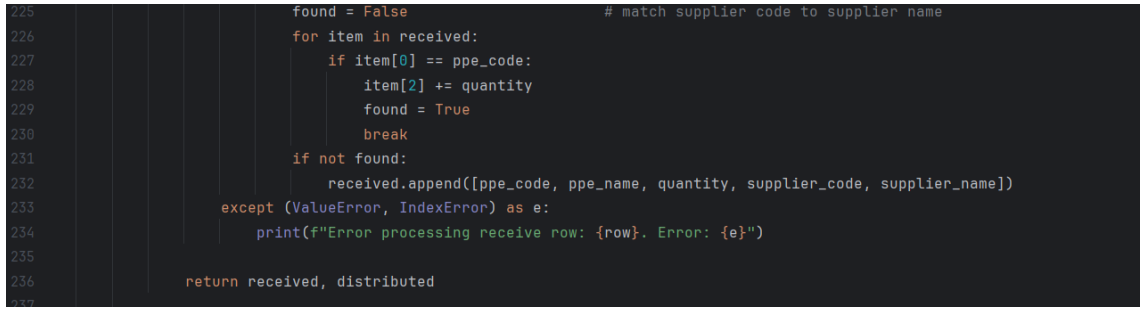
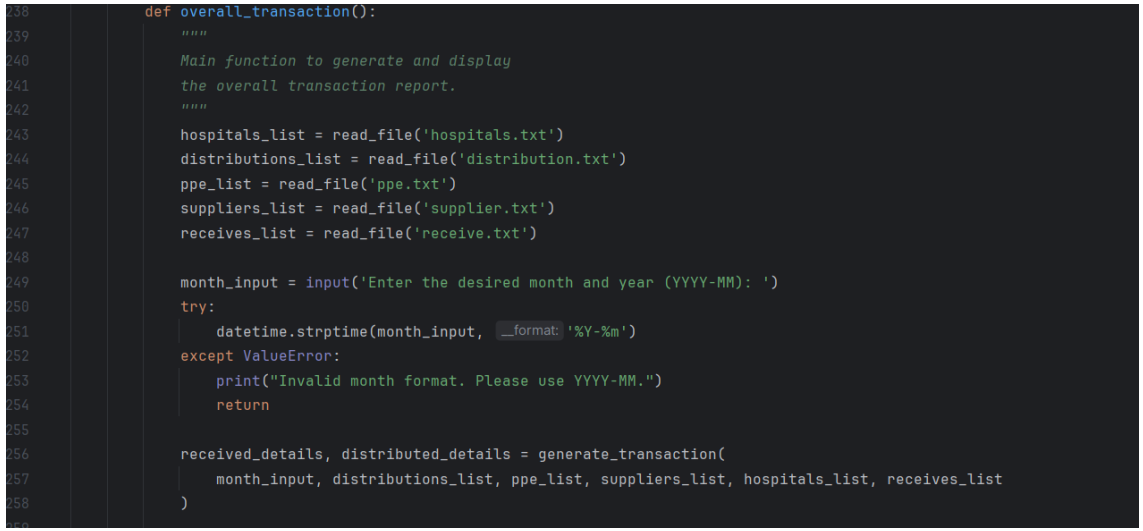
```
            for hospital_code in unique_hospitals:
                hospital_name = distribution_info.get(hospital_code, 'No Hospital')
                distribution_items = [distribution for distribution in distributions if distribution[0] ==
                                      hospital_code]
                distribution_record.append({
                    'Hospital Code': hospital_code,
                    'Hospital Name': hospital_name,
                    'Distributions': [
                        {'Item Code': item[1], 'Item Name': item[2], 'Date': item[3], 'Quantity': item[4]}
                        for item in distribution_items
                    ]
                })

            order = ['H1', 'H2', 'H3']
            distribution_record.sort(key=lambda x: order.index(x['Hospital Code']) if x['Hospital Code'] in order
                                     else len(order))

            return distribution_record

        except FileNotFoundError:
            print(f'{distribution_file} not found')
            return []
```

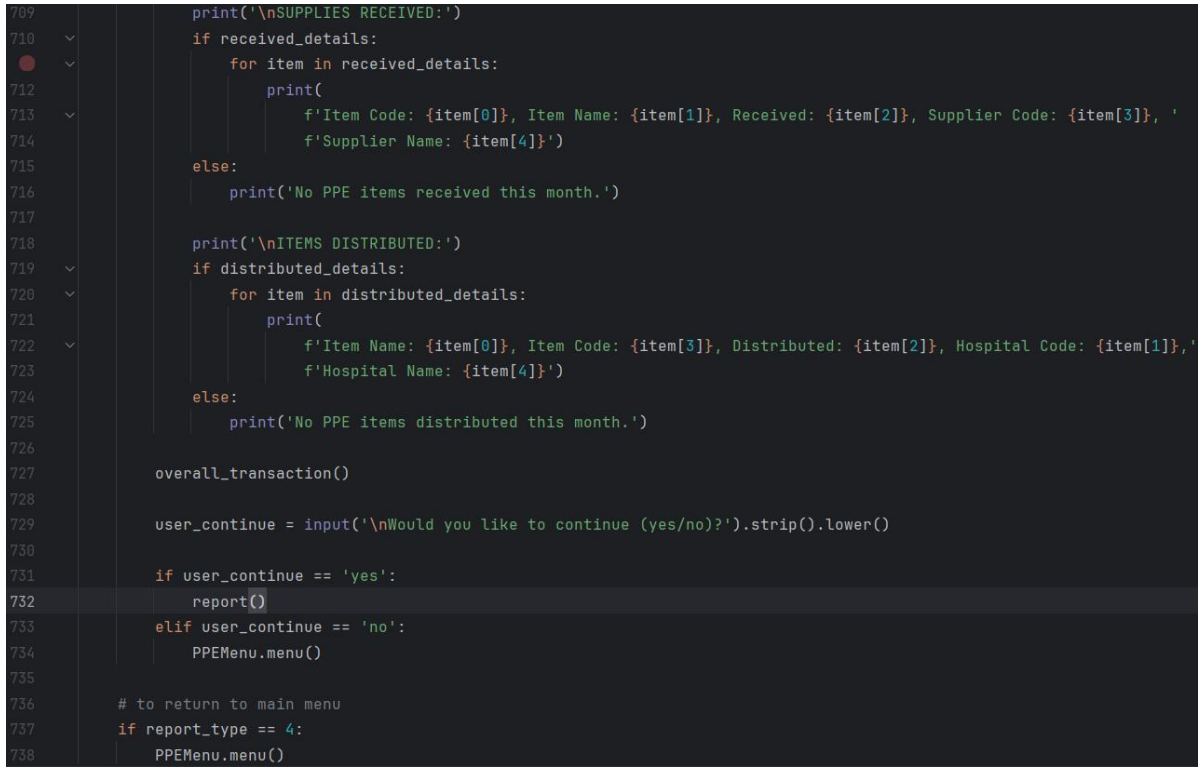
```
        import json
        distribution_report = distribution_report()
        print('\nHOSPITAL RECORD')
        print(json.dumps(distribution_report, indent=3))

        user_continue = input('\nWould you like to continue (yes/no)?').strip().lower()

        if user_continue == 'yes':
            report()
        elif user_continue == 'no':
            PPEMenu.menu()

    if report_type == 3:
        from datetime import datetime, timedelta

        def read_file(filename):
            with open(filename, 'r') as file:
                return [line.strip().split(',') for line in file]

        def get_value_from_list(data_list, key, key_index, value_index):
            """
            Find value in a list
            based on a key.
            """
            for row in data_list:
                if len(row) > max(key_index, value_index) and row[key_index].strip() == key:
                    return row[value_index].strip()
            return 'Unknown'
```

