PWP GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

AAPP010-4-2-PWP

PROGRAMMING WITH PYTHON

UCDF2109ICT(SE)

LECTURER NAME : DR.MASRINA AKMAL BINTI SALLEH

HAND OUT DATE : 1 SEPTEMBER 2022

HAND IN DATE : 30 OCTOBER 2022

WEIGHTAGE : 50%

GROUP MEMBER	TP NUMBER
GAN MING HUI	TP065539
HO FENG SHENG	TP063370
DYANIEL CHING CHEE XIONG	TP065406

Table of Contents

Table of Figures

1. 0 Introduction – Dyaniel Ching Chee Xiong

Nowadays, **Covid-19** is become more serious in Malaysia. Prevention is better than cure. As the only way to prevent all Malaysian from having Covid-19, **getting vaccine** has become the most important topic for Malaysia Government.

In order to manage, record, make statistics about the total number of Malaysian who have been vaccinated, APU (Asia Pacific University) is asked to write a **computer program** for the vaccination management purpose.

Malaysia Vaccination System is a **vaccination record management system** that is written by Python programming language. It is mainly used for recording all the vaccine details of patients such as vaccine code, dosage number and the date for dose 1 and dose 2.

2.0 Assumptions – Gan Ming Hui

The created program is about the **COVID-19 vaccination record management system**. There are two versions of this program which means it was created to be used by two types of users. The first one is the **patient**, where the patient refers to anyone who is going to receive the vaccine. And the second is **admin**, which refers to the person who will perform the administration of this vaccine record management system. The programs used by these two types of users will be slightly different. To ensure that the user has a good user experience, the program is a **user-friendly program**, and any important and helpful information is provided to the user when they use the program. In addition to this, the programmers have **validated** the program in order to ensure that no errors occur and that the information provided by the user is logica. For example, validate the user's name, account password, vaccine information, etc.

Figure 1: Main Menu for the Vaccination System

As shown in figure1 after executing the program, the user can see the main menu. They can choose to enter either as a patient or as an admin.

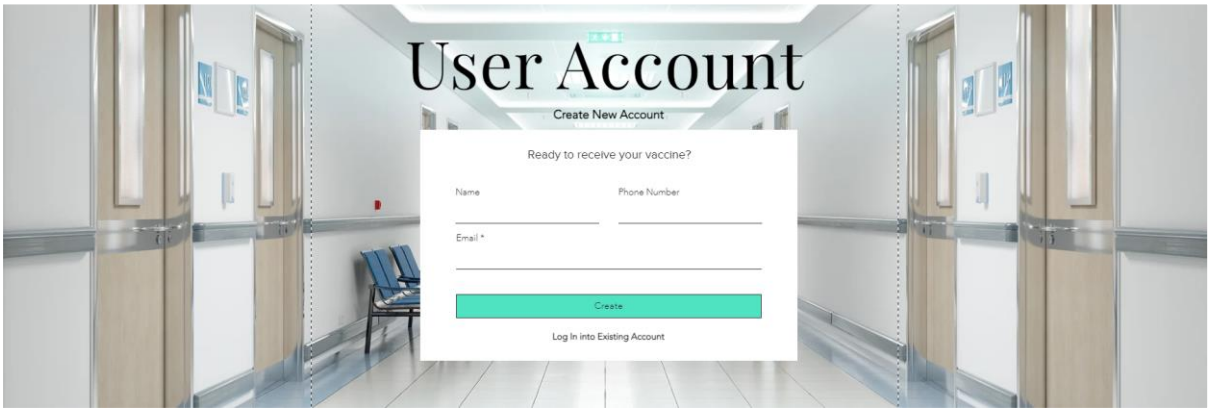


Figure 2: User Registration

As shown in figure 2 if a user chooses to enter as a patient, they must first register to have an account, during this registration they will need to fill out their account information such as name, phone number, address, etc. After they have filled in their personal information and selected the vaccine, they will be advised of a date to receive the vaccine at the vaccine center.

However, if the users want to receive the vaccine earlier or late, they are allowed to do so, they just need to walk in to the vaccine center. After the registration is done, a patient id will also be assigned to the patient.

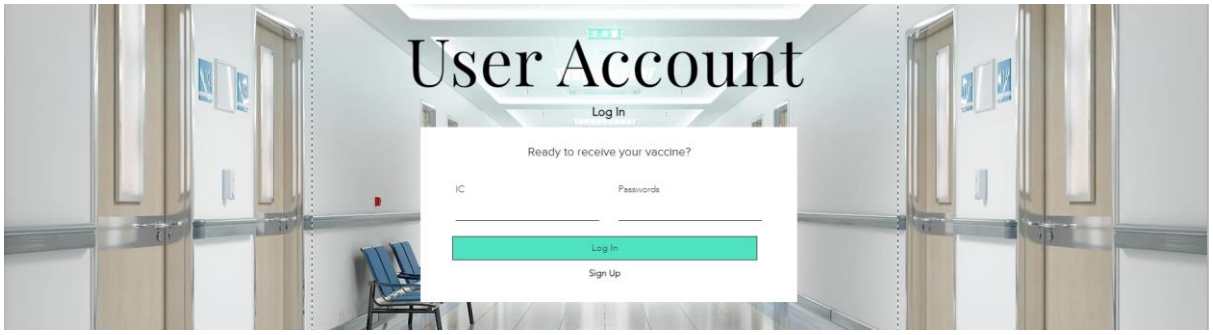


Figure 3: Login into Existing Account

As shown in figure 3 once they have registered, they can start logging into their account. At the login phase, the users need to enter their identification card (IC) and the password they set during registration.

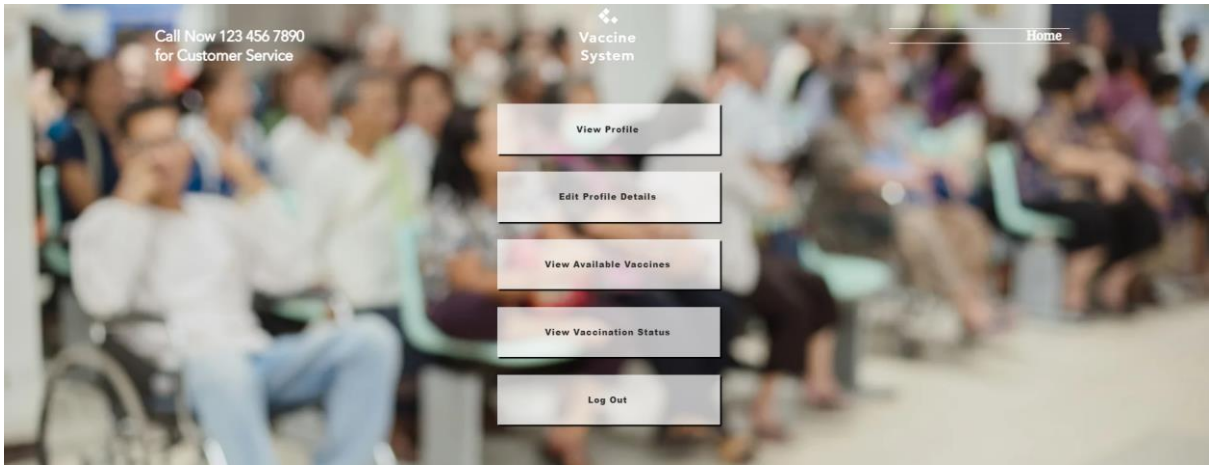


Figure 4: Functions inside the User Account

As shown in figure 4 after successfully logging in, they will be presented with a menu that allows them to select which functions they want to use. '**View Profile'** is where users can see their personal information, which is all of the information they filled out during registration, and if they want to change their profile, they can do so by clicking on the second function, **'Edit Profile Details'**. In addition, users can view all the latest vaccines at **'View Available Vaccines'**. When a new vaccine is added or an old one is deleted, it is immediately synchronized. Users can check their vaccination status by clicking on **'View Vaccination Status'**. They will be able to see which vaccination code they have chosen, as well as whether

the first and second doses are complete or incomplete. Also, users can log out of their accounts by clicking the **'Log Out'** button if they want.

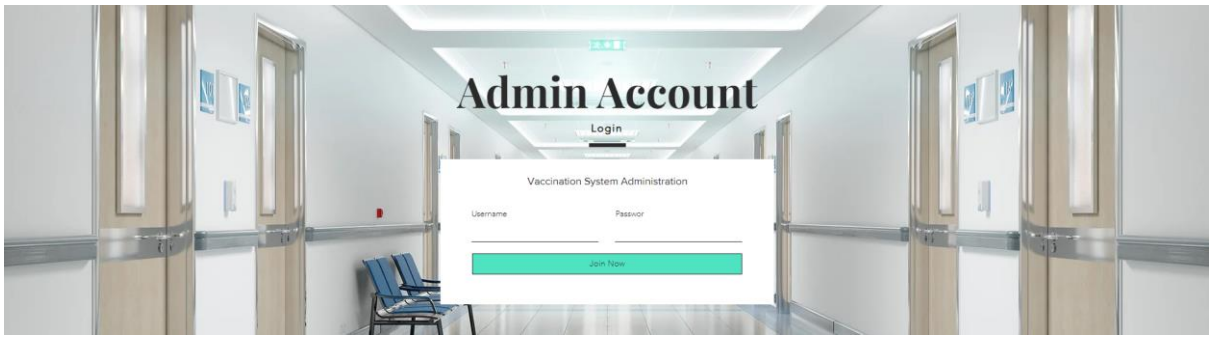


Figure 5: Admin Login

As shown in figure 5 if the users chooses to enter as admin, they will be taken to a login screen. All usernames and passwords of the admin have been entered into the source code. So, except for the accounts that the programmer has entered the source code, all other accounts cannot be logged in successfully.

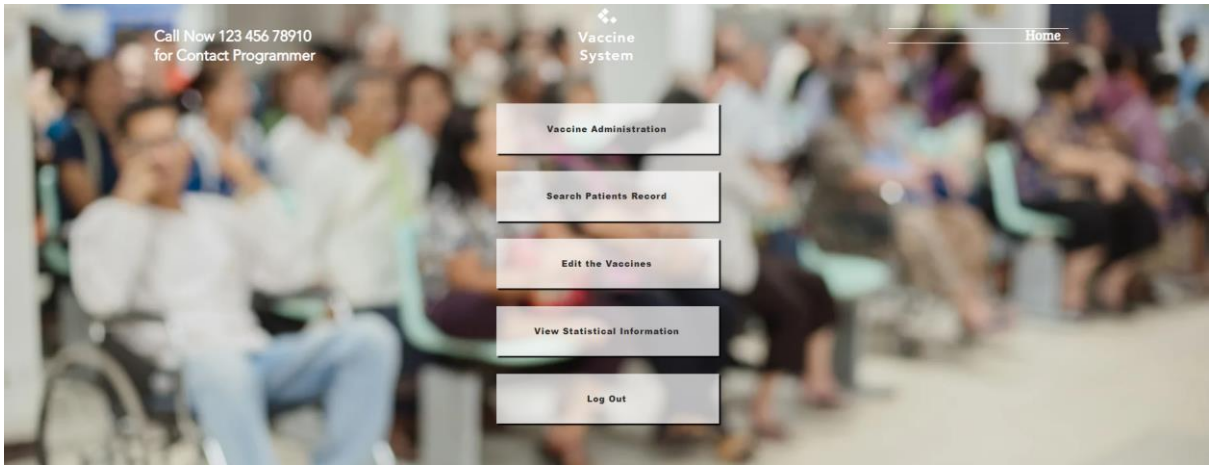


Figure 6: Functions inside the Admin Account

As shown in figure 6 once the administrators have successfully logged in, a menu with five options will be visible to them: 'Vaccine Administrations,' 'Search Patients Record,' 'Edit the Vaccines,' 'View Statistical Information,' and 'Log Out'. For **'Vaccine Administration'**, if a patient comes to the vaccine center to receive a vaccine, the admins will use this function to fill in the details of the vaccine received for that patient. Admin will key in the patient's patient id, the code of the vaccine received, the number of doses and if it is the first dose, the system will also produce the time of the second dose. Furthermore, the program also has a function called **'Search Patients Record'** that enables the admins to look up patients. They have the option of searching for a specific patient or viewing every patient at once. The program also

includes a feature called **'Edit the Vaccines'** that enables the admins to add new vaccines because new vaccines may be invented in the future and remove any vaccines that are no longer in use. Lastly, to make it easier for the admin to do statistics, the admin can use the **'View Statistical Information'** feature. In this function, the admin will be able to know the total number of patients, the patients who have received the vaccine, the patients who are waiting for the second dose, and the patients who have completed both doses for each of VC1 and VC2. The same function same as patients, the admin can log out of the account by clicking the **'Log Out'** button.

3.0 Design of the Program

3.1 Pseudocode

3.1.1 - GAN MING HUI

```
Start
        DEFINE FUNCTION menu()
                Declare selection
                Display 'Malaysia Vaccine System, Welcome Back, Do you want to enter as ?, 1. Patient, 2. Admin, 3.Close'
                Prompt user for a selection
                Read selection
                IF (selection == '1') THEN
                        Call function type_user()
                ELIF (selection == '2') THEN
                        Call function admin_acc
                ELIF (selection == '3') THEN
                        Use 'exit' function to terminate entire program
                ELSE
                        Display 'Error : Invalid Input'
                ENDIF
End
```

Figure 7: Pseudocode menu()

```
Start
    DEFINE FUNCTION type_user()
        Declare user_acc
        DOWHILE (True)
            Display 'User Account,  1. Create New Account, 2.  Log In, 3. Back')
            Prompt user for selection
            Read user_acc
            IF (user_acc == '1') THEN
                Call function new_user()
            ELIF (user_acc == '2') THEN
                Call function exist_user()
            ELIF (user_acc == '3') THEN
                Use return to exit from the function
            ELSE
                Display 'Error : Invalid Input'
            ENDIF
        ENDDO
End
```

Figure 8: Pseudocode type_user()

```
Start
    DEFINE FUNCTION user_menu(patientic)
        Declare opt,
        DOWHILE(True)
            Display '1. View Profile, 2. Edit Profile Details, 3. View Available Vaccines, 4. Veiw Vaccination Status, 5. Log Out')
            Prompt user for selection
            Read opt
            IF (opt == '1') THEN
                Call function user_info_user(patientic)
            ELIF (opt == '2') THEN
                Call function edit_profile_menu(patientic)
            ELIF (opt == '3') THEN
                Call function vaccine_details()
            ELIF (opt == '4') THEN
                Call function vac_status()
            ELIF (opt == '5') THEN
                Display 'You have been successfully logged out.'
                Use break to stop the while loop
            ELSE
                Display 'Invalid Input'
            ENDIF
        ENDDO
End
```

Figure 9: Pseudocode user_menu(patientic)

```
Start
    DEFINE FUNCTION admin_menu()
        Declare opt
        DOWHILE (TRUE)
            Display '1.Vaccine Administration, 2. Search Patient Record, 3. Edit the Vaccines, 4. View Statistical Information, 5. LogOut'
            Prompt user for selection
            Read opt
            IF (opt == '1') THEN
                Call function administration()
            ELIF (opt == '2') THEN
                Call function user_info_admin()
            ELIF (opt == '3') THEN
                Call function vaccine_edit()
            ELIF (opt == '4') THEN
                Call function statistical_report()
            ELIF (opt == '5') THEN
                Display 'You have been successfully logged out.'
                Use break to stop the while loop
            ELSE
                Display 'Invalid Input'
            ENDIF
        ENDDO
End
```

Figure 10: Pseudocode admin_menu()

```
Start
    DEFINE FUNCTION validation_password()
        Declare regex_password, a, b, passw, makesure_digit, makesure_alpha
        Set the pattern of regex_password as minimum 6 six characters and must contain only alpha and numbers
        DOWHILE (True)
            Set a = False, b = False
            Display "Enter '1' to exit"
            Prompt user for password
            Read passw
            IF (passw == '1') THEN
                Return passw and exit from the function
            ENDIF
            IF (passw contains the specified search pattern of regex_password) THEN
                makesure_digit = Result of checking if there are any numbers in the passw
                makesure_alpha = Result of checking if there are any alpha in the passw
                IF (makesure_digit == True) THEN
                    a = True
                ELSE
                    Display 'Error : Must contain digit'
                ENDIF

                IF (makesure_alpha == True) THEN
                    b = True
                ELSE
                    Display 'Error : Must contain alpha'
                ENDIF
                IF ((a == True) and (b == True)) THEN
                    Return passw and exit from the function
                ENDIF
            ELSE
                Display 'Error : minimum 6 characters'
            ENDIF
        ENDDO
End
```

Figure 11: Pseudocode validation_password()

```
Start
    DEFINE FUNCTION validation_ic()
        Declare regex_ic, ic
        Set the pattern of regex_ic as a Malaysia IC
        DOWHILE (True)
            Display "Enter '1' to exit"
            Prompt user for identity card
            Read ic
            IF (ic == '1') THEN
                Return ic and exit from the function
            ENDIF
            IF (length of (ic) == 14) THEN
                IF (ic contains the specified search pattern of regex_ic) THEN
                    Return ic and exit from the function
                ELSE
                    Display 'Error : Invalid IC'
                ENDIF
            ELSE
                Display 'Error : Invalid IC'
            ENDIF
        ENDDO
End
```

Figure 12: Pseudocode validation_ic()

```
Start
    DEFINE FUNCTION validation_postcode()
        Declare postcode, check
        DOWHILE (True)
            Display "Enter '1' to exit"
            Prompt user for postcode
            Read postcode
            check = Result of checking if all the characters in the postcode are numeric
            IF (postcode == '1') THEN
                Return postcode and exit from the function
            ENDIF
            IF ((check == True) and (length of (postcode) == 5)) THEN
                Return postcode and exit from the function
            ELSE
                Display 'Error : Invalid Postcode'
            ENDIF
        ENDDO
End
```

Figure 13: Pseudocode validation_postcode()

```
Start
    DEFINE FUNCTION validation_email()
        Declare regex_email, email
        Set the pattern of regex_email as a valid email
        DOWHILE (True)
            Display "Enter '1' to exit"
            Prompt user for email
            Read email
            IF (email == '1') THEN
                Return email and exit from the function
            ENDIF
            IF (email contains the specified search pattern of regex_email) THEN
                Return email and exit from the function
            ELSE
                Display 'Error : Invalid email'
            ENDIF
        ENDDO
End
```

Figure 14: Pseudocode validation_email()

```
Start
    DEFINE FUNCTION validation_phone()
        Declare regex_phone, phone_num
        Set the pattern of regex_phone as Malaysia Phone Number
        DOWHILE (True)
            Display "Enter '1' to exit"
            Prompt user for phone number
            Read phone_num
            IF (phone_num == '1') THEN
                Return phone_num and exit from the function
            ENDIF
            IF (phone_num contains the specified search pattern of regex_phone) THEN
                Return phone_num and exit from the function
            ELSE
                Display 'Error : Invalid Phone Number, Please refer to the format'
            ENDIF
        ENDDO
End
```