Report Functionality	 	Then, the code will gather all items received and distributed in a month as well as its details depending on its corresponding key index and appends it in either the distributed or received list

```
def generate_transaction(month, distributions, ppe_list, suppliers_list, hospitals_list, receives):
    """
    Generate the transaction report
    for the specified month.
    """
    month_start = datetime.strptime( _date_string: f'{month}-01',  _format: '%Y-%m-%d')
    month_end = datetime(month_start.year, month_start.month + 1,  day: 1) - timedelta(days=1)

    distributed = []
    received = []

    for row in distributions:
        try:
            date = datetime.strptime(row[0].strip(),  _format: '%Y-%m-%d')
            ppe_name = row[1].strip()
            hospital_code = row[3].strip()
            quantity = int(row[4].strip())
            if month_start <= date <= month_end:
                ppe_code = get_value_from_list(ppe_list, ppe_name,  key_index: 0,  value_index: 1)
                hospital_name = get_value_from_list(hospitals_list, hospital_code,  key_index: 0,
                                                    value_index: 1)
```

```
                found = False
                for item in distributed:
                    if item[0] == ppe_code and item[1] == hospital_code:
                        item[2] += quantity
                        found = True
                        break
                if not found:
                    distributed.append([ppe_code, hospital_code, quantity, ppe_name, hospital_name])
            except (ValueError, IndexError) as e:
                print(f"Error processing distribution row: {row}. Error: {e}")

        for row in receives:
            try:
                date = datetime.strptime(row[0].strip(),  _format: '%Y-%m-%d')
                ppe_code = row[1].strip()
                quantity = int(row[4].strip())
                if month_start <= date <= month_end:   # match item code to item name
                    ppe_name = get_value_from_list(ppe_list, ppe_code,  key_index: 0,  value_index: 1)
                    supplier_code = row[3].strip()     # match hospital code to hospital name
                    supplier_name = get_value_from_list(suppliers_list, supplier_code,  key_index: 0,
                                                        value_index: 1)
```

Report Functionality	 	Both lists are then returned, and the code prompts the user to input their desired month.

Report Functionality		The report is then printed.

```
            print('\nSUPPLIES RECEIVED:')
            if received_details:
                for item in received_details:
                    print(
                        f'Item Code: {item[0]}, Item Name: {item[1]}, Received: {item[2]}, Supplier Code: {item[3]}, '
                        f'Supplier Name: {item[4]}')
            else:
                print('No PPE items received this month.')

            print('\nITEMS DISTRIBUTED:')
            if distributed_details:
                for item in distributed_details:
                    print(
                        f'Item Name: {item[0]}, Item Code: {item[3]}, Distributed: {item[2]}, Hospital Code: {item[1]},'
                        f'Hospital Name: {item[4]}')
            else:
                print('No PPE items distributed this month.')

        overall_transaction()

        user_continue = input('\nWould you like to continue (yes/no)?').strip().lower()

        if user_continue == 'yes':
            report()
        elif user_continue == 'no':
            PPEMenu.menu()

    # to return to main menu
    if report_type == 4:
        PPEMenu.menu()
```

2.0 Sample Input / Output

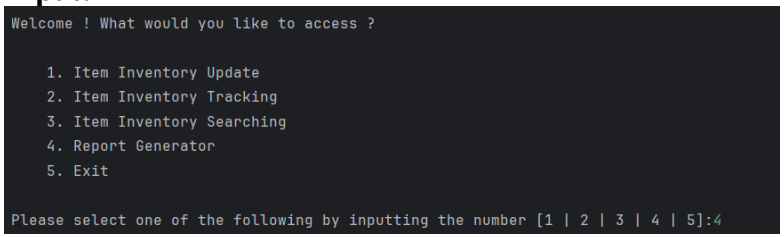
Below are the sample input / outputs alongside with the explanations

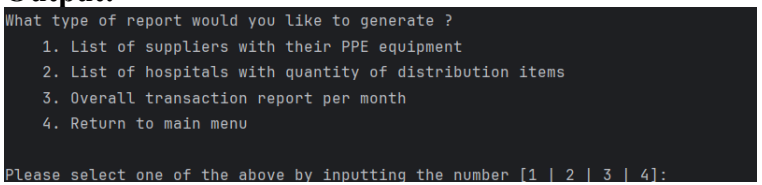
Name	Input / Output	Explanation

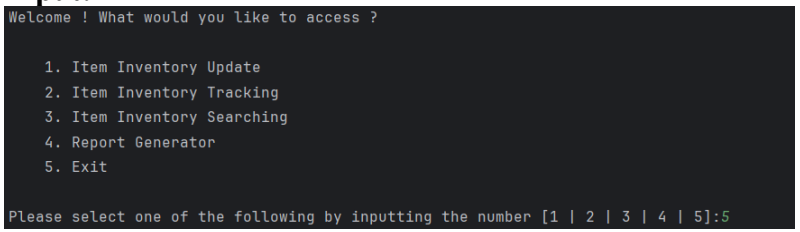
Login	**VALID LOGIN ENTRY**
	
	Input:
	
	`Welcome to Inventory Management for Personal Protective Equipment (PPE)`
	`Please Log in to proceed`
	
	`Please enter your username: IC1`
	`Please enter your password: ventcon123`
	
	Output:
	
	`Login Successful`
	
	`Ppe.txt File exists`
	
	`Supplier.txt File exists`
	
	`Hospitals.txt File exists`
	
	`Welcome ! What would you like to access ?`
	
	`1. Item Inventory Update`
	`2. Item Inventory Tracking`
	`3. Item Inventory Searching`
	`4. Report Generator`
	`5. Exit`
	
	`Please select one of the following by inputting the number [1 | 2 | 3 | 4 | 5]:`
		When the program starts, users will need to enter their login credentials. If successful, they will be given access to the program

Login	**INVALID LOGIN ENTRY** **Input:** Welcome to Inventory Management for Personal Protective Equipment (PPE) Please Log in to proceed Please enter your username: *IC2* Please enter your password: *ventcon123* Username or password is incorrect. Please try again. Please enter your username: *IC3* Please enter your password: *ventcon123* Username or password is incorrect. Please try again. Please enter your username: *IC1* Please enter your password: *ventd* **Output:** You have reached the maximum number of attempts to log in. Process finished with exit code 0	If the user exceeds the maximum attempts, the program will terminate

<table>
<tr>
<td rowspan="2">Menu</td>
<td>FUNCTION SELECTION

Input:
 

Output:
 
</td>
<td>Selecting any of 5 options will bring the user to the corresponding function</td>
</tr>
<tr>
<td>EXIT

Input:
 

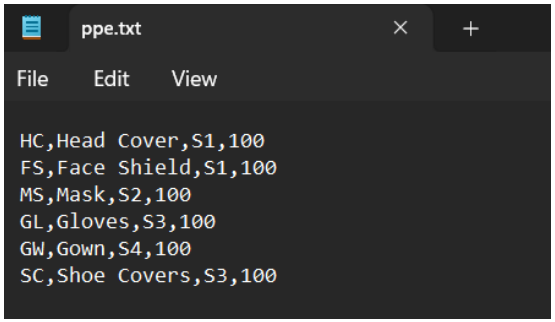
Output:
 
</td>
<td>Selecting 5 (Exit) will cause the program to terminate</td>
</tr>
</table>

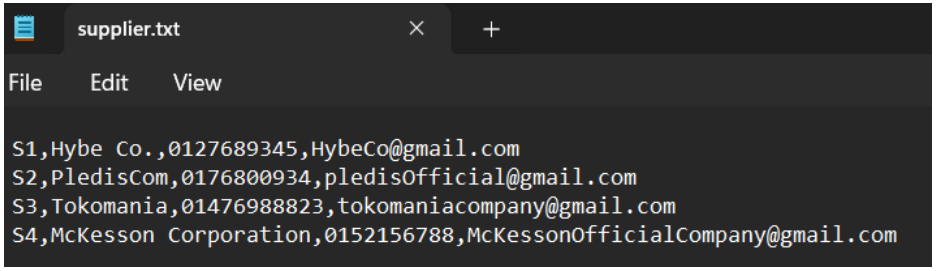
Item Inventory Creation	**FILE CREATION** **Input:** How many items do you want to input: 6 Enter item code: HC Enter item name: Head Cover Enter supplier code: S1 Enter item code: FS Enter item name: Face Shield Enter supplier code: S1 Enter item code: MS Enter item name: Mask Enter supplier code: S2 Enter item code: GL Enter item name: Gloves Enter supplier code: S3 Enter item code: GW Enter item name: Gown Enter supplier code: S4 Enter item code: SC Enter item name: Shoe Covers Enter supplier code: S3 How many suppliers are there: 4 Input the Supplier Code: S1 Input the Supplier Name: Hybe Co. Input the Supplier Number: 0123477845 Input the Supplier Email: HybeCo@gmail.com Input the Supplier Code: S2 Input the Supplier Name: PledisCom Input the Supplier Number: 0135784598 Input the Supplier Email: pledisOfficial@gmail.com Input the Supplier Code: S3 Input the Supplier Name: Tokomania Input the Supplier Number: 01476988823 Input the Supplier Email: tokomaniacompany@gmail.com Input the Supplier Code: S4 Input the Supplier Name: McKesson Corporation Input the Supplier Number: 01354781200 Input the Supplier Email: McKessonOfficial@gmail.com Suppliers successfully added to supplier.txt.	For each file that does not exist, the program prompts the user to populate the file with the required data. The file is then created and filled with the data inputted

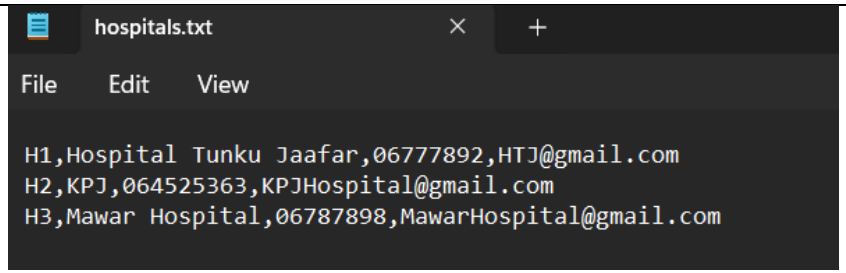
Item Inventory Creation

```
How many hospitals are there: 3
Input the Hospital Code: H1
Input the Hospital Name: Hospital Tuanku Jaafar
Input the Hospital Number: 06777892
Input the Hospital Email: HTJ@gmail.com
Input the Hospital Code: H2
Input the Hospital Name: KPJ
Input the Hospital Number: 062525363
Input the Hospital Email: KPJHospital@gmail.com
Input the Hospital Code: H3
Input the Hospital Name: Mawar Hospital
Input the Hospital Number: 06787898
Input the Hospital Email: MawarHospital@gmail.com
Hospitals successfully added to hospitals.txt.
```

Output:

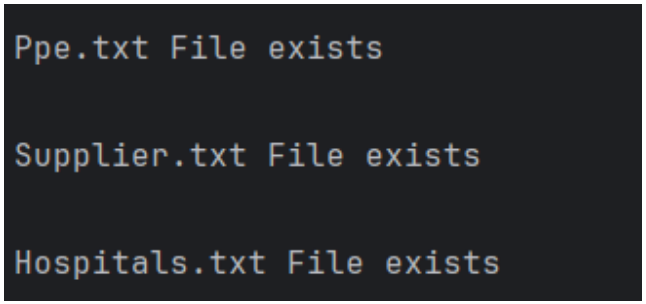




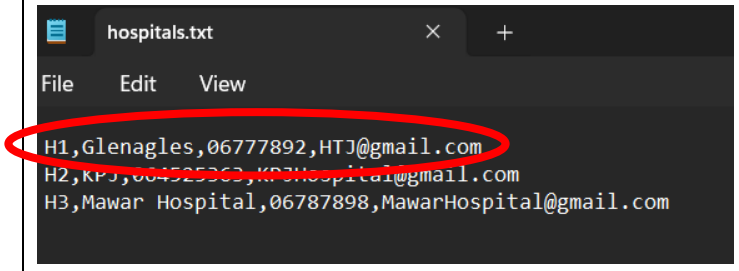
<table>
<tr>
<td rowspan="2">Item Inventory Creation</td>
<td>