Figure 15: Pseudocode validation_phone()

```
Start
    DEFINE FUNCTION id(vc)
        Declare vc, num1, pat_id, num2
        IF (vc == 'VC1') THEN
            Display 'Patient ID of the patient is VC1- ', num1, 'You can start to login into your account'
            pat_id = 'VC1-', num1
            Return pat_id and exit from the function
        ELIF (vc == 'VC2') THEN
            Display 'Patient ID of the patient is VC2- ', num2, 'You can start to login into your account'
            pat_id = 'VC2-', num2
            Return pat_id and exit from the function
        ENDIF
End
```

Figure 16: Pseudocode id(vc)

```
Start
    DEFINE FUNCTION line(vc)
        Declare vc, f, count1, line
        Open the patients text file in read mode
        Set count1 = 101
        LOOP record line in patients text file
            IF (record line starts with (vc)) THEN
                count = count + 1
            ENDIF
        ENDLOOP
        Close patients text file
        Return count1 and exit from the function
End
```

Figure 17: Pseudocode line(vc)

```
Start
    DEFINE FUNCTION name_list()
        Declare f, a, patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine
        Open the patients text file in read mode
        Display 'Patients List'
        Set a = 1
        LOOP record line in patients text file
            Split the record in line by ',' into patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine
            Display name, patientid
            a = a + 1
        ENDLOOP
        Close vaccine text file
End
```

Figure 18: Pseudocode name_list()

```
Start
    DEFINE FUNCTION statistical_report()
        Declare flag, opt, selection, vc
        Set flag = True
        DOWHILE (flag == True)
            Display 'Statistical Information, 1. View Statistical Report, 2. Back'
            Prompt user for a selection
            Read opt
            DOWHILE (True)
                IF (opt == '1') THEN
                    Display 'View Statistical Report, 1. Statistics VC1, 2. Statistics VC2, 3. Back'
                    Prompt user for a selection
                    Read selection
                    IF (selection == '1') THEN
                        Set vc = 'VC1'
                        Call function vc_menu(vc)
                    ELIF (selection == '2') THEN
                        Set vc = 'VC2'
                        Call function vc_menu(vc)
                    ELIF (selection == '3') THEN
                        Use break to stop the while loop
                    ELSE
                        Display 'Error : Invalid Input'
                    ENDIF

                ELIF (opt == '2') THEN
                    Set flag = False
                    Use break to stop the while loop
                ELSE
                    Display 'Error : Invalid Input'
                    Use break to stop the while loop
                ENDIF
            ENDDO
        ENDDO
End
```

Figure 19: Pseudocode statistical_report

```
Start
    DEFINE FUNCTION vc_menu(vc)
        Declare vc
        IF (vc == 'VC1') THEN
            Display 'VC1 Statistics Report'
            Call function total_patients(vc), vaccinated(vc), wait_dose2(vc), completed(vc)
        ELIF (vc == 'VC2') THEN
            Display 'VC2 Statistics Report'
            Call function total_patients(vc), vaccinated(vc), wait_dose2(vc), completed(vc)
        ENDIF
End
```

Figure 20: Pseudocode vc_menu(vc)

```
Start
        DEFINE FUNCTION total_patients(vc)
                Declare vc, count, f, line
                Set count = 0
                Open the patients text file in read mode
                LOOP record line in patients text file
                        IF (the record line starts with vc) THEN
                                count = count + 1
                        ENDIF
                ENDLOOP
                Display 'Total Number of Patients : ', count
                Close patients text file
End
```

Figure 21: Pseudocode total_patients(vc)

```
Start
        DEFINE FUNCTION vaccinated(vc)
                Declare vc, count, f, line
                Set count = 0
                Open the vaccine text file in read mode
                LOOP record line in vaccine text file
                        IF ((the record line starts with vc) AND ('dose' is in the record line)) THEN
                                count = count + 1
                        ENDIF
                ENDLOOP
                Display 'Total Number of Vaccinated Patients : ', count
                Close vaccine text file
End
```

Figure 22: Pseudocode vaccinated(vc)

```
Start
        DEFINE FUNCTION wait_dose2(vc)
                Declare vc, count, f, line
                Set count = 0
                Open the vaccine text file in read mode
                LOOP record line in vaccine text file
                        IF ((the record line starts with vc) AND ('dose1' is in the record line)) THEN
                                count = count + 1
                        ENDIF
                ENDLOOP
                Display 'Total Number of Patients who waiting for Dose 2 : ', count
                Close vaccine text file
End
```

Figure 23: Pseudocode wait_dose2(vc)

```
Start
        DEFINE FUNCTION completed(vc)
                Declare vc, count, f, line
                Set count = 0
                Open the vaccine text file in read mode
                LOOP record line in vaccine text file
                        IF ((the record line starts with (vc)) AND ('dose2' is in the record line)) THEN
                                count = count + 1
                        ENDIF
                ENDLOOP
                Display 'Total Number of Patients who have finished receiving all doses : ', count
                Close vaccine text file
End
```

Figure 24: Pseudocode completed(vc)

```
Start
    DEFINE FUNCTION vaccine_edit()
        Declare opt
        DOWHILE (True)
            Call function vaccine_details()
            Display 'Edit Vaccination, 1. Add New Vaccines, 2. Delete Existing Vaccines,3. Back'
            Prompt user for a selection
            Read opt
            IF (opt == '1') THEN
                Call function add_vaccine()
            ELIF (opt == '2') THEN
                Call function delete_vaccine()
            ELIF (opt == '3') THEN
                Use break to stop the while loop
            ELSE
                Display 'Error: Invalid Input'
            ENDIF
        ENDDO
End
```

Figure 25: Pseudocode vaccine_edit()

Pseudocode add_vaccine()

```
Start
    DEFINE FUNCTION add_vaccine()
        Declare flag, vc_code, f, lists, line, line_split, dosage_required, interval, age_group, min_num, max_num, opt
        Set flag = False
        DOWHILE (flag == False)
            Display “Add New Vaccine, Enter ‘exit’ to exit”
            Prompt user for new vaccine code
            Read vc_code
            IF (vc_code == ‘EXIT’) THEN
                Use return to exit from the function
            ENDIF
            Open the vaccine text file in read mode
            Set lists equal to empty list
            LOOP record line in vaccine text file
                Split the record in line by ‘,’
                Append the record into a list
            ENDLOOP
            LOOP the range of list length values
                IF (vc_code in the list) THEN
                    Display ‘Vaccine code exists, Please try again’
                    Call function add_vaccine()
                    Use return to exit from the function
                ENDIF
            ENDLOOP
            Close vaccine text file
            Set dosage_required = Return value from validation_dosage()
            IF (dosage_required == ‘exit) THEN
                Use return to exit from the function
            ENDIF
            Set interval = Return value from validation_interval()
            IF(interval == ‘exit) THEN
                Use return to exit from the function
            ENDIF
            Set age_group = Return value from validation_age(), min_num = index 0 and 1 of age_group, max = From the third index of age_group to the end
            IF (age_group == ‘exit’) THEN
                Use return to exit from the function
            ENDIF
            Open the vaccine text file in append mode
```

```
                    Write vc_code, dosage_required, interval, min_num, max_num into vaccine text file
                    Close vaccine text file
                    DOWHILE (TRUE)
                        Display 'Record added'
                        Prompt user for a selection
                        Read opt
                        IF (opt == 'yes') THEN
                            Use break to stop the while loop
                        ELIF (opt == 'no') THEN
                            Set flag = True
                            Use break to stop the while loop
                        ELSE
                            Display 'Errpr : Invalid Input'
                            Use return to exit from the function
                        ENDIF
                    ENDDO
                ENDDO
END
```

```
Start
    DEFINE FUNCTION validation_dosage()
        Declare regex_dosage, dosage_required
        Set the pattern of regex_dosage as only a number from 1 to 3
        DOWHILE (True)
            Display "Enter 'exit' to exit"
            Prompt user for dosage required
            Read dosage_required
            IF (dosage_required == 'exit') THEN
                Return dosage_required and exit from the function
            IF (the length of dosage_required == 1) THEN
                IF (dosage_required contains the specified search pattern of regex_dosage) THEN
                    Return dosage_required and exit from the function
                ELSE
                    Display 'Error : Invalid dosage required'
                ENDIF
            ELSE
                Display 'Error : Invalid dosage required'
            ENDIF
        ENDDO
End
```

Figure 26: Pseudocode validation_dosage()

```
Start
    DEFINE FUNCTION validation_interval()
        Declare regex_interval, interval
        Set the pattern of regex_interval as only a number from 1 to 4
        DOWHILE (True)
            Display "Enter 'exit' to exit"
            Prompt user for interval
            Read interval
            IF (interval == 'exit') THEN
                Return interval and exit from the function
            IF (the length of interval == 1) THEN
                IF (interval contains the specified search pattern of regex_interval) THEN
                    Return interval and exit from the function
                ELSE
                    Display 'Error : Invalid interval'
                ENDIF
            ELIF (interval == 'no') THEN
                Return interval and exit from the function
            ELSE
                Display 'Error : Invalid interval'
            ENDIF
        ENDDO
End
```

Figure 27: Pseudocode validation_interval()

```
Start
    DEFINE FUNCTION vaccine_details()
        Declare f, n, a, lists, line_strip, line_split
        Display 'Vaccine Details'
        Open the vaccine text file in read mode
        Set n = 0, a = 1, lists equal to empty list
        LOOP record line in vaccine text file
            Split the record in line by ','
            Append the record into a list
            Display the vaccine and its details
            n = n + 1
            a = a + 1
        ENDLOOP
        Close vaccine text file
End
```

Figure 28: Pseudocode vaccine_details()

Pseudocode admin_acc()

```
Start
    DEFINE FUNCTION admin_acc()
    Declare opt, admin_user, admin_pass
        DOWHILE (True)
            Display 'Admin Account, 1.Log in, 2.Back'
            Prompt user for selection
            Read opt
            IF (opt == '1') THEN
                Display "Enter '1' to exit"
                Prompt user for username
                Read admin_user
                IF (admin_user == "1') THEN
                    Use return to exit from the function
                ENDIF
                Display "Enter '1' to exit"
                Prompt user for password
                Read admin_pass
                IF (admin_pass == '1') THEN
                    Use return to exit from the function
                ENDIF
                IF ((admin_user == 'minghui') and (admin_pass == 'tp065539')) THEN
                    Display 'Login Successfully, Welcome Back'
                    Call function admin_menu()
                    Use break to stop the while loop
                ELIF ((admin_user == 'fengsheng') and (admin_pass == 'tp063370')) THEN
                    Display 'Login Successfully, Welcome Back'
                    Call function admin_menu()
                    Use break to stop the while loop
                ELIF ((admin_user == 'dyaniel') and (admin_pass == 'tp065406')) THEN
                    Display 'Login Successfully, Welcome Back'
                    Call function admin_menu()
                    Use break to stop the while loop
                ELSE
```

```
                    Display 'Error : Invalid Account'
                ENDIF

            ELIF (opt == '2') THEN
                Use return to exit from the function
            ELSE
                Display 'Error : Invalid Input'
            ENDIF
        ENDDO
End
```

Pseudocode new_user()

```
Start
    DEFINE FUNCTION new_user()
        Declare name, a, passw, f, flag, ic, lists, line, line_split, gender, address, vc, postcode, age_int, age_str, email, phone_num, illness, is_allergic, vaccine, pat_id
        Display 'Please register your details'
        DOWHILE (True)
            Prompt user for name
            Read name
            a = Result of checking if all the characters in the name are alphaets
            IF (name == '1') THEN
                Use return to exit from the function
            ELIF (a == True) THEN
                Use break to terminate the while loop
            ENDIF
            IF (a == False) THEN
                IF (whitespace is found in the name) THEN
                    Use break to terminate the while loop
                ELSE
                    Display 'Error : Invalid Input'
                ENDIF
            ENDIF
        ENDDO
        passw = Returned value from validation_password()
        IF (passw == '1') THEN
            Use return to exit from the function
        ENDIF
        Open patients text file in read mode
        DOWHILE (True)
            flag = False
            ic = Returned value from validation_ic()
            IF (ic == '1') THEN
                Use return to exit from the function
            ENDIF
            lists equal to an empty list
            LOOP record line in patients text file
                Split the record in line by ','
                Append the record into a list
            ENDLOOP
            Close the patients text file.
```

```
        Open patients text file in read mode
        LOOP the range of list length values
            IF ((ic) in the list) THEN
                flag = True
            ENDIF
        ENDLOOP
        IF (flag == True) THEN
            Display 'Error : IC exists, Please try again'
            Use 'continue' to return the control to the beginning of the while loop.
        ELSE
            Use break to terminate the while loop
        ENDIF
        Close the patients text file
ENDDO
DOWHILE (True)
        Display "Enter '1' to exit"
        Prompt user for gender
        Read gender
        IF ((gender == 'm') or (gender == 'f') or (gender == 'male') or (gender == 'female')) THEN
            Use break to terminate the while loop
        ELIF (gender == '1') THEN
            Use return to exit from the function
        ELSE
            Display 'Error : Invalid Input either male(m) or female(f)
        ENDIF
ENDDO
Display "Enter '1' to exit"
Prompt user for address
Read address
IF (address == '1') THEN
        Use return to exit from the function
postcode = Returnted value from validation_postcode()
IF (postcode == '1') THEN
        Use return to exit from the function
DOWHILE (True)
        Display "Enter '1' to exit"
        Prompt user for vacciene centre
        Read vc
        IF (vc == '1') THEN
```

```
                    Use return to exit from the function
            ENDIF
            IF ((vc == 'VC1') or (vc == 'VC2')) THEN
                    Use break to terminate the while loop
            ELSE
                    Display 'Error : Invalid Input'
            ENDIF
    ENDDO
    DOWHILE (True)
            Try
                    Display "Enter '1' to exit"
                    Prompt user for age
                    Read age_int
                    IF (age_int == 1) Then
                            Use return to exit from the function
                    ENDIF
                    IF ((age_int > 11) and (age_int < 101)) THEN
                            age_str = converted from age_int into the form of a string
                            Use break to terminate the while loop
                    ELSE
                            Display 'Error : Invalid age'
                    ENDIF
            Except
                    Display 'Error : Invalid Input'
    ENDDO
    email = Returned value from validation_email()
    IF (email == '1') THEN
            Use return to terminate the while loop
    ENDIF
    phone_num = Returned value from validation_phone()
    IF (phone_num == '1') THEN
            Use return to terminate the while loop
    ENDIF
    DOWHILE (True)
            Display "Enter '1' to exit"
            Prompt user for illness
            Read illness
            IF (illness == '1') THEN
                    Use return to terminate the while loop
```

```
                ENDIF
                IF ((illness == 'yes') or (illness == 'no')) THEN
                        Use break to terminate the while loop
                ELSE
                        Display 'Error : Invalid Input'
        ENDDO
        DOWHILE (True)
                Display "Enter '1' to exit"
                Prompt user for allergic
                Read is_allergic
                IF (is_allergic == '1') THEN
                        Use return to terminate the while loop
                ENDIF
                IF ((allergic == 'yes') or (allergic == 'no')) THEN
                        Use break to terminate the while loop
                ELSE
                        Display 'Error : Invalid Input'
                ENDIF
        ENDDO
        vaccine = Returned value from vaccine_selection(age_int)
        IF (selection == '1') THEN
                Use return to exit from the while loop
        ENDIF
        pat_id = Returned value from id(vc)
        Open patients text file

        Write pat_id, ic, name, passw, gender, vc,  address, postcode, age_str, email, phone_num, illness, is_allergic, vaccine into patients text file

        Close patients text file
End
```

Pseudocode exist_user()

```
Start
    DEFINE FUNCTION exist_user()
        Declare access1, access2, a, b, ic, passw, i, f, lists, line, line_strip, line_split, again
        DOWHILE (access1 == False)
            Display “Please login to your personal account, Enter ‘1’ to exit”
            Prompt user for IC
            Read ic
            IF (ic == ‘1’) THEN
                Use return to exit from the function
            ENDIF
            Display “Enter ‘1’ to exit’
            Prompt user for password
            Read passw
            IF (passw == ‘1’) THEN
                Use return to exit from the function
            i = 0
            Open patients text file in read mode
            lists1 equal to an empty list
            LOOP record line in patients text file
                Strip the spaces at the beginning and at the end of the record lines
                Split the record in line by ‘,’
                Append the record into a list
                IF (( ic is found in the list) and (passw is found in the list)) THEN
                    Display ‘Login Successfully, Welcome Back’
                    Call function user_menu(ic)
                    access1 = True
                    Use break to terminate the for loop
                ELIF ((ic is not found in the list) or (passw is not found in the list)) THEN
                    i = i + 1
                ELSE
                    access1 = False
                    access2 = False
                ENDIF
```

```
                ENDLOOP
                IF (access1 == False) THEN
                        Display ‘Error : Invalid Account’
                        DOWHILE (access2 == False)
                                Prompt user for confirmation
                                Read again
                                IF (again == ‘yes’) THEN
                                        Use break to terminate the while loop
                                ELIF (again == ‘no’) THEN
                                        Access1 = True
                                        Use break to terminate the while loop
                                ELSE
                                        Display ‘Error : Invalid Input’
                                ENDIF
                        ENDDO
                ENDIF
                Close the patients text file
        ENDDO
End
```

3.1.2 - HO FENG SHENG

Vac_status(patientic)

```
START
    DEFINE function AS vac_status(patientic)
        SET flag = False
        OPEN Patients.txt IN reading mode
        FOR EACH line IN Patients.txt
            IF (patientic) IN line THEN:
                READ patient_id
                SET patient_id = patientid
                STOP LOOPING
            ENDIF
        CLOSE Patients.txt file
        OPEN vaccination.txt IN reading mode
        FOR EACH i IN vaccination.txt
            IF (i start with patient_id) THEN:
                SET flag = True
                IF i [2] == 'AF' OR i[2] == 'BV' OR i[2] =='CZ' OR i[2]
                =='DM' THEN:
                    READ i [2]
                        DISPLAY ("Vaccination Code:" + i[2])
                    IF i [1] == 'dose1' THEN:
                        READ i [4]
                        DISPLAY ("Dose 1 status: Completed")
                        DISPLAY ("Dose 1 completed date:" + i [4])
                        STOP LOOPING
                    ELIF i [1] == 'dose2' THEN:
                        READ i [4]
                        DISPLAY ("Dose 1 status: Completed")
                        DISPLAY ("Dose 1 completed date:" + i [4])
```

```
                    STOP LOOPING
                IF i [3] =='NONE' THEN:
                    READ i [5]
                    DISPLAY ("Dose 2 status: Completed")
                    DISPLAY ("Dose 2 completed date: " + i [5])
                    STOP LOOPING
                ELSE
                    READ i [3]
                    DISPLAY ("Dose 2 status: Incomplete")
                    DISPLAY ("Suggested for Dose 2 date:" + i [3])
                    STOP LOOPING
                ENDIF
                ELIF i [2] == 'EC':
                READ i [2], i [4]
                DISPLAY ("Vaccination Code:" + i [2])
                DISPLAY ("Dose 1 status: Completed")
                DISPLAY ("Dose 1 completed date:" + i [4])
                DISPLAY ("Dose 2 status: Only for one dose")
                STOP LOOPING
            ENDIF
        CLOSE vaccination.txt file
        IF flag == False THEN:
            OPEN Patients.txt file in reading mode
            SET flag2 = False
            FOR EACH i IN Patients.txt
                IF (i start with patient_id) THEN
                    SET flag2 = TRUE
                    DISPLAY ("Vaccination Code:" + vaccine_code)
                    DISPLAY ("Dose 1 status: Incomplete")
                    DISPLAY ("Dose 2 status: Incomplete")
```

STOP LOOPING

IF flag2 == False THEN:

DISPLAY (“Invalid patient id”)

CLOSE Patients.txt file

END

vac_status_admin (patient_id)

```
START
    DEFINE function AS vac_status_admin (patient_id)
        SET flag = False
        OPEN vaccination.txt IN reading mode
        FOR EACH line IN vaccination.txt
            IF (i start with patient_id) THEN:
                SET flag = True
                IF i [2] == 'AF' OR i[2] == 'BV' OR i[2] =='CZ' OR i[2]
                =='DM' THEN:
                        READ i [2]
                        DISPLAY ("Vaccination Code:" + i[2])
                    IF i [1] == 'dose1' THEN:
                        READ i [4]
                        DISPLAY ("Dose 1 status: Completed")
                        DISPLAY ("Dose 1 completed date:" + i [4])
                        STOP LOOPING
                    ELIF i [1] == 'dose2' THEN:
                        READ i [4]
                        DISPLAY ("Dose 1 status: Completed")
                        DISPLAY ("Dose 1 completed date:" + i [4])
                        STOP LOOPING
                    IF i [3] =='NONE' THEN:
                        READ i [5]
                        DISPLAY ("Dose 2 status: Completed")
                        DISPLAY ("Dose 2 completed date: " + i [5])
                        STOP LOOPING
                    ELSE
                        READ i [3]
                        DISPLAY ("Dose 2 status: Incomplete")
```

```
                                    DISPLAY ("Suggested for Dose 2 date:" + i [3])
                                    STOP LOOPING
                                ENDIF

                            ELIF i [2] == 'EC':
                                READ i [2], i [4]
                                DISPLAY ("Vaccination Code:" + i [2])
                                DISPLAY ("Dose 1 status: Completed")
                                DISPLAY ("Dose 1 completed date:" + i [4])
                                DISPLAY ("Dose 2 status: Only for one dose")
                                STOP LOOPING
                            ENDIF
                    CLOSE vaccination.txt file
                    IF flag == False THEN:
                        OPEN Patients.txt file in reading mode
                        SET flag2 = False
                        FOR EACH i IN Patients.txt
                            IF (i start with patient_id) THEN
                                SET flag2 = TRUE
                                READ vaccine_code
                                DISPLAY ("Vaccination Code:" + vaccine_code)
                                DISPLAY ("Dose 1 status: Incomplete")
                                    DISPLAY ("Dose 2 status: Incomplete")
                                STOP LOOPING
                        IF flag2 == False THEN:
                            DISPLAY ("Invalid patient id")
                    CLOSE Patients.txt file
END
```

User_info_admin()

```
START
    DEFINE function AS user_info_admin()
        DOWHILE True
            DISPLAY ("\n--------------------Patient Record--------------------")
            DISPLAY ("\n1.Search a patient\n2.All patients\n3.Back\n")
            PROMPT USER for (“Please Enter the selection: “)
            DECLARE opt
            READ opt
                IF opt ==1 THEN:
                    SET patient_id = one_user_info_admin()
                    IF patient_id ==1 THEN:
                        RETURN
                ELIF opt ==2 THEN:
                    RUN all_user_info_admin()
                ELIF opt ==3 THEN:
                    BREAK
                ELSE:
                    DISPLAY (“Invalid Input”)
                ENDIF
        ENDDO
END
```

One_user_info_admin()

START

DEFINE function AS one_user_info_admin()

OPEN Patients.txt file IN reading mode

DISPLAY ("\n*Enter '1' to exit")

PROMPT USER for ("Please Enter the Patient ID: ") in upper case

DECLARE patient_id

READ patient_id

IF patient_id == 1 THEN:

RETURN patient_id

SET flag = TRUE

FOR EACH line IN Patients.txt

IF patient_id In line THEN:

DECLARE (patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine)

READ (patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine)

DISPLAY ("This is the Personal Details of Patient", patient_id," ")

DISPLAY (": Patient ID\t: ", patientid)

DISPLAY (": IC\t\t: ", ic)

DISPLAY (": Name\t\t: ", name)

DISPLAY (": Gender(m/f)\t: ", gender)

DISPLAY (": VC(vc1/vc2)\t: ", vc)

DISPLAY (": Address\t: ", address)

DISPLAY (": Postcode\t: ", postcode)

DISPLAY (": Age\t\t: ", age)

DISPLAY (": Email\t\t: ", email)

DISPLAY (": Phone Number\t: ", phonenumber)

DISPLAY (": Illness\t: ", illness)

DISPLAY (": Allergic\t: ", allergic)

DISPLAY (": Vaccine\t: ", vaccine)

SET flag = False

```
            ENDIF
        IF flag ==True THEN:
            DISPLAY ("\n*Error: The patient id cannot be found")
        ENDIF
        CLOSE Patients.txt file
END
```

All_user_info_admin()

```
START
    DEFINE function AS all_user_info_admin()
        SET count = 0
        OPEN Patients.txt file in reading mode
        FOR EACH line in Patients.txt
            Count + = 1
            DECLARE Count
            DISPLAY (“This is the Personal Details of Patient", count”)
            DECLARE (patient_id, ic, name, password, gender, vc, address,
            postcode, age, email, phonenumber, illness, allergic, vaccine)
            READ(patient_id, ic, name, password, gender, vc, address, postcode,
            age, email, phonenumber, illness, allergic, vaccine)
            DISPLAY ("This is the Personal Details of Patient", patient_id," ")
            DISPLAY (": Patient ID\t: ", patientid)
            DISPLAY (": IC\t\t: ", ic)
            DISPLAY (": Name\t\t: ", name)
            DISPLAY (": Gender(m/f)\t: ", gender)
            DISPLAY (": VC(vc1/vc2)\t: ", vc)
            DISPLAY (": Address\t: ", address)
            DISPLAY (": Postcode\t: ", postcode)
            DISPLAY (": Age\t\t: ", age)
            DISPLAY (": Email\t\t: ", email)
            DISPLAY (": Phone Number\t: ", phonenumber)
            DISPLAY (": Illness\t: ", illness)
            DISPLAY (": Allergic\t: ", allergic)
            DISPLAY (": Vaccine\t: ", vaccine)
            RUN vac_status_admin(patient_id)
        CLOSE Patients.txt file
END
```

User_info_user(patient_id)

START

DEFINE function AS user_info_user(patient_id)

OPEN Patients.txt file in reading mode

FOR EACH line In Patients.txt

IF patient_id IN line THEN:

DECLARE (patient_id, ic, name, password, gender,

vc, address, postcode, age, email, phonenumber, illness,

allergic, vaccine)

READ(patient_id, ic, name, password, gender, vc, address,

postcode, age, email, phonenumber, illness, allergic, vaccine)

DISPLAY ("This is the Personal Details of Patient", patient_id," ")

DISPLAY (": Patient ID\t: ", patientid)

DISPLAY (": IC\t\t: ", ic)

DISPLAY (": Name\t\t: ", name)

DISPLAY (": Gender(m/f)\t: ", gender)

DISPLAY (": VC(vc1/vc2)\t: ", vc)

DISPLAY (": Address\t: ", address)

DISPLAY (": Postcode\t: ", postcode)

DISPLAY (": Age\t\t: ", age)

DISPLAY (": Email\t\t: ", email)

DISPLAY (": Phone Number\t: ", phonenumber)

DISPLAY (": Illness\t: ", illness)

DISPLAY (": Allergic\t: ", allergic)

DISPLAY (": Vaccine\t: ", vaccine)

ENDIF

CLOSE Patients.txt file

END

Edit_profile(patientic)

```
START
    DEFINE function AS edit_profile(patientic)
        DECLARE opt
        DOWHILE True
            DISPLAY ("\n--------------------Edit Profile--------------------")
            DISPLAY ("\n1.Password\n2.Address & Postcode\n3.Email\n4.
            Phone Number\n5.Vaccine\n6.Back\n")
            PROMPT USER for ("Please Enter the selection: ")
            IF opt == ‘1’ THEN:
                RUN edit_password(patientic)
            ELIF opt ==’2’ THEN
                RUN edit_address_postcode(patientic)
            ELIF opt == ‘3’ THEN
                RUN edit_email(patientic)
            ELIF opt == ‘4’ THEN
                RUN edit_phone_number(patientic)
            ELIF opt ==’5’ THEN
                RUN edit_vaccine(patientic)
            ELIF opt == ‘6’ THEN
                Break the loop
            ELSE:
                DISPLAY (“Invalid Input”)

            ENDIF
        ENDDO
END
```

Edit_phone_number(patientic)

START

DEFINE function AS edit_phone_number(patientic)

DECLARE line_number, access, f, lists, new_phone_number, makesure

SET line_number = 0

SET access = False

OPEN Patients.txt file in read mode

SET lists to empty list

FOR EACH line in Patients.txt file

Split the lines in Patients.txt file by “,”

Append the record into a list

DOWHILE access == False

IF (lists[line_number][1] == patientic) THEN:

SET patientid, ic, name, password, gender, vc, address, postcode, age, email, old_phone_number, illness, allergic, vaccine into lists[line_number]

SET access = True

ELSE:

Line_number += 1

ENDIF

CLOSE Patients.txt file

ENDO

DISPLAY ("\n----------Edit Phone Number----------\n")

PROMPT USER for ("Please Enter new phone number: ")

PROMPT USER for ("Do you really want to edit(yes/no)? \n> ") in lower case

DOWHILE True

IF makesure ==”yes” THEN:

DISPLAY ("\n----------Edited Successfully----------")

DISPLAY ("\n* Phone Number has been edited from",

old_phone_number,"to",new_phone_number,"*")

```
                    OPEN Patients.txt file in append mode
                    Write lines into Patient.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in read mode
                    READ lines in Patients.txt file
                    DELETE the latest lines in the Patients.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in writing mode
                    FOR EACH line in Patients.txt file
                        Write lines into the Patients.txt file
                    CLOSE Patients.txt file
                    Break the loop
                ELIF makesure ==”no”
                    DISPLAY ("*Nothing has changed*")
                    Break the loop

                ELSE:
                    DISPLAY (“Invalid Input”)
                ENDIF
            ENDDO
END
```

Edit_email(patientic)

START

DEFINE function AS edit_email(patientic)

DECLARE line_number, access, f, lists, new_email, makesure

SET line_number = 0

SET access = False

OPEN Patients.txt file in read mode

SET lists to empty list

FOR EACH line in Patients.txt file

Split the lines in Patients.txt file by “,”

Append the record to a list

DOWHILE access == False

IF (lists[line_number][1] == patientic) THEN:

SET patientid, ic, name, password, gender, vc, address,

postcode, age, email, old_phone_number, illness, allergic,

vaccine into lists[line_number]

SET access = True

ELSE:

Line_number += 1

ENDIF

CLOSE Patients.txt file

ENDO

DISPLAY ("\n----------Edit Email----------\n")

PROMPT USER for ("Please Enter new email: ")

PROMPT USER for ("Do you really want to edit(yes/no)? \n> ") in lower case

DOWHILE True

IF makesure ==”yes” THEN:

DISPLAY ("\n----------Edited Successfully----------")

DISPLAY ("\n* Email has been edited from",

```
                    old_email,"to",new_email,"*")



                    OPEN Patients.txt file in append mode
                    Write lines into Patient.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in read mode
                    READ lines in Patients.txt file
                    DELETE the latest lines in the Patients.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in writing mode
                    FOR EACH line in Patients.txt file
                        Write lines into the Patients.txt file
                    CLOSE Patients.txt file
                    Break the loop
                ELIF makesure ==”no”
                    DISPLAY ("*Nothing has changed*")
                    Break the loop

                ELSE:
                    DISPLAY (“Invalid Input”)
                ENDIF

            ENDDO
END
```

Edit_address_postcode(patientic)

START

DEFINE function AS edit_address_postcode(patientic)

DECLARE line_number, access, f, lists, new_address,new_postcode, makesure

SET line_number = 0

SET access = False

OPEN Patients.txt file in read mode

SET lists to empty list

FOR EACH line in Patients.txt file

Split the lines in Patients.txt file by “,”

Append the record to a list

DOWHILE access == False

IF (lists[line_number][1] == patientic) THEN:

SET patientid, ic, name, password, gender, vc, address,

postcode, age, email, old_phone_number, illness, allergic,

vaccine into lists[line_number]

SET access = True

ELSE:

Line_number += 1

ENDIF

CLOSE Patients.txt file

ENDO

DISPLAY ("\n----------Edit Address & Postcode----------\n")

PROMPT USER for ("Please Enter new address: ")

PROMPT USER for ("Please Enter new postcode: ")

PROMPT USER for ("Do you really want to edit(yes/no)? \n> ") in lower case

DOWHILE True

IF makesure ==”yes” THEN:

DISPLAY ("\n----------Edited Successfully----------")

```
                DISPLAY ("\n* Address and Postcode has been edited from",
                old_address,"to",old_postcode,“&",new_address,"to",new_post
                code,"*")

                OPEN Patients.txt file in append mode
                Write lines into Patient.txt file
                CLOSE Patients.txt file

                OPEN Patients.txt file in read mode
                READ lines in Patients.txt file
                DELETE the latest lines in the Patients.txt file
                CLOSE Patients.txt file

                OPEN Patients.txt file in writing mode
                FOR EACH line in Patients.txt file
                        Write lines into the Patients.txt file
                CLOSE Patients.txt file
                Break the loop
        ELIF makesure ==”no”
                DISPLAY ("*Nothing has changed*")
                Break the loop

        ELSE:
                DISPLAY (“Invalid Input”)
        ENDIF

    ENDDO
END
```

Edit_password(patientic)

START

DEFINE function AS edit_password(patientic)

DECLARE line_number, access, f, lists, new_password, makesure

SET line_number = 0

SET access = False

OPEN Patients.txt file in read mode

SET lists to empty list

FOR EACH line in Patients.txt file

Split the lines in Patients.txt file by “,”

Append the record to a list

DOWHILE access == False

IF (lists[line_number][1] == patientic) THEN:

SET patientid, ic, name, password, gender, vc, address,

postcode, age, email, old_phone_number, illness, allergic,

vaccine into lists[line_number]

SET access = True

ELSE:

Line_number += 1

ENDIF

CLOSE Patients.txt file

ENDO

DISPLAY ("\n----------Edit Password----------\n")

PROMPT USER for ("Please Enter new password: ")

PROMPT USER for ("Do you really want to edit(yes/no)? \n> ") in lower case

DOWHILE True

IF makesure ==”yes” THEN:

DISPLAY ("\n----------Edited Successfully----------")

DISPLAY ("\n* Password has been edited from",

```
                    old_password,"to",new_password,"*")

                    OPEN Patients.txt file in append mode
                    Write lines into Patient.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in read mode
                    READ lines in Patients.txt file
                    DELETE the latest lines in the Patients.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in writing mode
                    FOR EACH line in Patients.txt file
                        Write lines into the Patients.txt file
                    CLOSE Patients.txt file
                    Break the loop
                ELIF makesure ==”no”
                    DISPLAY ("*Nothing has changed*")
                    Break the loop

                ELSE:
                    DISPLAY (“Invalid Input”)
                ENDIF

            ENDDO
END
```

Edit_vaccine(patientic)

START

DEFINE function AS edit_vaccine(patientic)

DECLARE line_number, access, f, lists, new_vaccine, makesure

SET line_number = 0

SET access = False

OPEN Patients.txt file in read mode

SET lists to empty list

FOR EACH line in Patients.txt file

Split the lines in Patients.txt file by “,”

Append the record to a list

DOWHILE access == False

IF (lists[line_number][1] == patientic) THEN:

SET patientid, ic, name, password, gender, vc, address,

postcode, age, email, old_phone_number, illness, allergic,

vaccine into lists[line_number]

SET access = True

ELSE:

Line_number += 1

ENDIF

CLOSE Patients.txt file

ENDO

DISPLAY ("\n----------Edit Email----------\n")

PROMPT USER for ("Please Enter new vaccine: ")

PROMPT USER for ("Do you really want to edit(yes/no)? \n> ") in lower case

DOWHILE True

IF makesure ==”yes” THEN:

DISPLAY ("\n----------Edited Successfully----------")

DISPLAY ("\n* Vaccine has been edited from",

```
                    old_vaccine,"to",new_vaccine,"*")

                    OPEN Patients.txt file in append mode
                    Write lines into Patient.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in read mode
                    READ lines in Patients.txt file
                    DELETE the latest lines in the Patients.txt file
                    CLOSE Patients.txt file

                    OPEN Patients.txt file in writing mode
                    FOR EACH line in Patients.txt file
                        Write lines into the Patients.txt file
                    CLOSE Patients.txt file
                    Break the loop
                ELIF makesure ==”no”
                    DISPLAY ("*Nothing has changed*")
                    Break the loop

                ELSE:
                    DISPLAY (“Invalid Input”)
                ENDIF

            ENDDO
END
```

3.1.3 - DYANIEL CHING CHEE XIONG

Make sure key in valid patient id

```
Define function as checking_id()
Declare flag, patient_id, line

Prompt user for patient_id
Read patient_id
Open patient.txt in reading mode
Set flag = False

For each line in patient.txt
        IF (line start with id) Then
                Set flag = True
                Stop looping
        ENDIF

IF (flag = False) THEN
        Display ('Invalid patient id, pls re-submit.')
        Set id = checking_id()
Close patient.txt
Return id
```

Figure 29:Pseudocode checking_id()

Determine patient age

```
Define function as checking_age(id)
Declare age, line, id

Open patient.txt as reading mode
        for each line in patient.txt
                IF(line start with id) THEN:
                        Read age of patient from line
                        Stop looping
                ENDIF

Close patient.txt
Return age
```

Figure 30:Pseudocode checking_age ()

Approve having vaccine

```
Define function as approve (type, age)
Declare flag, type, age

IF (type == 'AF') or (type == 'DM') THEN:
        IF (age <12) THEN:
                Display ('This patient is not suitable for having this dosage.')
                Set flag = False
        ELSE:
                Set flag = True
        ENDIF

ELIF (type == 'BV') or (type == 'EC') THEN:
        IF (age < 18) THEN:
                Display ('This patient is not suitable for having this dosage.')
                Set flag = False
        ELSE:
                Set flag = True
        ENDIF
ELIF (type == 'CZ') THEN:
        IF (age <12 or age >45) THEN:
                Display ('This patient is not suitable for having this dosage.')
                Set flag = False
        ELSE:
                Set flag = True
        ENDIF

ELSE:
        Set flag = False
        Display ('Invalid vaccine code, pls re-submit')
        Run administration ()
ENDIF
Return flag
```