</td>
<td></td>
</tr>
<tr>
<td>
FILE EXISTS

Output:

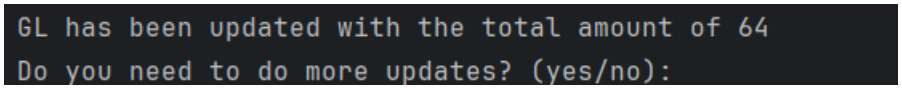
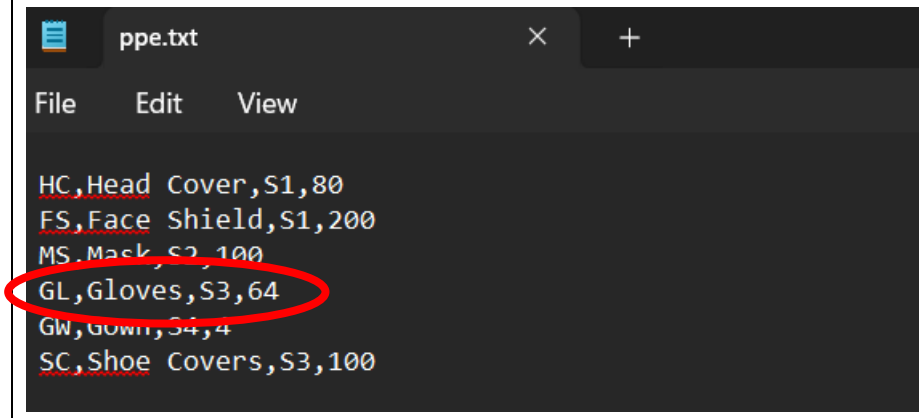
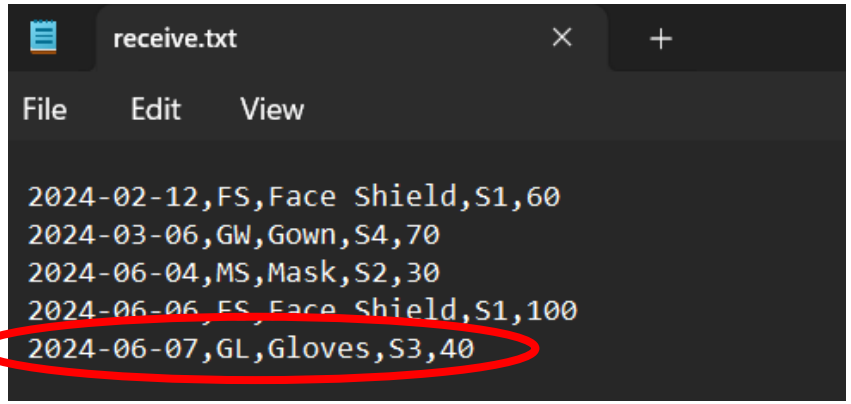


</td>
<td>If the file exists, a confirmation message is printed.</td>
</tr>
</table>

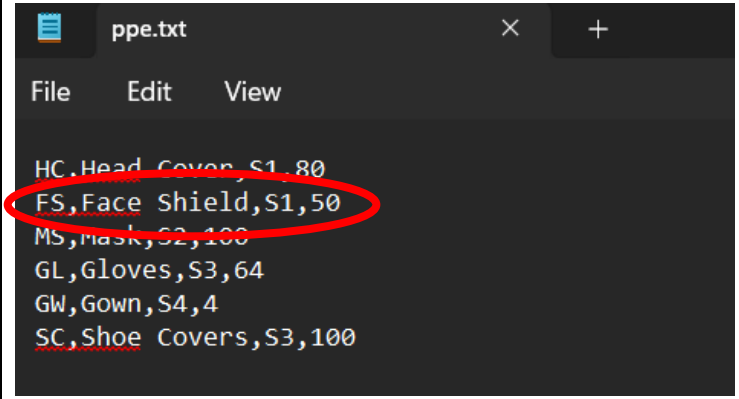
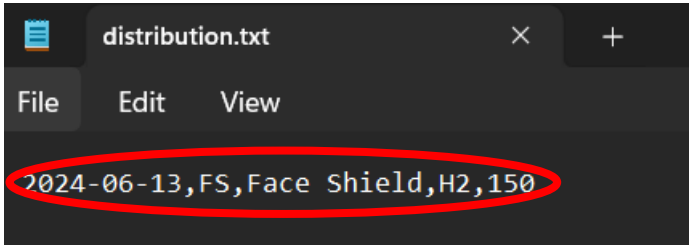
Item Inventory Update	**HOSPITAL DETAILS UPDATE** **Input:** What would you like to update? 1: Hospital Details 2: Item Inventory 3: Main Menu Enter either number 1, 2, 3: *1* Input the Hospital Code: *H1* Which Hospital detail needs to be updated? (name/number/email) *name* Enter the new Hospital Name: *Glenagles* **Output:** Hospital Details have been updated. Glenagles 06777892 HTJ@gmail.com  	When updating hospital details, users will be required to enter the hospital code, the field they wish to update & the new value. Once done, the new values will appear in the hospitals.txt file

Item Inventory Update	**INVALID HOSPITAL CODE** **Input:** What would you like to update? 1: Hospital Details 2: Item Inventory 3: Main Menu Enter either number 1, 2, 3: *1* Input the Hospital Code: *H5* Which Hospital detail needs to be updated? (name/number/email) *name* Enter the new Hospital Name: *Ampang Puteri* **Output:** Hospital Code not found. What would you like to update? 1: Hospital Details 2: Item Inventory 3: Main Menu Enter either number 1, 2, 3:	If the Hospital Code does not exist, the program will return to the update menu.