Figure 31: Pseudocode approve ()

Check vaccine code key in same with vaccine code in Patients text file

```
Define function as check_dose1(vac_code,id)
Declare vaccine_code, flag, line, id

Open patient.txt as reading mode
for each line in patient.txt
        IF (line start with id) THEN:
                Read vaccine_code of patient from line
                Stop looping
IF (vac_code == vaccine_code) THEN:
        Set flag = True

ELSE:
        Display ('The vaccine code key in is different with registration. Pls re-submit')
        Set flag = False
ENDIF
Close patient.txt
Return flag
```

Figure 32: Pseudocode check_dose1 ()

Confirm patient reach 2nd dose date

```
Define function as check_2nd_dose_date(id)
Declare current_date, 2nd_dose_date, flag,line, id

Open vaccination.txt in reading mode
Set the current_date from system
for each line in vaccination.txt
    IF (id in line) THEN:
        Read the 2nd_dose_date from vaccination.txt
        IF (current_date > 2nd_dose_date) THEN:
            Set flag = True
            Stop looping
        ELSE:
            Display ('The second dose date haven't reach.')
            Set flag = False
            Stop looping
        ENDIF
    ENDIF
Close vaccination.txt
Return flag
```

Figure 33: Pseudocode check_2nd_dose_date (id)

Check vaccine code of 2nd dose same with 1st dose

```
Define function as check_dose2(dose2_code, id)
Declare vaccine_code, flag, line, dose2_code, id

Open vaccination.txt as reading mode
for each line in vaccination.txt
    IF (id in line) THEN:
        Read vaccine_code from vaccination.txt
        IF (dose2_code == vaccine_code) THEN:
            Set flag = True
            Stop looping

        ELIF (vaccine_code == 'EC') THEN:
            Display ('EC vaccine no need have second dose.')
            Set flag = None
            Stop looping
        ELSE:
            Display ('vaccine code is different with dose 1. Pls re-submit')
            Set flag = False
            Stop looping
        ENDIF

Close vaccination.txt
Return flag
```

Figure 34: Pseudocode check_dose2(dose2_code, id)

Dose 1 step

Define function as dose1_step(id,code)

Declare current_date, 2nd_dose_date, id, code

Open vaccination.txt as append mode

Set the current_date from system

IF (code == 'AF') THEN:

    Calculate 2nd_dose_date = current_date + 14 days

    Write id,'dose1', code, 2nd_dose_date into vaccination.txt

    Display ('Submit Successfully') and 2nd_dose_date

ELIF (code == 'BV' or code == 'CZ') THEN:

    Calculate 2nd_dose_date = current_date + 21 days

    Write id,'dose1', code, 2nd_dose_date into vaccination.txt

    Display ('Submit Successfully') and 2nd_dose_date

ELIF (code == 'DM') THEN:

    Calculate 2nd_dose_date = current_date + 28 days

    Write id,'dose1', code, 2nd_dose_date into vaccination.txt

    Display ('Submit Successfully') and 2nd_dose_date

ELIF (code == 'EC') THEN:

    Write id,'dose1', code, 'NONE' into vaccination.txt

    Display ('Submit Successfully') and 'patient get EC vaccine no need has second dose.'

ENDIF

Close vaccination.txt

Return

Figure 35: Pseudocode dose1_step ()

Dose 2 step

```
Define function as dose2_step(id,code)

Declare count, lines, line, id, code

Set count = 0


Open vaccination.txt as reading mode

For each line in vaccination.txt

        Calculate count = count+1

        IF Stop THEN:

                Stop looping

        ENDIF


Read all lines in vaccination.txt and save in list named lines
Replace the dose1 status of patient with dose2 in list
close vaccination.txt


Open vaccination.txt in writing mode

For each line in lines

        Write line into vaccination.txt

ENDLOOP


Display ('Submit Successfully')

Close vaccination.txt

Return
```

Figure 36: Pseudocode dose2_step(id,code)

Vaccine Administration

```
Define function as administration ()
Declare patient_id , dosage_num, vac_code, age, approve_status, flag, check_dose2_approve, date_approve, check_dose1_approve, line
Set patient_id = checking_id()
Prompt user for dosage_num and vac_code
Read patient_id , dosage_num and vac_code
Set age = checking_age(patient_id)
Set approve_status = approve (vac_code, age)

IF (approve_status == True) THEN:
        IF (dosage_num == 'dose2') :
                Set flag = True
                Open vaccination.txt in reading mode

                For each line in vaccination.txt
                        IF (line start with patient_id) THEN:
                                IF ('dose1' in line) THEN:
                                        IF (vac_code == 'EC') THEN:
                                                Display ('EC vaccine no need have second dose.')
                                                Set flag = False
                                                Stop looping
                                        ELSE:
                                        Set flag = False
                                        Set check_dose2_approve = check_dose2(vac_code,patient_id)
                                                IF (check_dose2_approve == True) THEN:
                                                        Set date_approve = check_2nd_dose_date (patient_id)
                                                        IF (date_approve == True) THEN:
                                                                Run dose2_step(patient_id,vac_code)
                                                                Stop looping
                                                        ENDIF
                                                ELIF (check_dose2_approve == False) THEN:
                                                        Run administration ()
                                                ELSE:
                                                        Stop looping
                                                ENDIF
```

```
                                        ENDIF
                                ELSE:
                                Display ('This patient has done second dose')
                                Set flag = False
                                Stop looping
                                ENDIF
                        ENDIF
                IF (flag == True) THEN:
                        Display ('This patient no having dose one yet. System will set as dose1 status')
                        Set check_dose1_approve = check_dose1(vac_code, patient_id)
                        IF (check_dose1_approve == True) THEN:
                                Run dose1_step(patient_id,vac_code)
                        ELSE:
                                Run administration ()
                        ENDIF
                Close vaccination.txt
                ENDIF
ELIF (dosage_done == 'dose1') THEN:
                Set flag = True
                Open vaccination.txt in reading mode

                 For each line in vaccination.txt
                        IF (line start with patient_id) THEN:
                                IF ('dose1' in line) THEN:
                                        IF (vac_code == 'EC') THEN:
                                                Display ('EC vaccine no need have second dose.')
                                                Set flag = False
                                                Stop looping
                                        ELSE:
                                        Display ('This patient has done first dose. System will set as dose2 status')
                                        Set flag = False
                                        Set check_dose2_approve = check_dose2(vac_code,patient_id)
                                                IF (check_dose2_approve == True) THEN:
                                                        Set date_approve = check_2nd_dose_date (patient_id)
                                                        IF (date_approve == True) THEN:
```

```
                                                    Run dose2_step(patient_id,vac_code)
                                                    Stop looping
                                                ENDIF
                                            ELIF (check_dose2_approve == False) THEN:
                                                Run administration ()
                                            ELSE:
                                                Stop looping
                                            ENDIF
                                        ENDIF
                                    ELSE:
                                    Display ('This patient has done second dose')
                                    Set flag = False
                                    ENDIF
                                ENDIF
                            IF (flag == True) THEN:
                                Set check_dose1_approve = check_dose1(vac_code, patient_id)
                                IF (check_dose1_approve == True) THEN:
                                    Run dose1_step(patient_id,vac_code)
                                ELSE:
                                    Run administration ()
                            Close vaccination.txt
                            ENDIF
                    ELSE:
                            Display ('Invalid enter for dose number, pls re-submit.')
                            Run administration ()
                    ENDIF
ENDIF
Run admin_menu ()
```

Figure 37: Pseudocode administration ()

Delete_vaccine () function

Define function as delete_vaccine ()

Declare flag, vc_code, lists, line, line_split, ans, text, opt

Set flag = True

DOWHILE (flag==True):

Display "\n----------Delete Vaccine----------" and "*Enter '1' to exit"

Prompt user for vc_code

Read vc_code

IF (vc_code == "1") THEN:

Return

ENDIF

Open “vaccine.txt” text file in reading mode

Set an empty list named “lists”

For each line in “vaccine.txt”

Positioning every data of line that splitting by “,” and save in “line_split”

Appending the ‘line_split’ list into the ‘lists’ list.

For each line in “lists” according to the number of “lists”

IF (vc_code in start of the line of “lists”) THEN:

Set ans = “yes”

Stop looping

ELSE

Set ans = “no”

ENDIF

Close “vaccine.txt” text file

```
IF (ans == "yes") THEN:
        Open "vaccine.txt" text file in reading mode
        Read all data in "vaccine.txt" file and save into "text"
        Delete the data of the line number of above looping in "text"
        Close "vaccine.txt" text file

        Open "vaccine.txt" text file in writing mode
        For each line in "text"
            Write the data in line into "vaccine.txt"
        Close "vaccine.txt" text file
        Display "*Deleted Successfully"

    ELSE
        Display "*vaccine code not exists, Please try again"
        Run delete_vaccine () function
        Return
    ENDIF
    DOWHILE (True):
        Prompt user for opt
        Read opt
        IF (opt == "yes") THEN:
            Stop looping
        ELIF (opt == "no") THEN:
            Set flag = False
            Stop looping
        ELSE
            Display "Invalid Input"
    ENDDO
ENDDO
Return
```

Figure 38: Pseudocode delete_vaccine ()

3.2 Flowchart

3.2.1 - GAN MING HUI

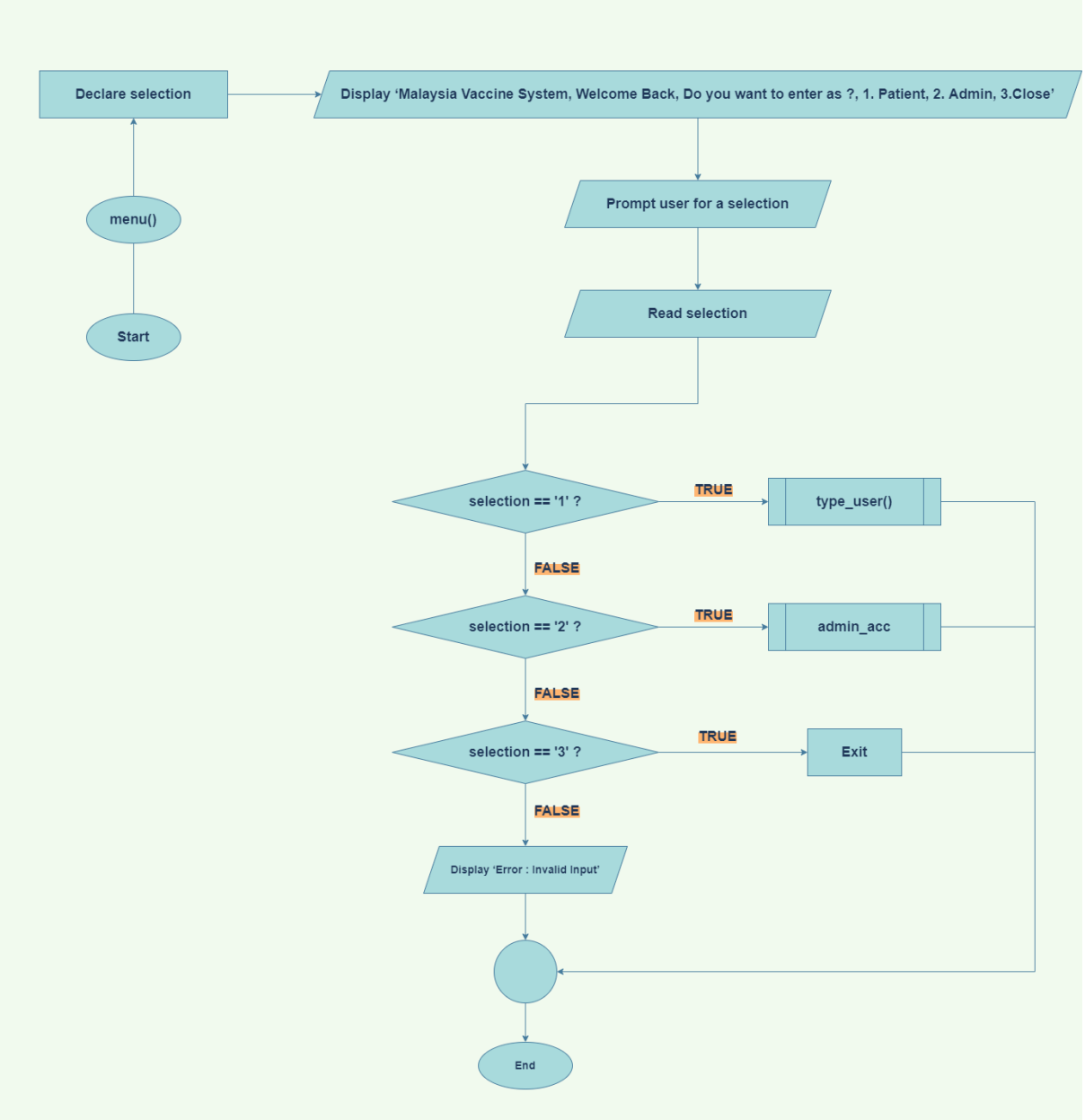


Figure 39: Flowchart menu()

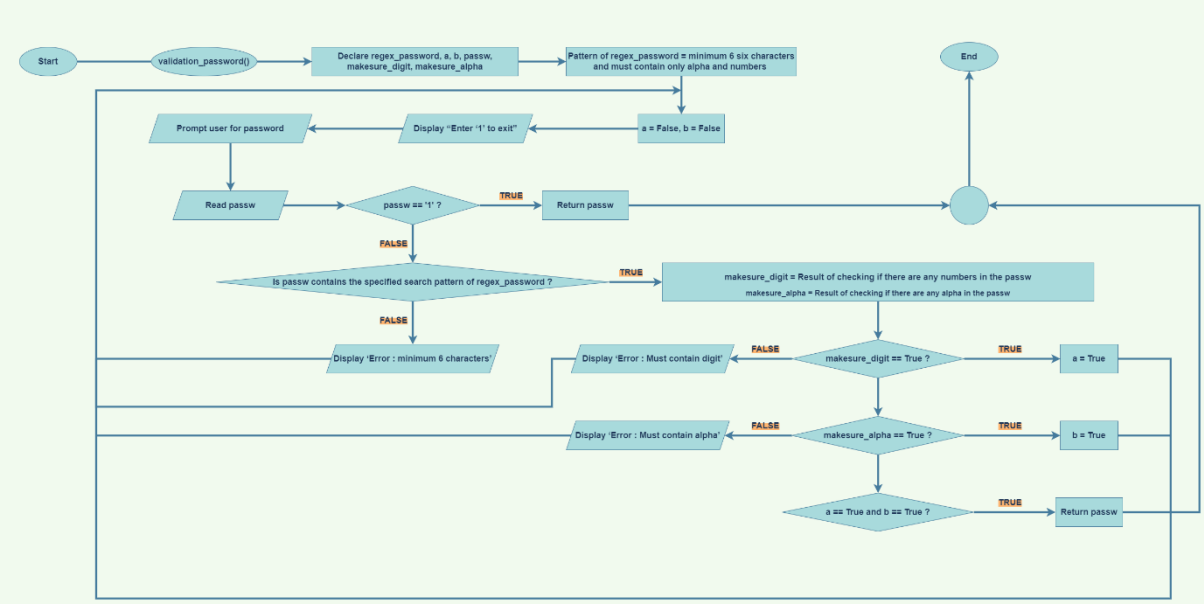


Figure 40: Flowchart validation_password()

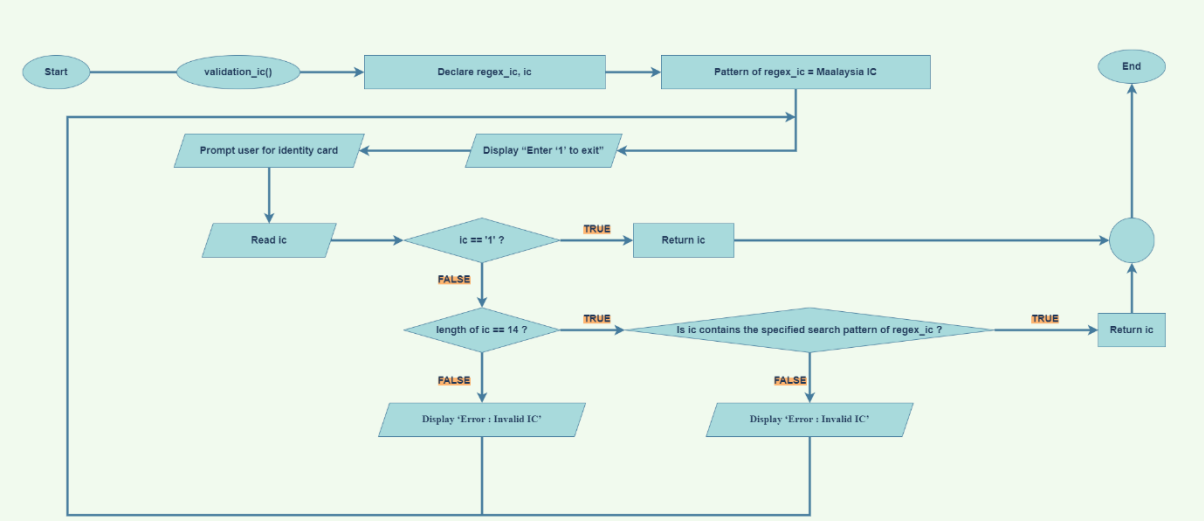


Figure 41: Flowchart validation_ic()

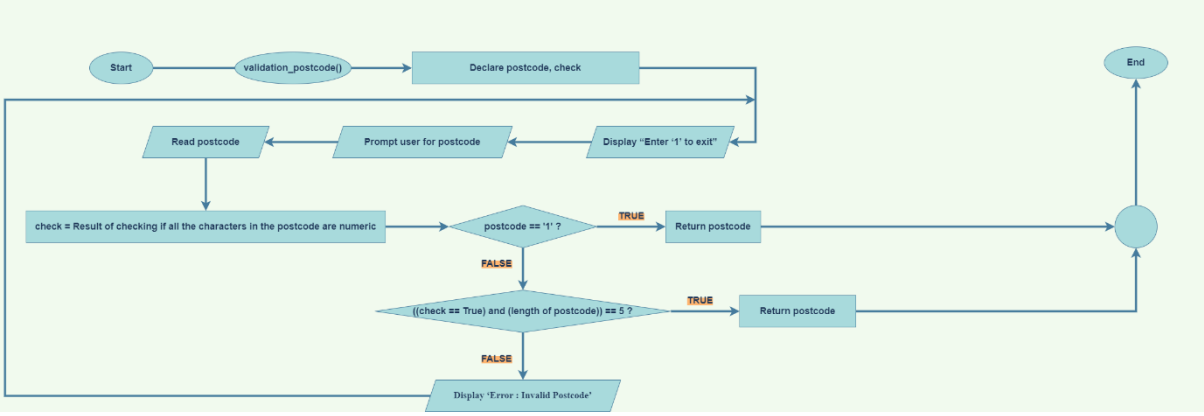


Figure 42: Flowchart validation_postcode()