Item Inventory Update	**RECEIVE SUPPLY** **Input:** What would you like to update? 1: Hospital Details 2: Item Inventory 3: Main Menu Enter either number 1, 2, 3: *2* Inventory Code options: 1) HC[Head Cover] 2) FS[Face Shield] 3) MS[Mask] 4) GL[Gloves] 5) GW[Gown] 6) SC[Shoe Covers] Enter Inventory Code: *GL* Would you like to receive or distribute items? (receive/distribute): *receive* Enter the date of items received/distributed (YYYY-MM-DD): *2024-06-07* Item Name: 1) HC[Head Cover] 2) FS[Face Shield] 3) MS[Mask] 4) GL[Gloves] 5) GW[Gown] 6) SC[Shoe Covers] Enter the Item Name: *Gloves* Please enter the amount of item quantity that has been received: *40* Please enter Supplier Code: *S3*	When receiving supply, users will be required to supply the Item Code, Date, Item Name, Quantity & Supplier Code

Item Inventory Update	**Output:**    	Once this is done the program will confirm the new quantity and subsequently update the new value in both ppe.txt & receive.txt

Item Inventory Update	**DISTRIBUTE SUPPLY** **Input:** What would you like to update? 1: Hospital Details 2: Item Inventory 3: Main Menu Enter either number 1, 2, 3: *2* Inventory Code options: 1) HC[Head Cover] 2) FS[Face Shield] 3) MS[Mask] 4) GL[Gloves] 5) GW[Gown] 6) SC[Shoe Covers] Enter Inventory Code: *FS* Would you like to receive or distribute items? (receive/distribute): *distribute* Enter the date of items received/distributed (YYYY-MM-DD): *2024-06-13* Item Name: 1) Head Cover[HC] 2) Face Shield[FS] 3) Mask[MS] 4) Gloves[GL] 5) Gown[GW] 6) Shoe Covers[SC] Enter the Item Name: *Face Shield* Enter Hospital Code: *H2* Enter the quantity of items distributed: *150*	When distributing supply, users will be required to supply the Item Code, Date, Item Name, Quantity & Hospital Code

Item Inventory Update	**Output:**
	
	`FS has been updated with the remaining amount of 50 on the 2024-06-13`
	`Do you need to do more updates? (yes/no):`
	
	 
	
	 
	
		Once this is done the program will confirm the new quantity and subsequently update the new value in both ppe.txt & distributions.txt

Item Inventory Update	**INSUFICCIENT QUANTITY** **Input:** Enter Inventory Code: *FS* Would you like to receive or distribute items? (receive/distribute): *distribute* Enter the date of items received/distributed (YYYY-MM-DD): *2026-05-05* Item Name: 1) Head Cover[HC] 2) Face Shield[FS] 3) Mask[MS] 4) Gloves[GL] 5) Gown[GW] 6) Shoe Covers[SC] Enter the Item Name: *Face Shield* Enter Hospital Code: *H1* Enter the quantity of items distributed: *150* **Output:** Not enough stock. Available: 50 What would you like to update? 1: Hospital Details 2: Item Inventory 3: Main Menu Enter either number 1, 2, 3:	If the quantity available is insufficient, the program will return the user to the update menu

<table>
<tr>
<td rowspan="2">Item Inventory Tracking</td>
<td>PRINT TOTAL QUANTITY

Input:

<pre>Item Inventory Tracking
    1:Total Quantity
    2:Low Stock Items
    3.Menu

Please input the number of would you like to track [1 | 2 | 3]:1</pre>

Output:

<pre>GW, Gown, 4
GL, Gloves, 24
HC, Head Cover, 80
FS, Face Shield, 100
MS, Mask, 100
SC, Shoe Covers, 100

Do you want to track again? (yes/no)</pre></td>
<td>If the user inputs 1, the program will print out the quantity of all items in ascending order</td>
</tr>
<tr>
<td>PRINT LOW STOCK ITEMS

Input:

<pre>Item Inventory Tracking
    1:Total Quantity
    2:Low Stock Items
    3.Menu

Please input the number of would you like to track [1 | 2 | 3]:2</pre>

Output:

<pre>GL, Gloves, 24
GW, Gown, 4

Do you want to track again? (yes/no)</pre></td>
<td>If the user inputs 2, the program will print out all items with quantities less than 25</td>
</tr>
</table>

<table>
<tr>
<td rowspan="2">Item Inventory Tracking</td>
<td>NO LOW STOCK ITEMS

Input:


```
Item Inventory Tracking
    1:Total Quantity
    2:Low Stock Items
    3.Menu

Please input the number of would you like to track [1 | 2 | 3]:2
```


Output:


```
No low stock items found.
```

</td>
<td>If there are no entries, then the program will alert the user</td>
</tr>
<tr>
<td>INVALID ENTRY

Input:


```
Item Inventory Tracking
    1:Total Quantity
    2:Low Stock Items
    3.Menu

Please input the number of would you like to track [1 | 2 | 3]:4
```


Output:


```
Please enter a valid input
```

</td>
<td>If the user inputs an invalid entry, the program will repeatedly prompt the user for a valid entry until successful</td>
</tr>
</table>

<table>
<tr>
<td rowspan="2">Searching Functionality</td>
<td>ITEM CODE ENTRY

Input:

```
ITEM CODES
    HC: Head Cover  FS: Face Shield
    MS: Mask        GL: Gloves
    GW: Gown        SC: Shoe Cover

    RE: Return to Main Menu

Please enter the item code you would like to search for [eg: MS]: MS
```


Output:

```
DISTRIBUTIONS FOR ITEM CODE MS:
Hospital Code: H3, Hospital Name: Mawar Hospital, Total Quantity Distributed: 50
Hospital Code: H1, Hospital Name: Hospital Tunku Jaafar, Total Quantity Distributed: 20
```

</td>
<td>The user shall input an item code based on the list above. If valid, the code will then show all distributions made under the item code alongside which hospital the item was distributed to as well as its quantity</td>
</tr>
<tr>
<td>INVALID ITEM CODE ENTRY

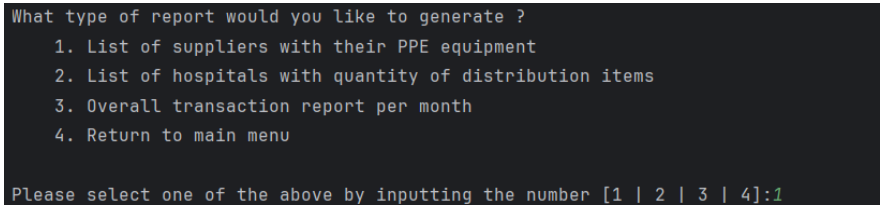
Input:

```
ITEM CODES
    HC: Head Cover  FS: Face Shield
    MS: Mask        GL: Gloves
    GW: Gown        SC: Shoe Cover

    RE: Return to Main Menu

Please enter the item code you would like to search for [eg: MS]: st
```