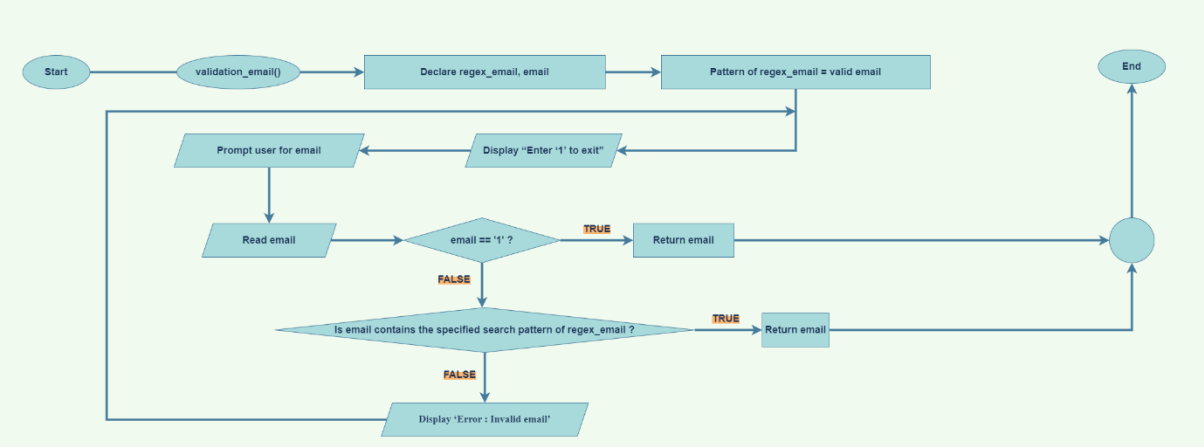


Figure 43: Flowchart validation_email()

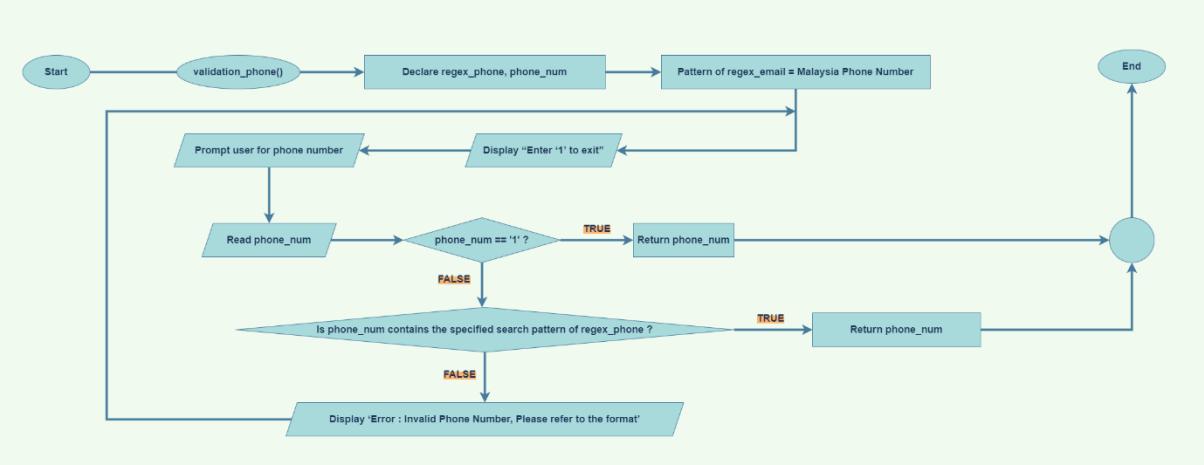


Figure 44: Flowchart validation_phone()

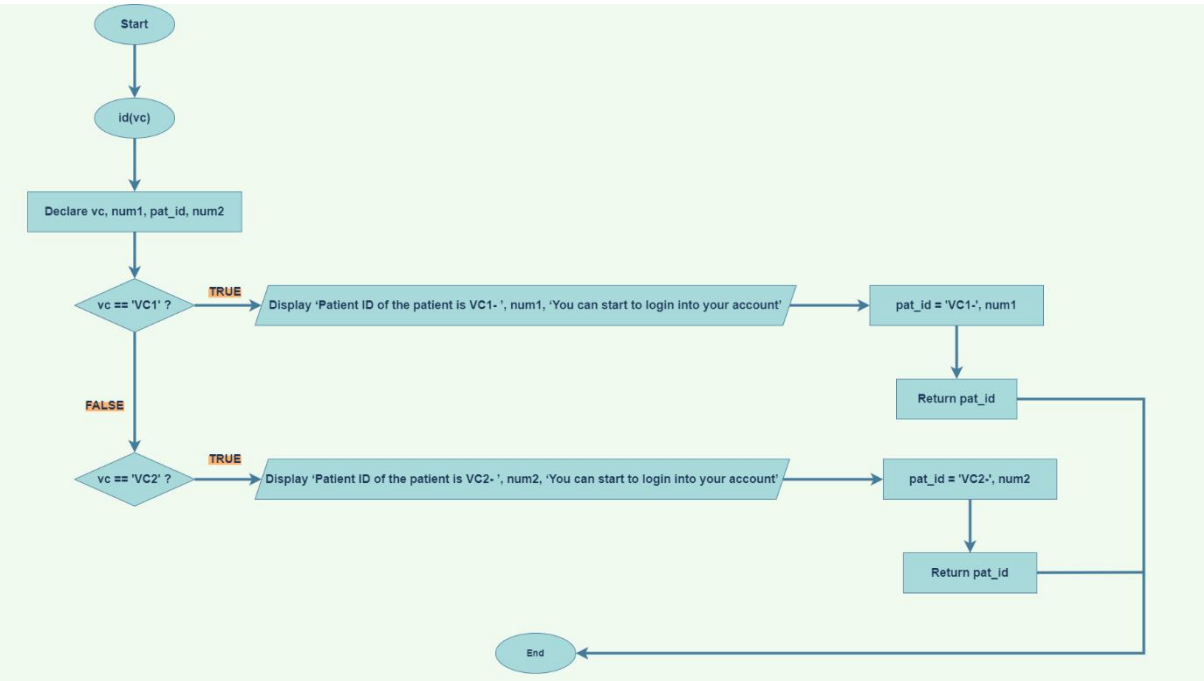


Figure 45: Flowchart id(vc)

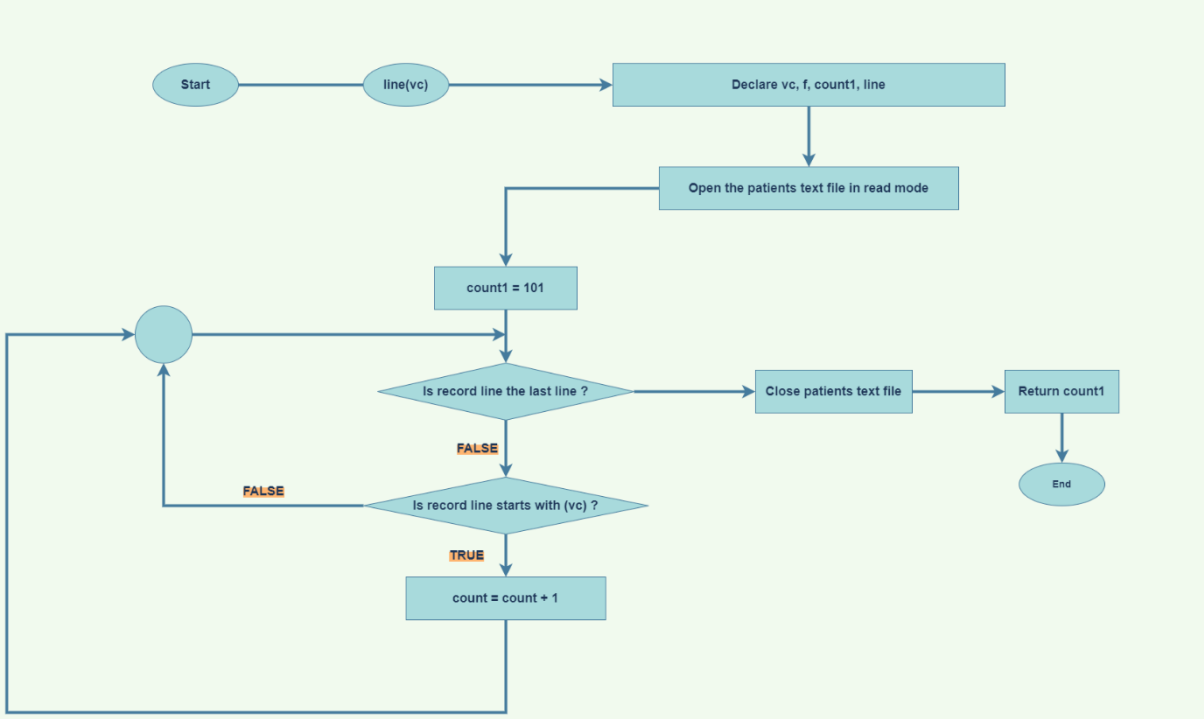


Figure 46: Flowchart line(vc)

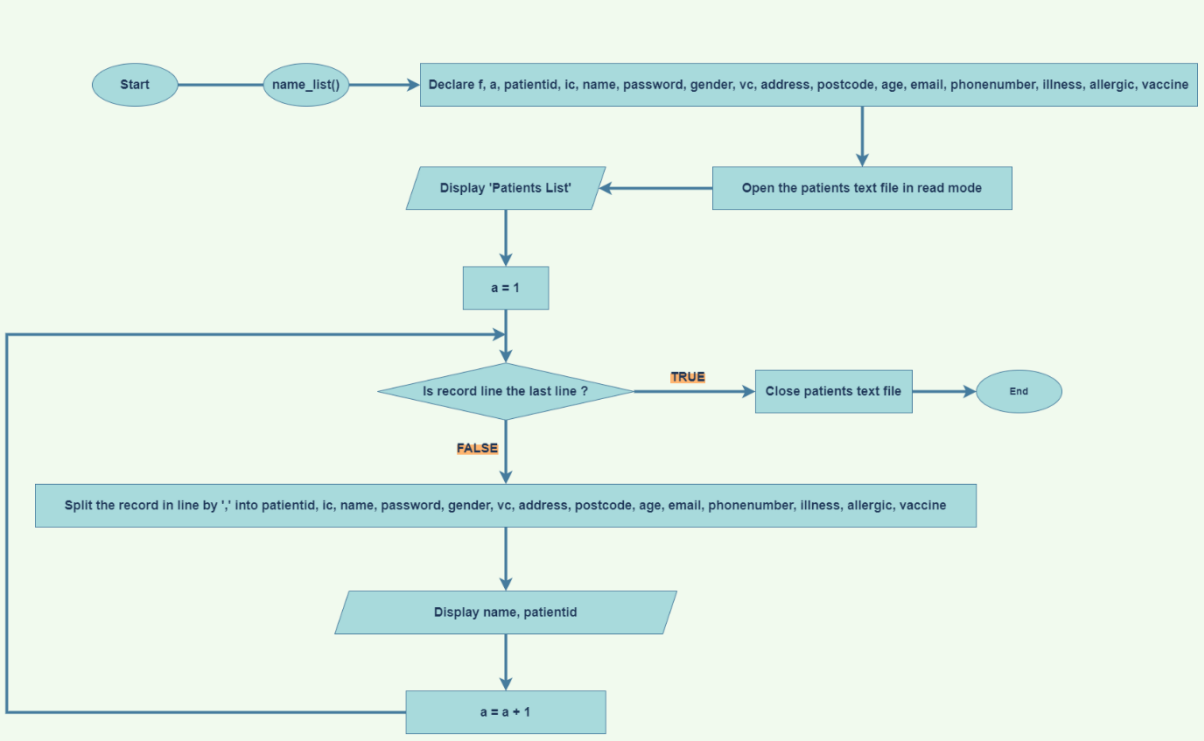


Figure 47: Flowchart name_list()

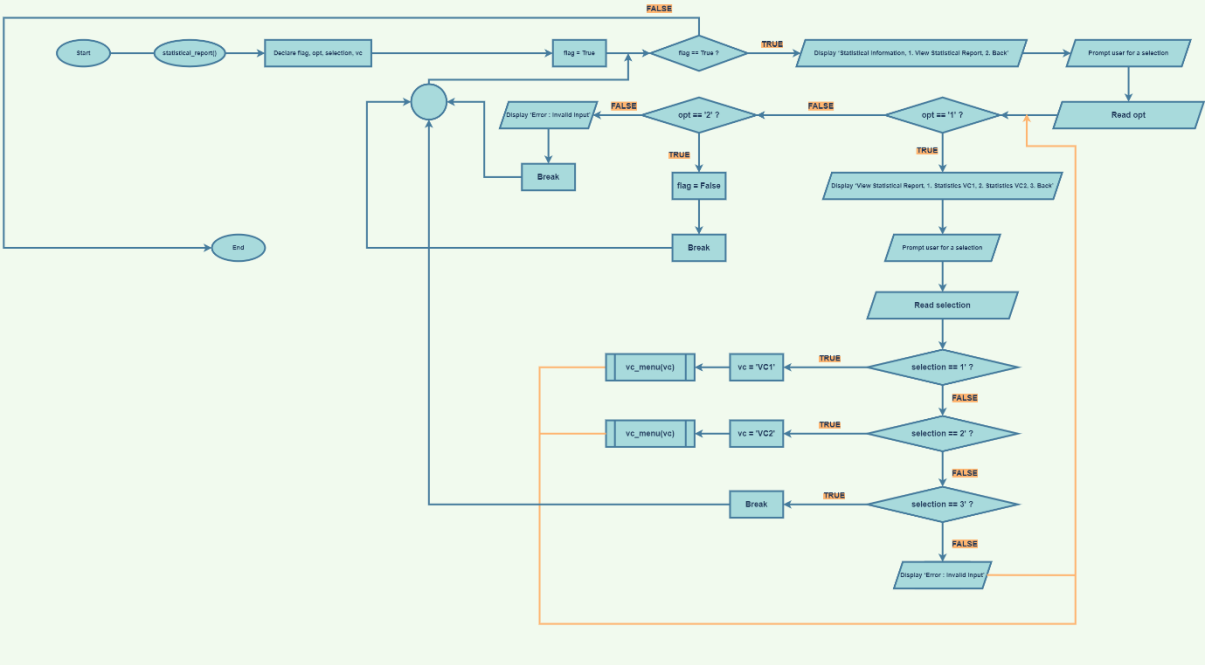


Figure 48: Flowchart statistical_report()

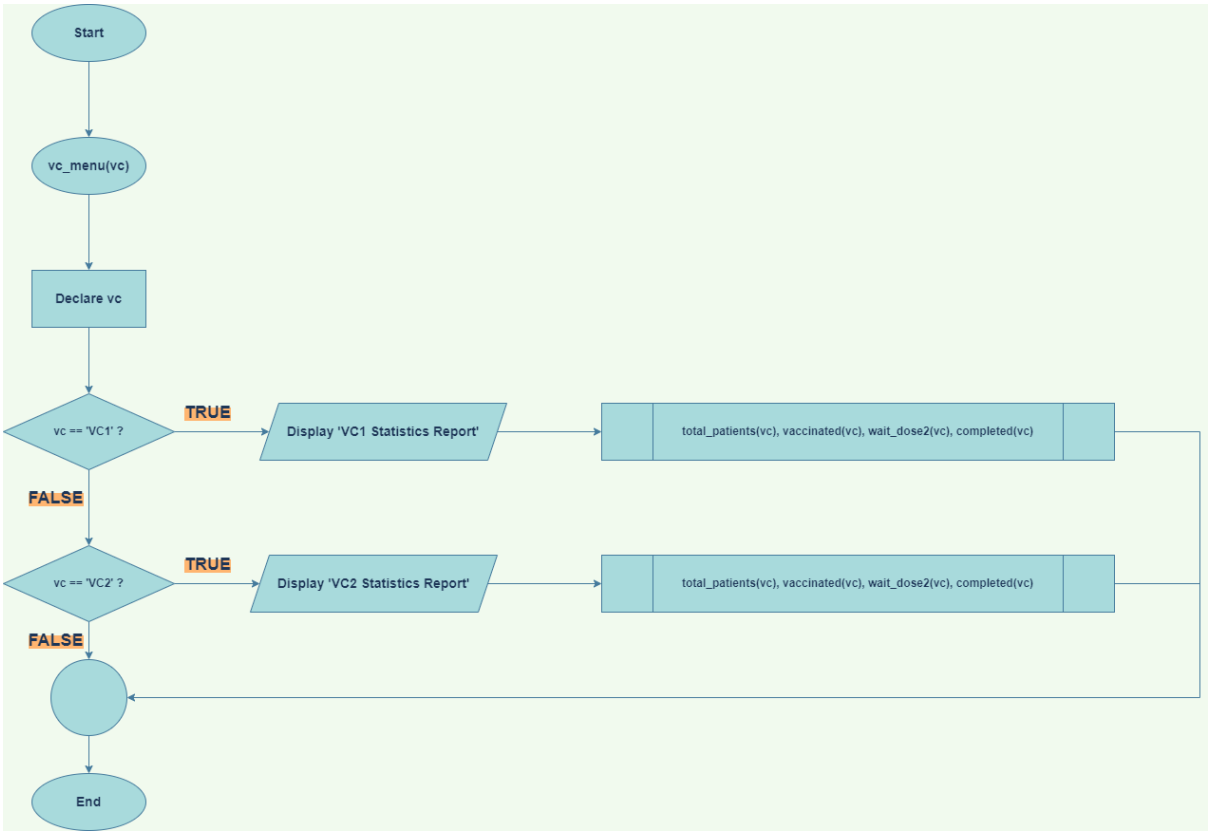


Figure 49: Flowchart vc_menu(vc)