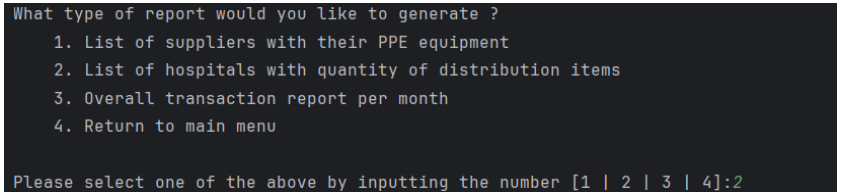
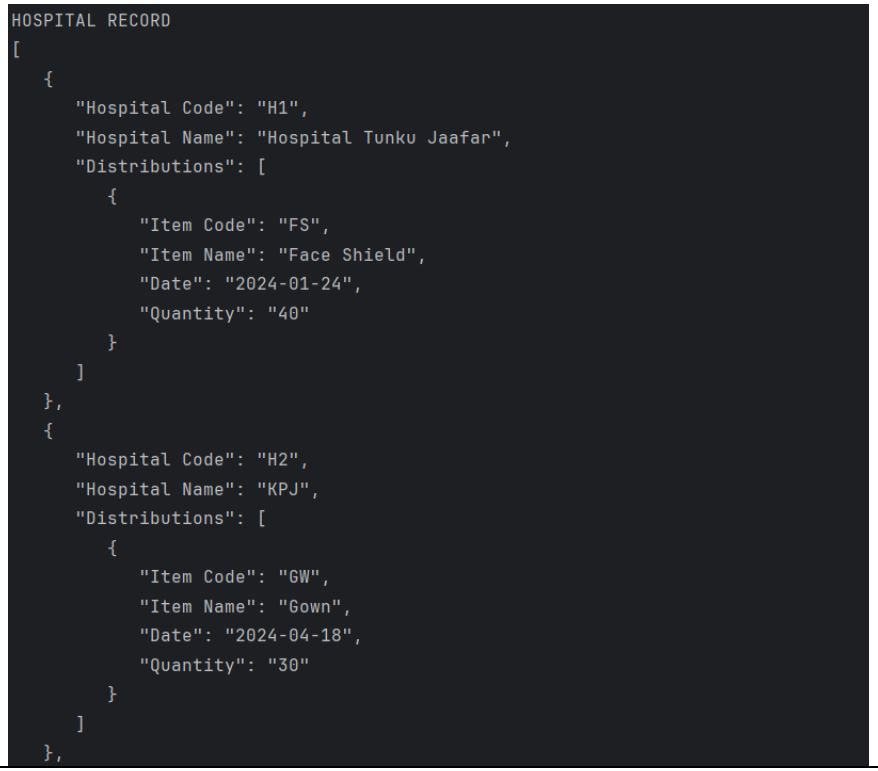

Output:

```
Please enter a valid entry
```

</td>
<td>If the item code is invalid, the program will then repeatedly prompt the user to enter a valid item code until successful</td>
</tr>
</table>

Report Functionality	**SUPPLIER REPORT** **Input:**  **Output:** 	If user inputs 1, the output will show all the items supplied by each supplier alongside with its details

Report Functionality	"Supplier Code": "S2",
	"Supplier Name": "PledisCom",
	"Items": [
	{
	"Item Code": "MS",
	"Item Name": "Mask"
	}
	]
	},
	{
	"Supplier Code": "S3",
	"Supplier Name": "Tokomania",
	"Items": [
	{
	"Item Code": "GL",
	"Item Name": "Gloves"
	},
	{
	"Item Code": "SC",
	"Item Name": "Shoe Covers"
	}
	]
	},
	{
	"Supplier Code": "S4",
	"Supplier Name": "McKesson Corporation",
	"Items": [
	{
	"Item Code": "GW",
	"Item Name": "Gown"
	}
	]
	}
	]		

Report Functionality	**DISTRIBUTION REPORT** **Input:**  **Output:** 	If user inputs 2, a report will be made of all distributions made according to the hospital

<table>
<tr>
<td rowspan="2">Report Functionality</td>
<td>

<pre>
  {
    "Hospital Code": "H3",
    "Hospital Name": "Mawar Hospital",
    "Distributions": [
      {
        "Item Code": "MS",
        "Item Name": "Mask",
        "Date": "2024-06-07",
        "Quantity": "20"
      }
    ]
  }
]
</pre>

</td>
<td></td>
</tr>
<tr>
<td>

OVERALL TRANSACTION
Input:

<pre>
What type of report would you like to generate ?
    1. List of suppliers with their PPE equipment
    2. List of hospitals with quantity of distribution items
    3. Overall transaction report per month
    4. Return to main menu

Please select one of the above by inputting the number [1 | 2 | 3 | 4]:3
Enter the desired month and year (YYYY-MM): 2024-06
</pre>

Output:

<pre>
SUPPLIES RECEIVED:
Item Code: MS, Item Name: Mask, Received: 30, Supplier Code: S2, Supplier Name: PledisCom

ITEMS DISTRIBUTED:
Item Name: Mask, Item Code: MS, Distributed: 20, Hospital Code: H3, Hospital Name: Mawar Hospital
</pre>

</td>
<td>If user inputs 3, they are them required to input a valid date (YYYY-MM). The output will then show all the items received and distributed in said month alongside with its details</td>
</tr>
</table>

<table>
<tr>
<td rowspan="2">Report Functionality</td>
<td>IVNALID REPORT ENTRY
Input:


```
What type of report would you like to generate ?
    1. List of suppliers with their PPE equipment
    2. List of hospitals with quantity of distribution items
    3. Overall transaction report per month
    4. Return to main menu

Please select one of the above by inputting the number [1 | 2 | 3 | 4]:5
```

Output:


```
Please enter a valid entry

Please select one of the above by inputting the number [1 | 2 | 3]:
```

</td>
<td>If user inputs an invalid number, they will then be prompted to enter a valid entry until they successfully do so</td>
</tr>
<tr>
<td>INVALID DATE
Input:


```
What type of report would you like to generate ?
    1. List of suppliers with their PPE equipment
    2. List of hospitals with quantity of distribution items
    3. Overall transaction report per month
    4. Return to main menu

Please select one of the above by inputting the number [1 | 2 | 3 | 4]:3
Enter the desired month and year (YYYY-MM): 22
```

Output:


```
Invalid month format. Please use YYYY-MM.
```

</td>
<td>If user inputs an invalid date, they will then be prompted to enter a valid entry until they successfully do so</td>
</tr>
</table>