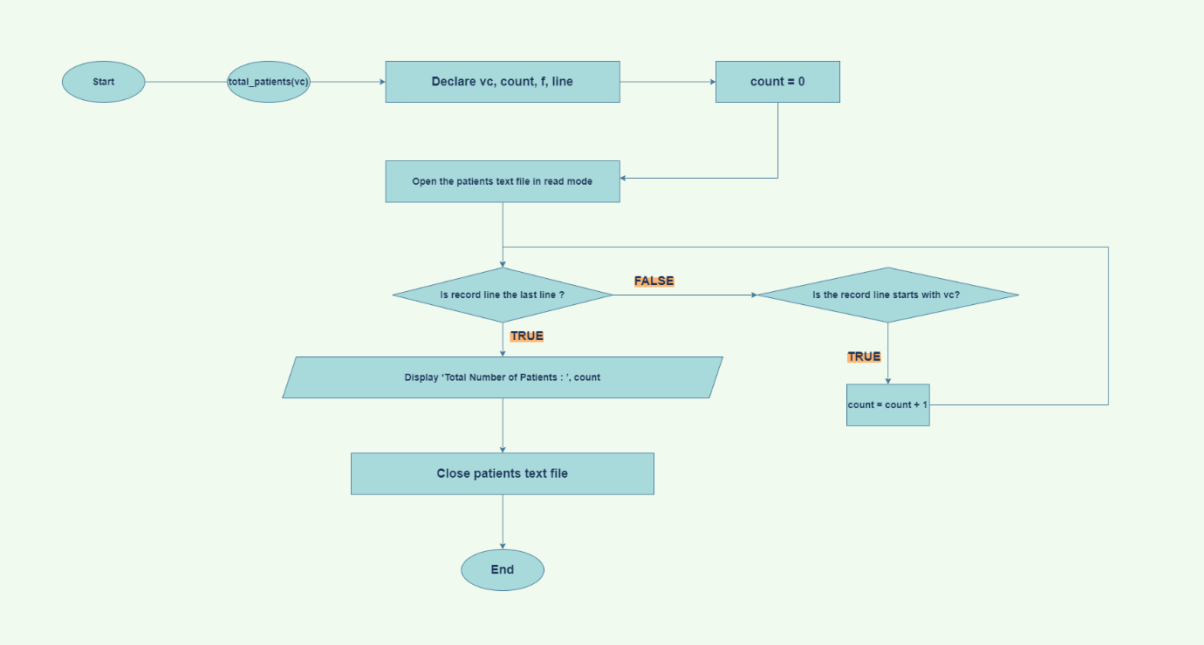


Figure 50: Flowchart total_patients(vc)

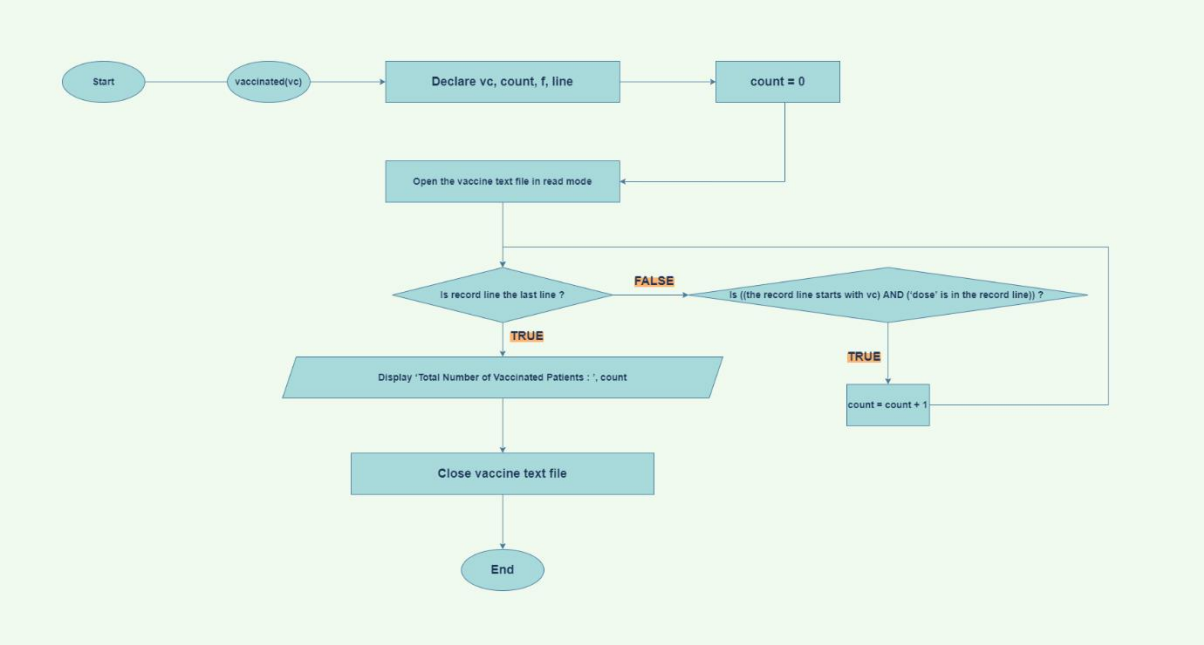


Figure 51: Flowchart vaccinated(vc)

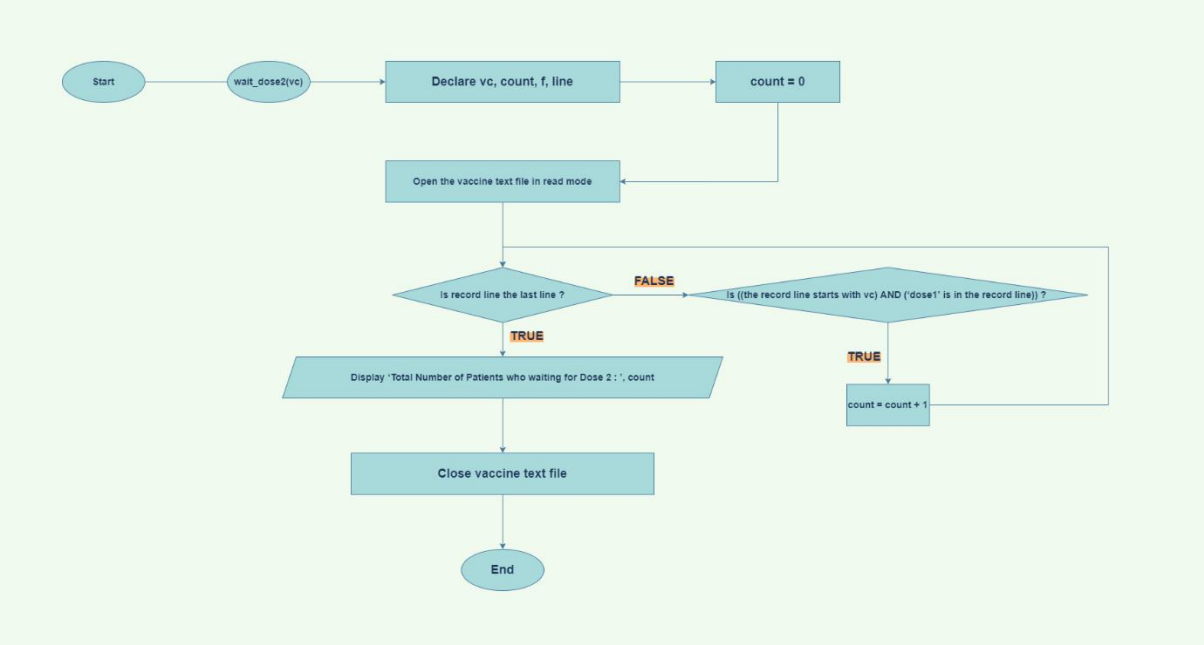


Figure 52: wait_dose2(vc)

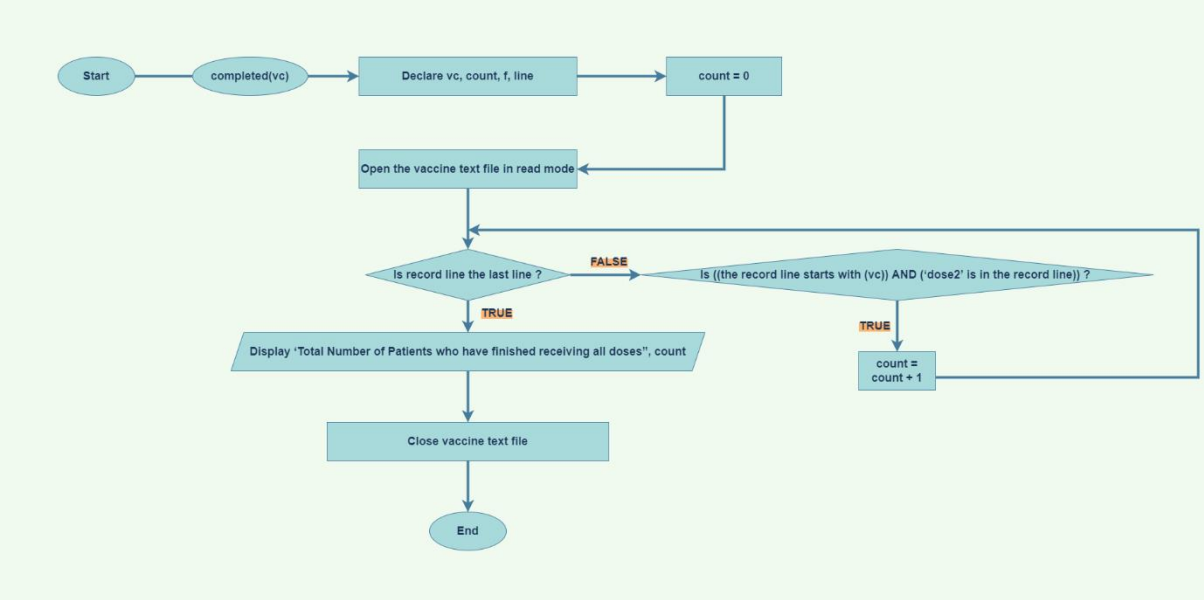


Figure 53: Flowchart completed(vc)

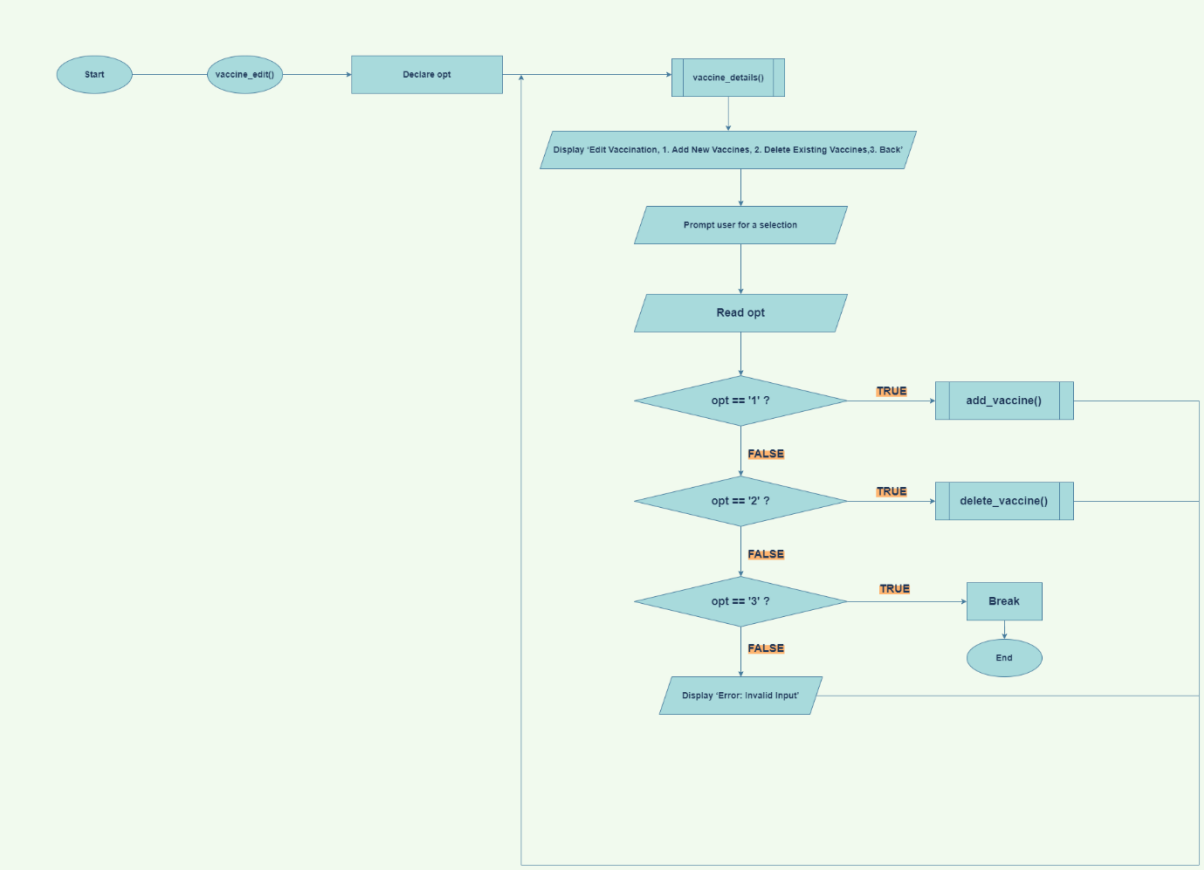


Figure 54: Flowchart vaccine_edit()

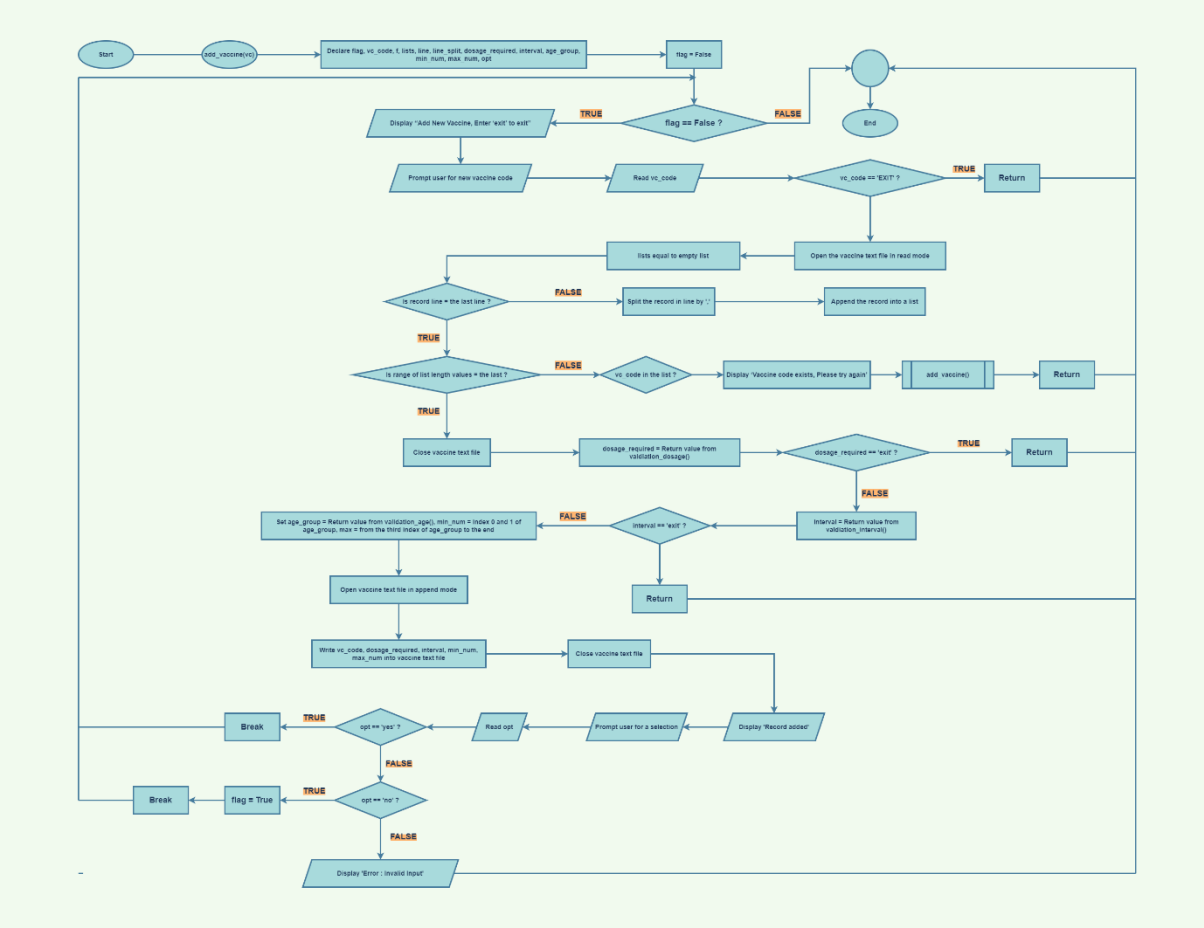


Figure 55: Flowchart add_vaccine()

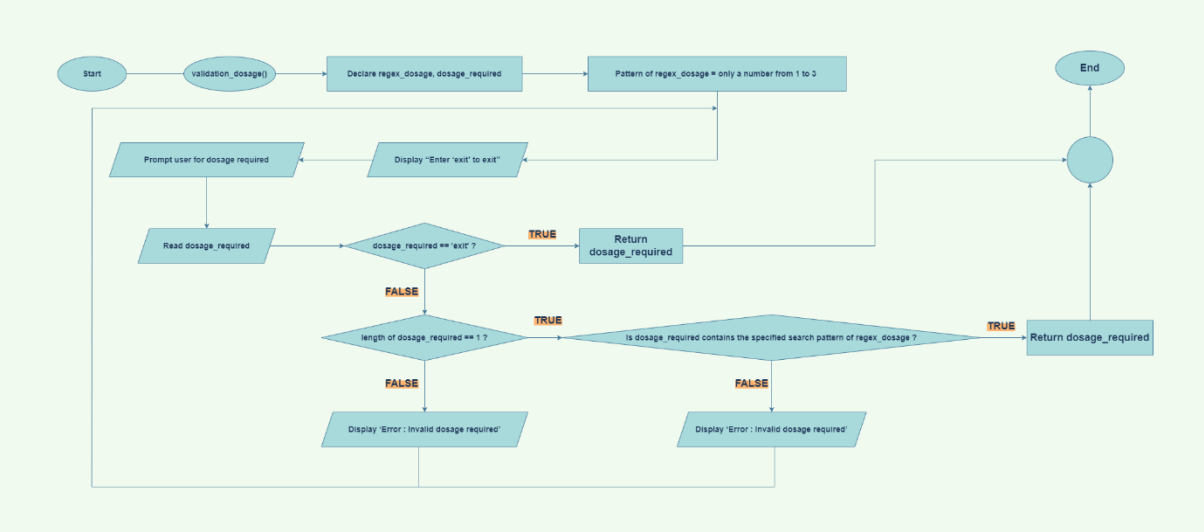


Figure 56: Flowchart validation_dosage()