3.0 Text Files

Name	Text File
ppe.txt	ppe.txt File Edit View HC,Head Cover,S1,100 FS,Face Shield,S1,100 MS,Mask,S2,100 GL,Gloves,S3,100 GW,Gown,S4,100 SC,Shoe Covers,S3,100
supplier.txt	supplier.txt File Edit View S1,Hybe Co.,0127689345,HybeCo@gmail.com S2,PledisCom,0176800934,pledisOfficial@gmail.com S3,Tokomania,01476988823,tokomaniacompany@gmail.com S4,McKesson Corporation,0152156788,McKessonOfficialCompany@gmail.com
hospitals.txt	hospitals.txt File Edit View H1,Hospital Tunku Jaafar,06777892,HTJ@gmail.com H2,KPJ,064525363,KPJHospital@gmail.com H3,Mawar Hospital,06787898,MawarHospital@gmail.com
distributions.txt	distribution.txt File Edit View 2024-01-24,FS,Face Shield,H1,40 2024-04-18,GW,Gown,H2,30 2024-06-07,MS,Mask,H3,20 2024-06-09,MS,Mask,H3,30 2024-06-09,MS,Mask,H1,20
receive.txt	receive.txt File Edit View 2024-02-12,FS,Face Shield,S1,60 2024-03-06,GW,Gown,S4,70 2024-06-04,MS,Mask,S2,30

4.0 Conclusion

To conclude, our program is successful in meeting the objectives laid out by us in Part 1 which were:

i) To clearly store and list all PPEs in an organized manner
ii) To reduce the manual labour needed managing inventory by creating a completely digital inventory management system
iii) To create a user-friendly system that can be easily understood by all users
iv) To provide an error-free system that negates the possibility of any inventory mishaps as well as maximizing inventory productivity

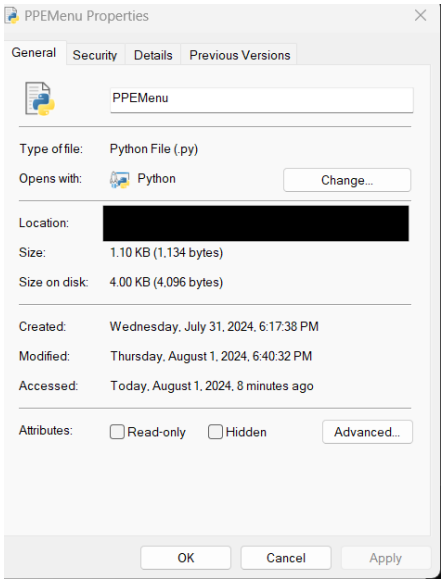
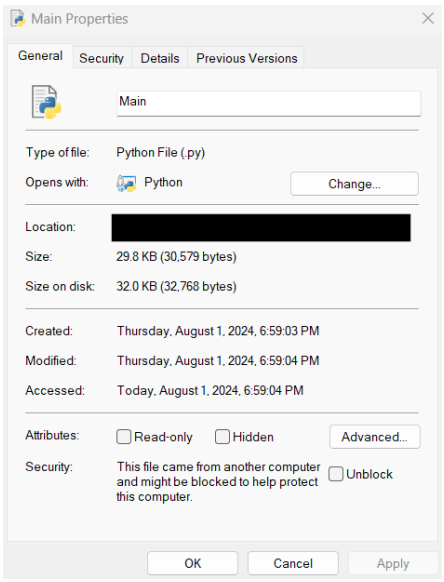
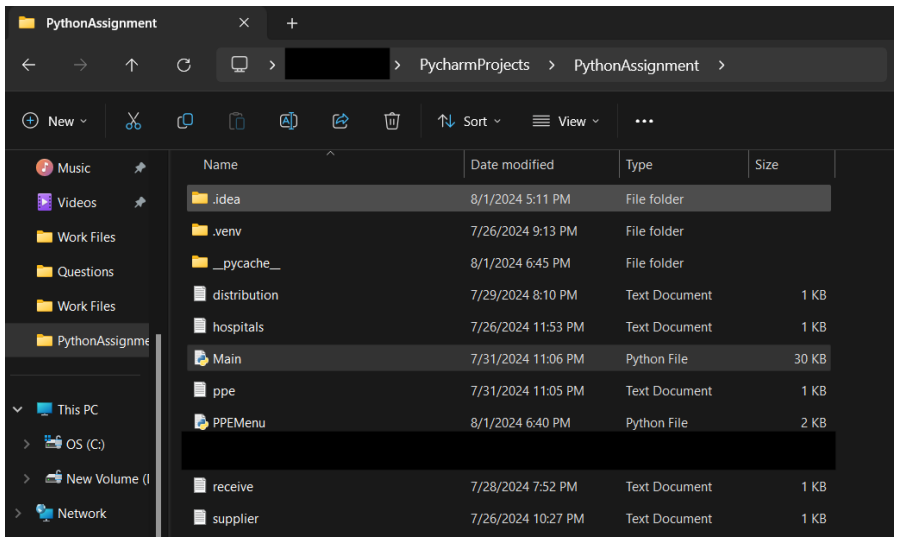
We were able to achieve these goals through the integration of several key features into our program such as the Item Inventory Creation, Item Inventory Update, Item Inventory Tracking, Item Searching & Report; all of which was in service of creating a seamless and intuitive experience to allow for maximum optimisation for inventory management. Beyond that, rigorous testing and debugging was done to the code to ensure an optimised and bug-free system.

While our program was successful in meeting the objectives, it is still held back by some limitations. For example, our program does not allow the customisation of inventory controllers profile. To elaborate, inventory controllers are strictly confined to the username & password initially set preventing them from changing or resetting the data if they so desire. Besides that, the deletion of old inventory controller profiles and the subsequent addition of new ones are not allowed meaning that only 4 inventory controllers may be registered at once which could dampen overall work optimisation. Remedying this could prove favourable as not only is the solution simple (solution only involves basic list manipulation) but also beneficiary in improving user experience.