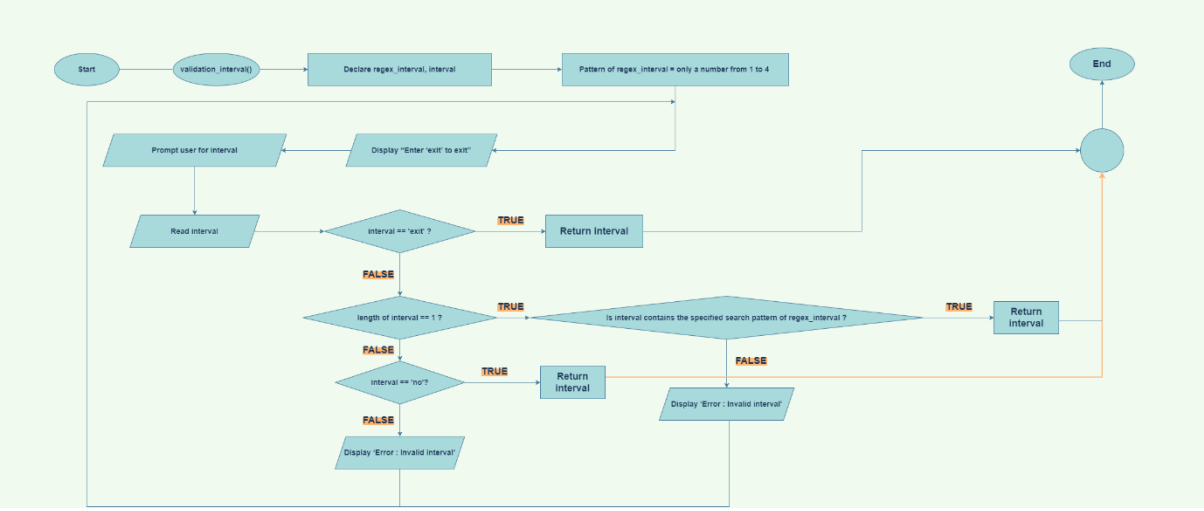


Figure 57: Flowchart validation_interval()

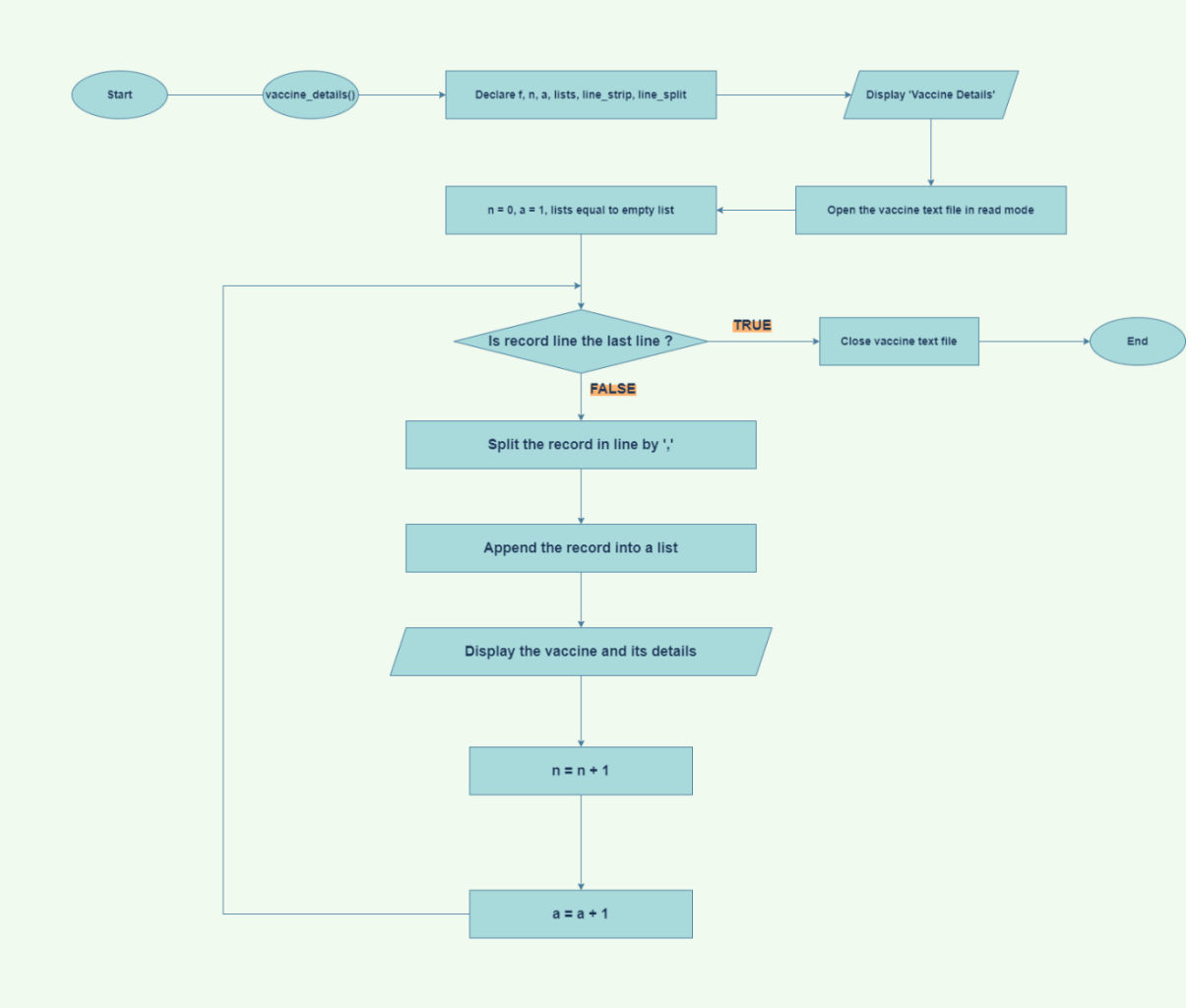


Figure 58: Flowchart vaccine_details()

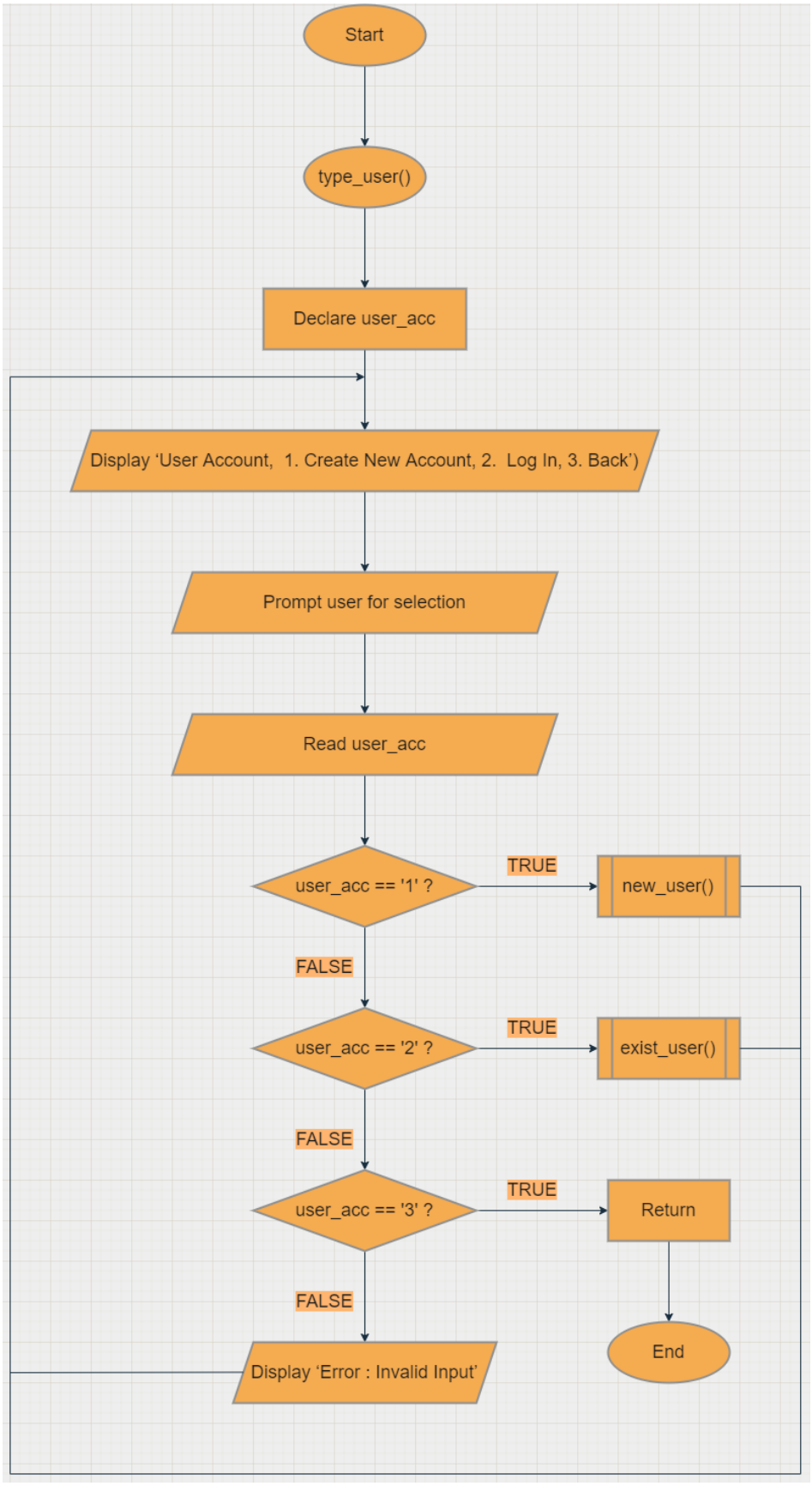


Figure 59: Flowchart type_user()

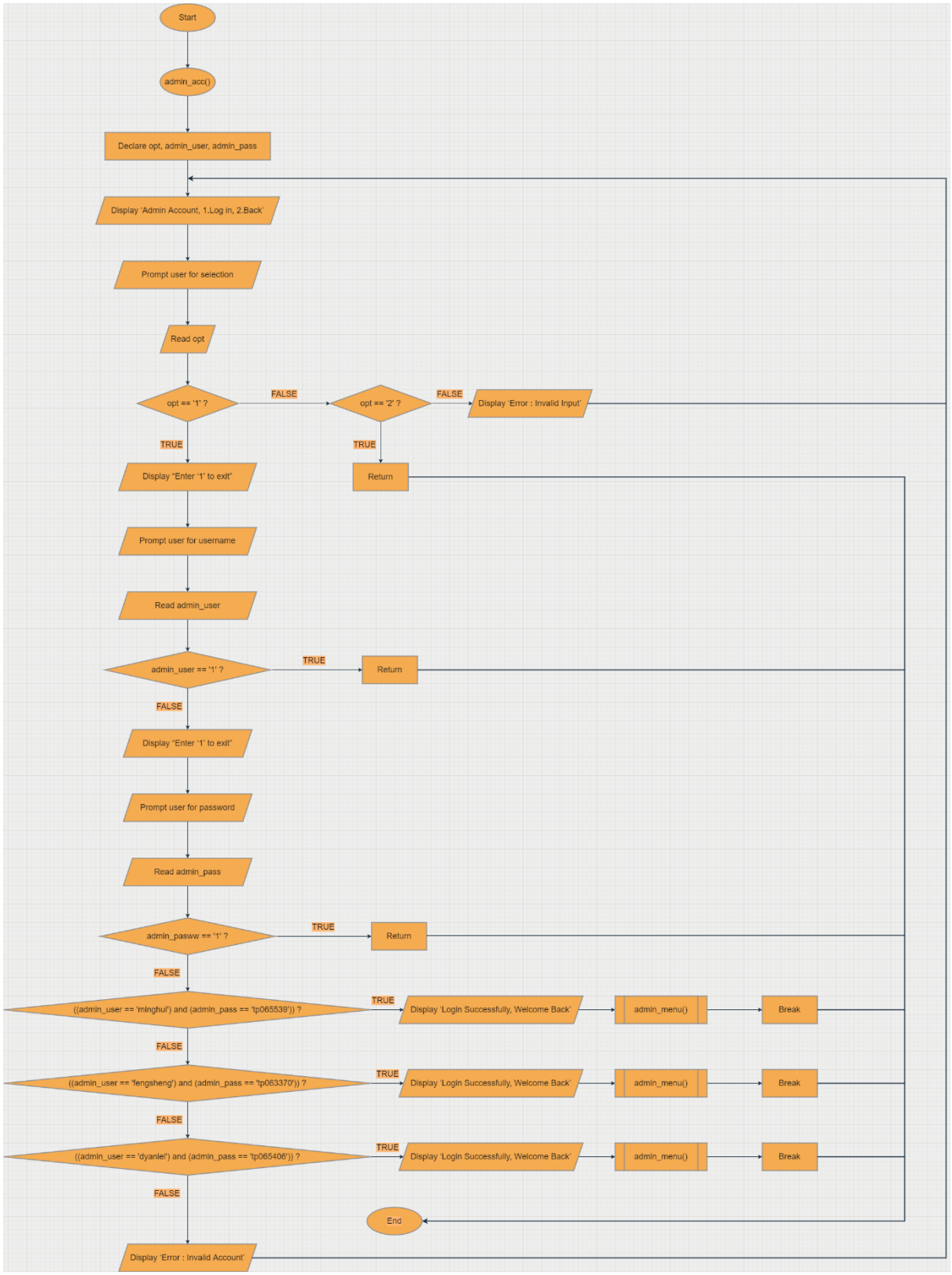


Figure 60: Flowchart admin_acc()

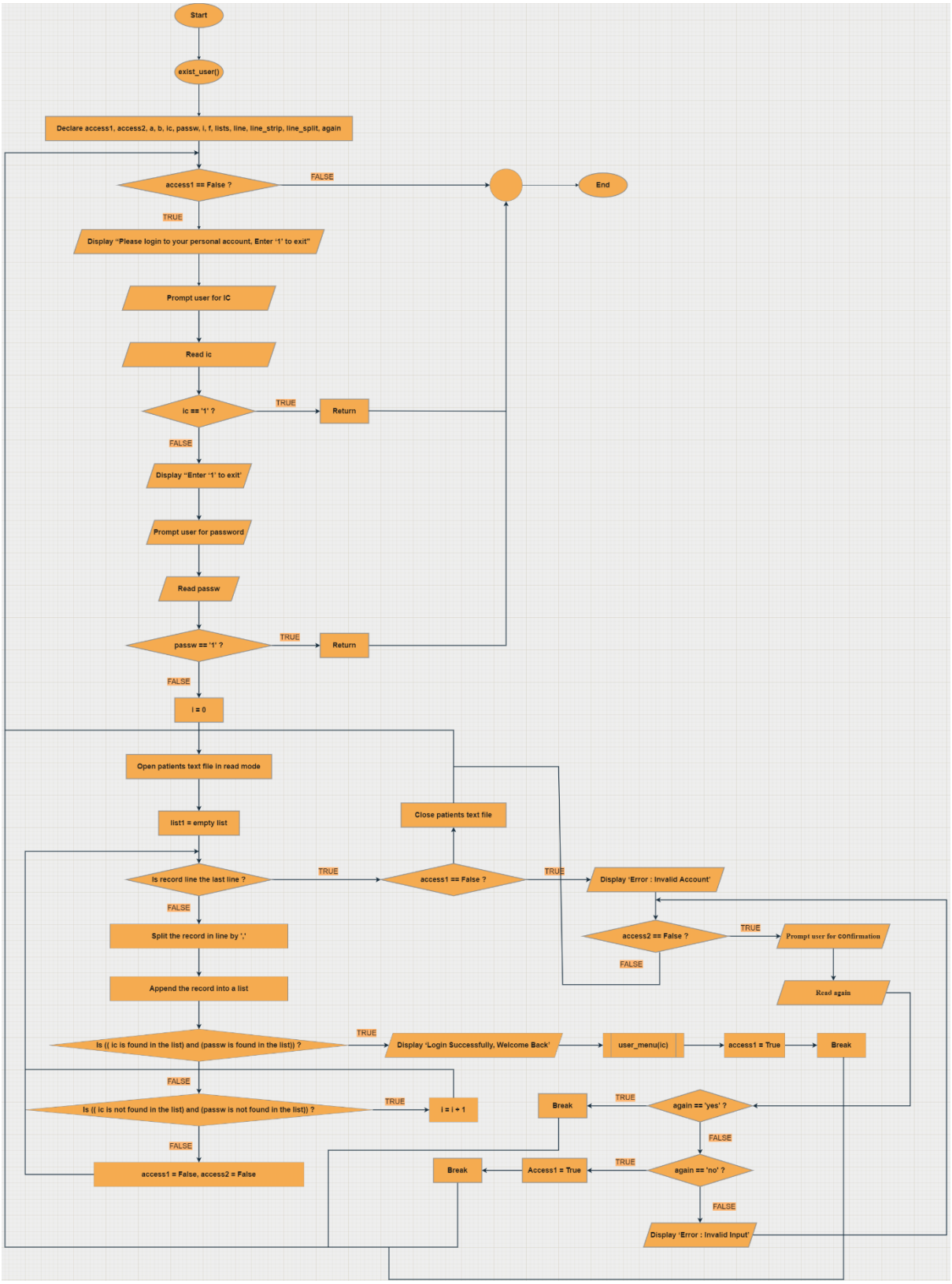


Figure 61: Flowchart exist_user()

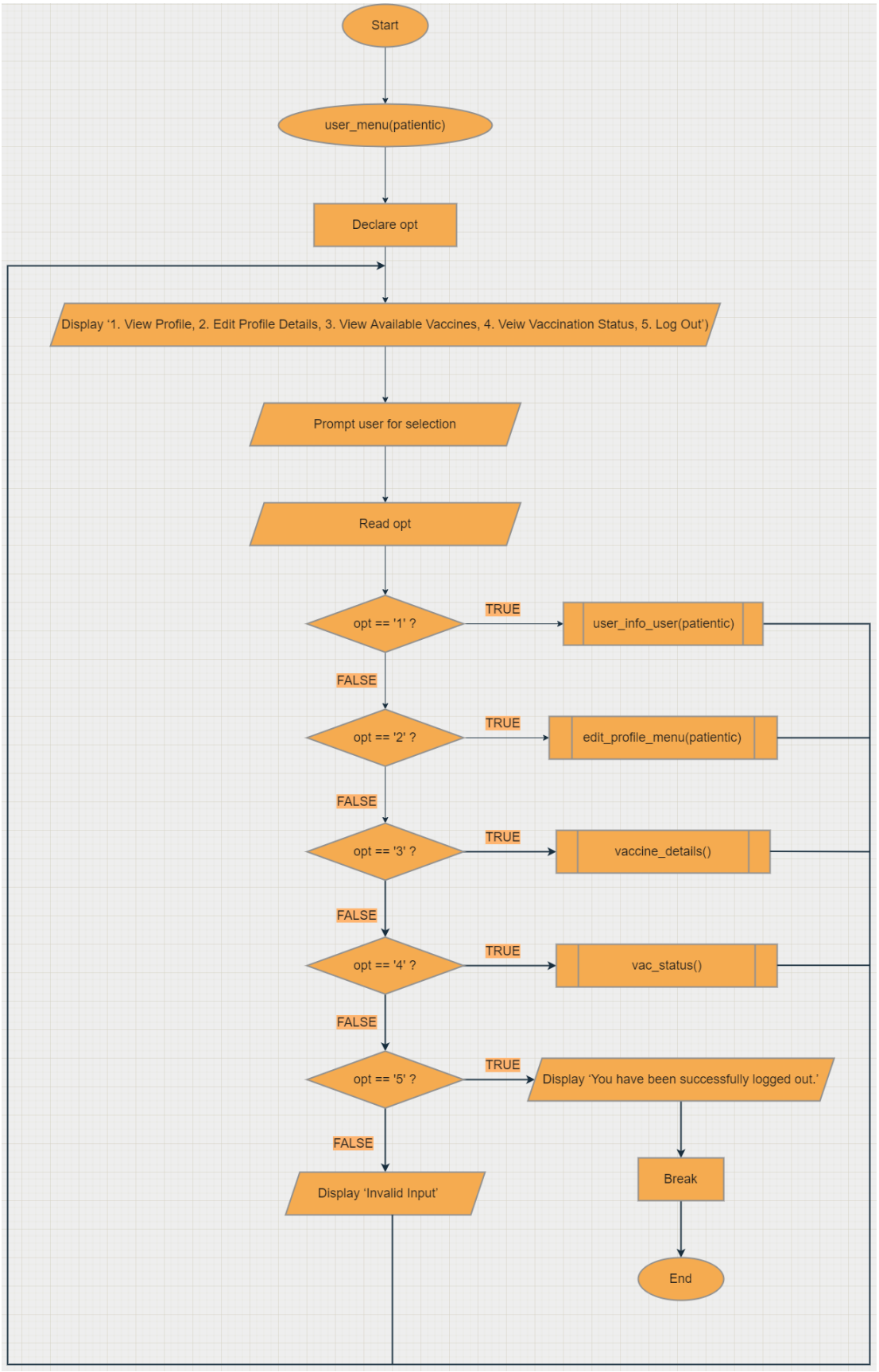


Figure 62: Flowchart user_menu(patientic)