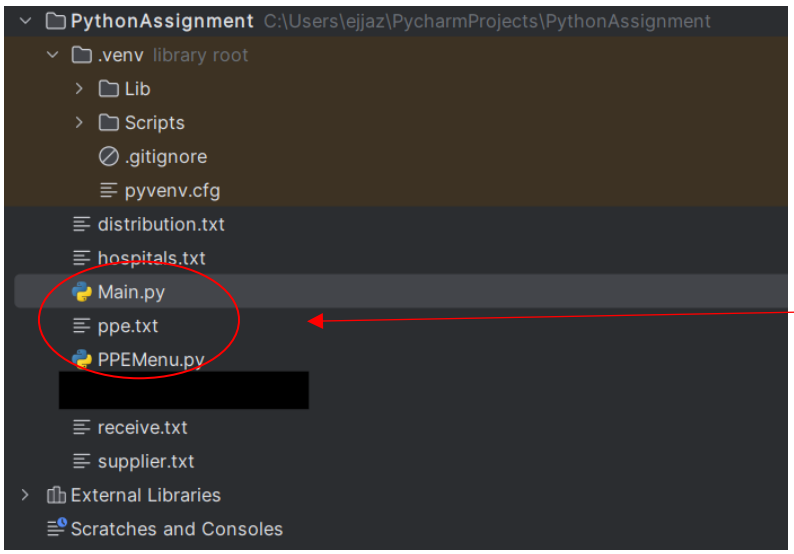
Furthermore, another setback seen in our program is the inability to update supplier information unlike the update feature seen for the hospital information. This is mainly due to the fact that supplier codes are set during the initial creation of the ppe.txt file which locks out the ability to solely update the supplier.txt as to prevent data inconsistency. This can be improved by ensuring that updates made to supplier.txt is mirrored in ppe.txt as well. Finally, our program is also held back by the fact that the data is the text files are not classified which may cause confusion. This can be emended by labelling the data to improve readability.

Appendix A

Setup Guide

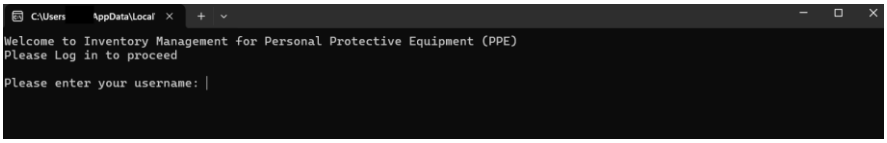
1.	**Please ensure that both PPEMenu.py & Main.py is installed**  
2.	**Please ensure that the names of both files are unchanged (The name of the files should be PPEMenu.py / Main.py**
3.	**Ensure that both .py files are in the same folder** [It should look like this in File Explorer] 

[It should look like this in PyCharm]

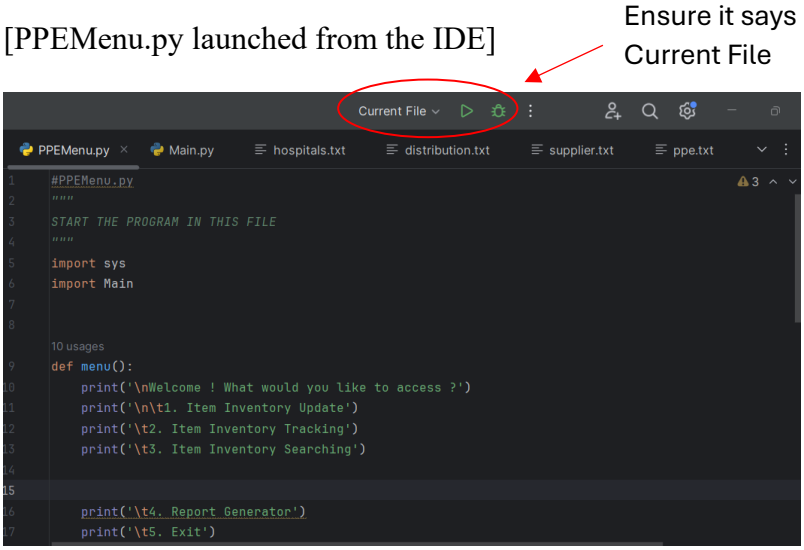


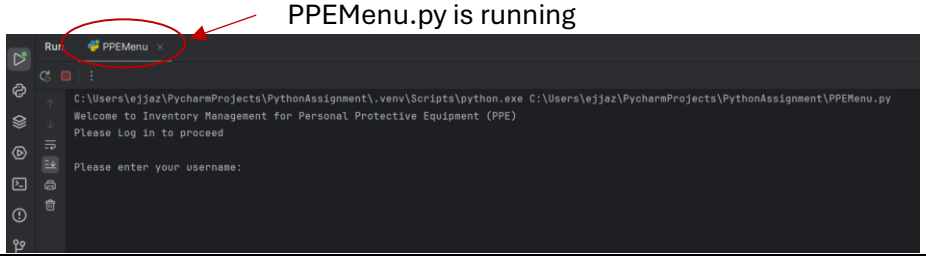
4. **Launch the PPEMenu.py file directly or from the IDE**

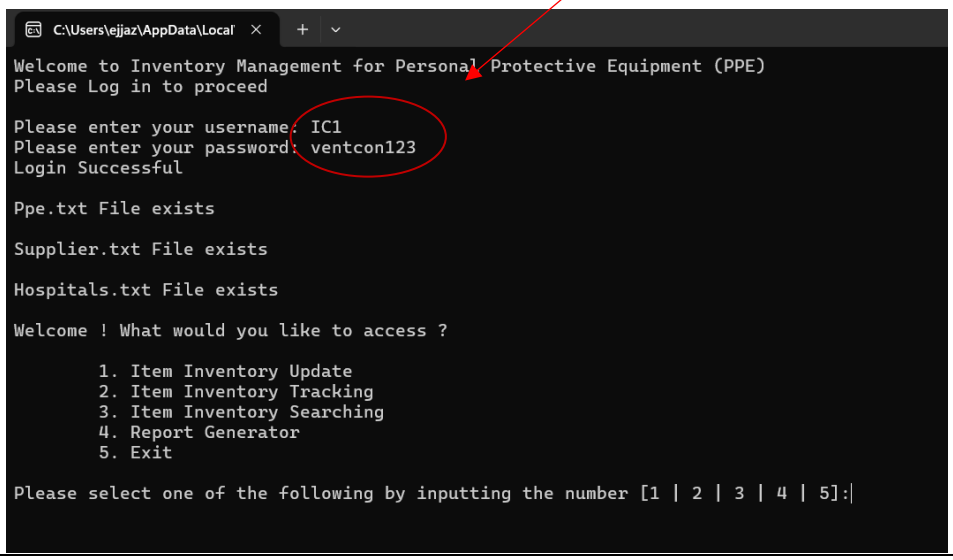
[PPEMenu.py launched directly from file]



[PPEMenu.py launched from the IDE]





5.	**Insert the matching pair of username & passwords to access** Username = [‘IC1’, ‘IC2’, ‘IC3’, ‘IC4’] Passwords = [‘ventcon123’, ‘ventcon456’, ‘ventcon789’, ‘ventcon321’] [Example] 
6.	**Congratulations! You now have access to our program**