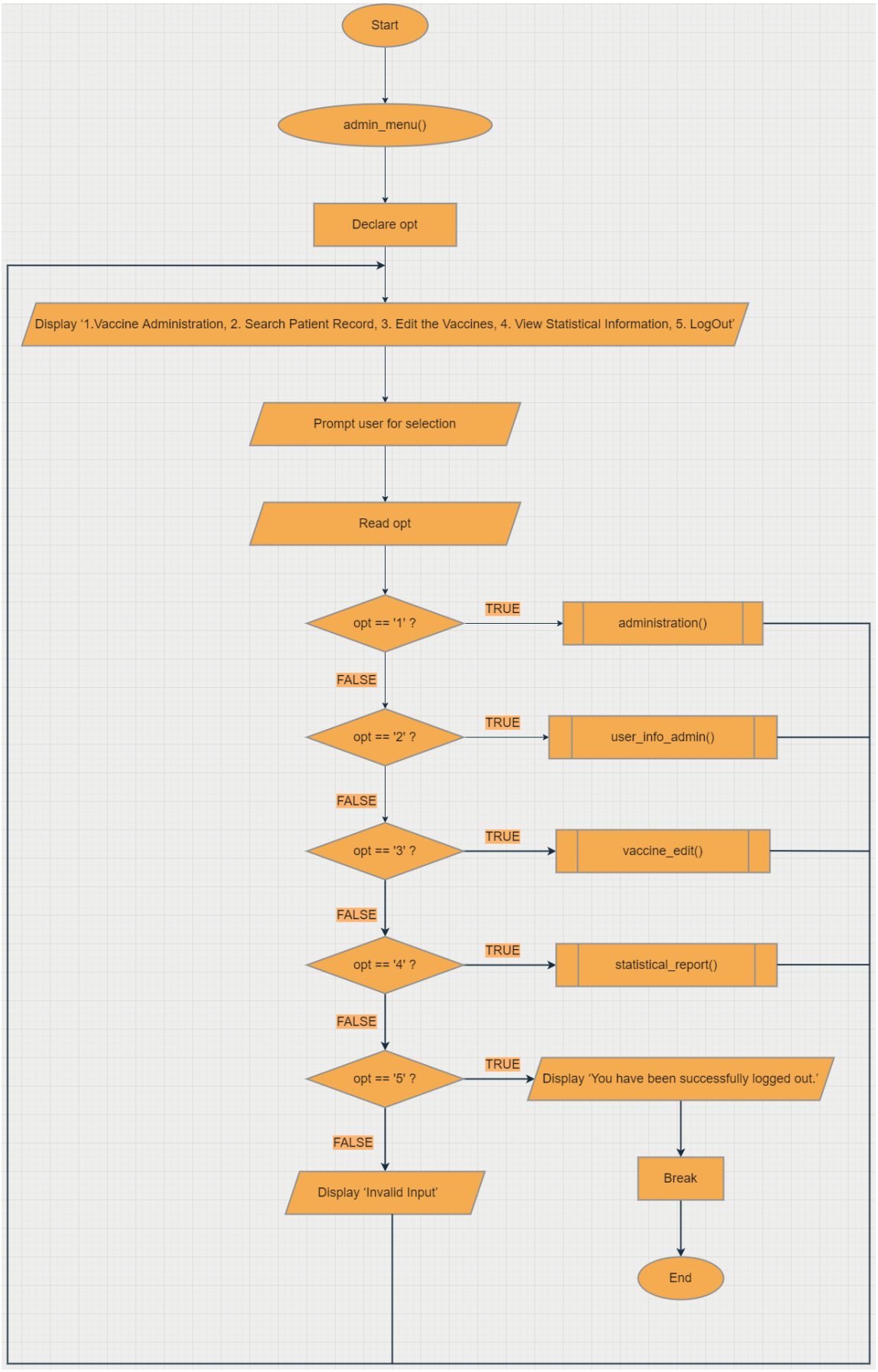
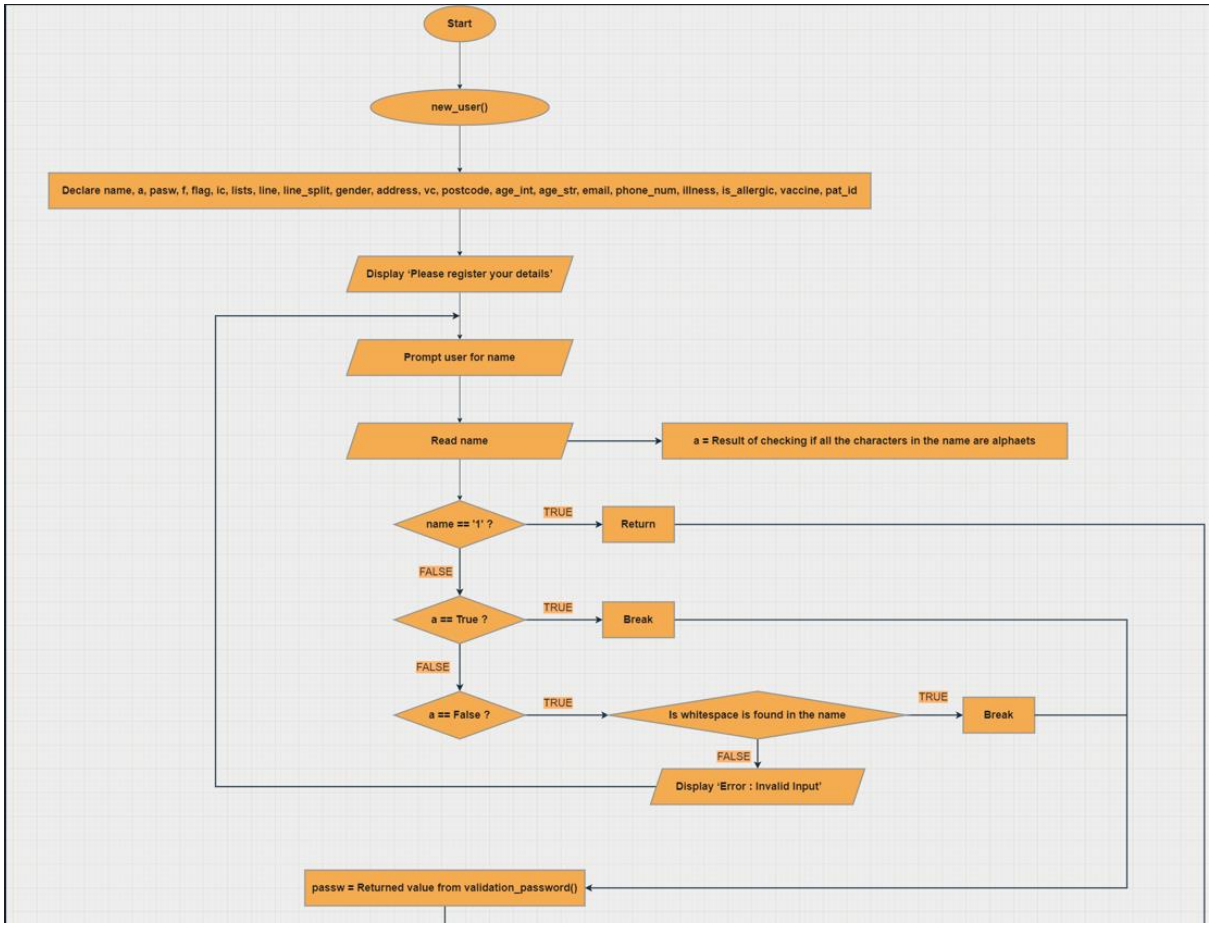
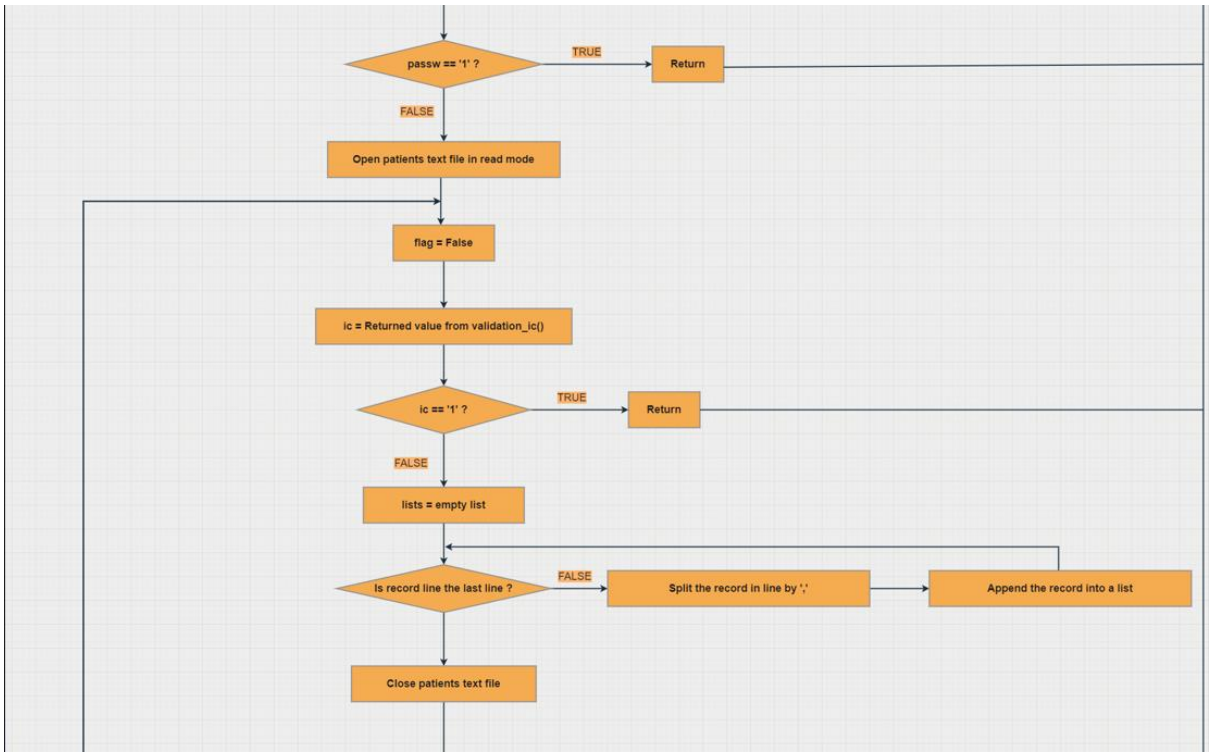
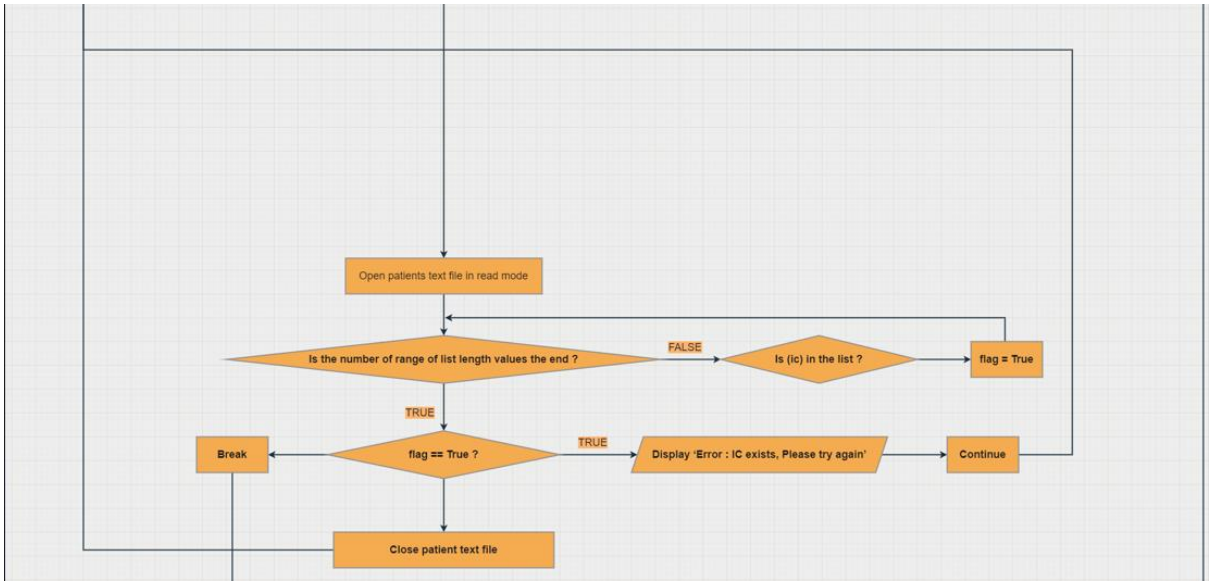
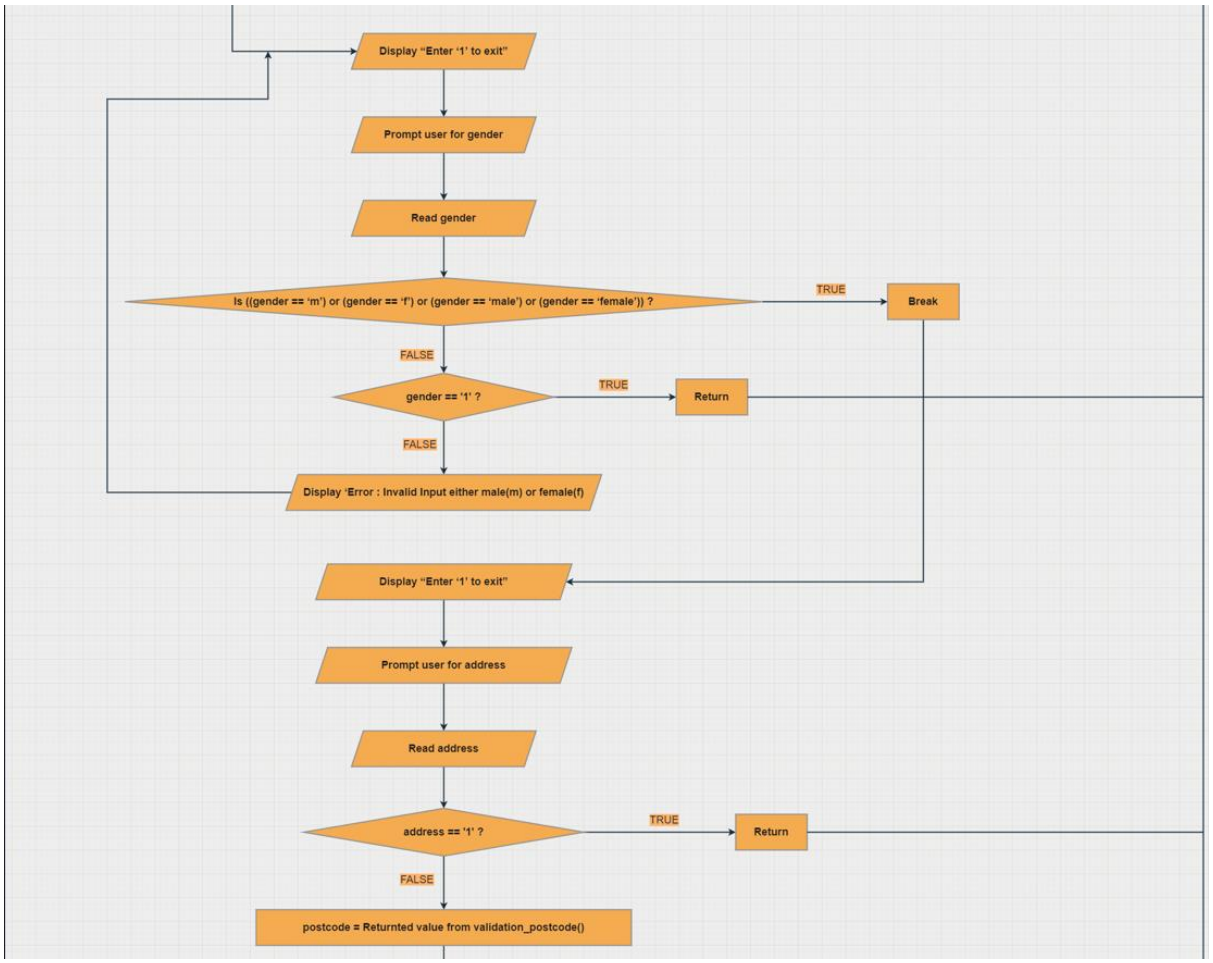


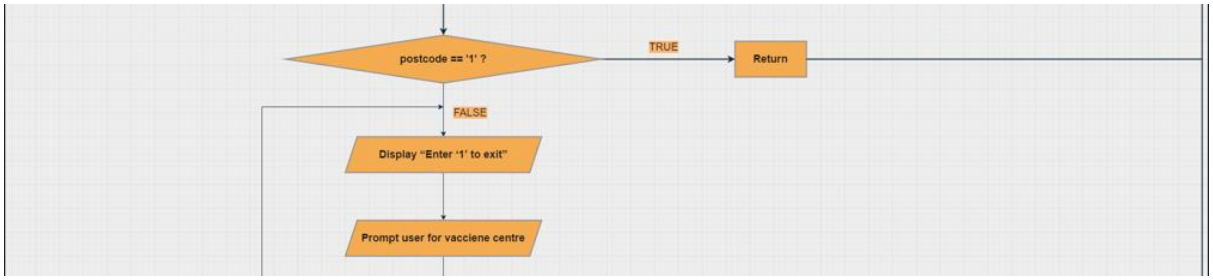
Figure 63: admin_menu( )

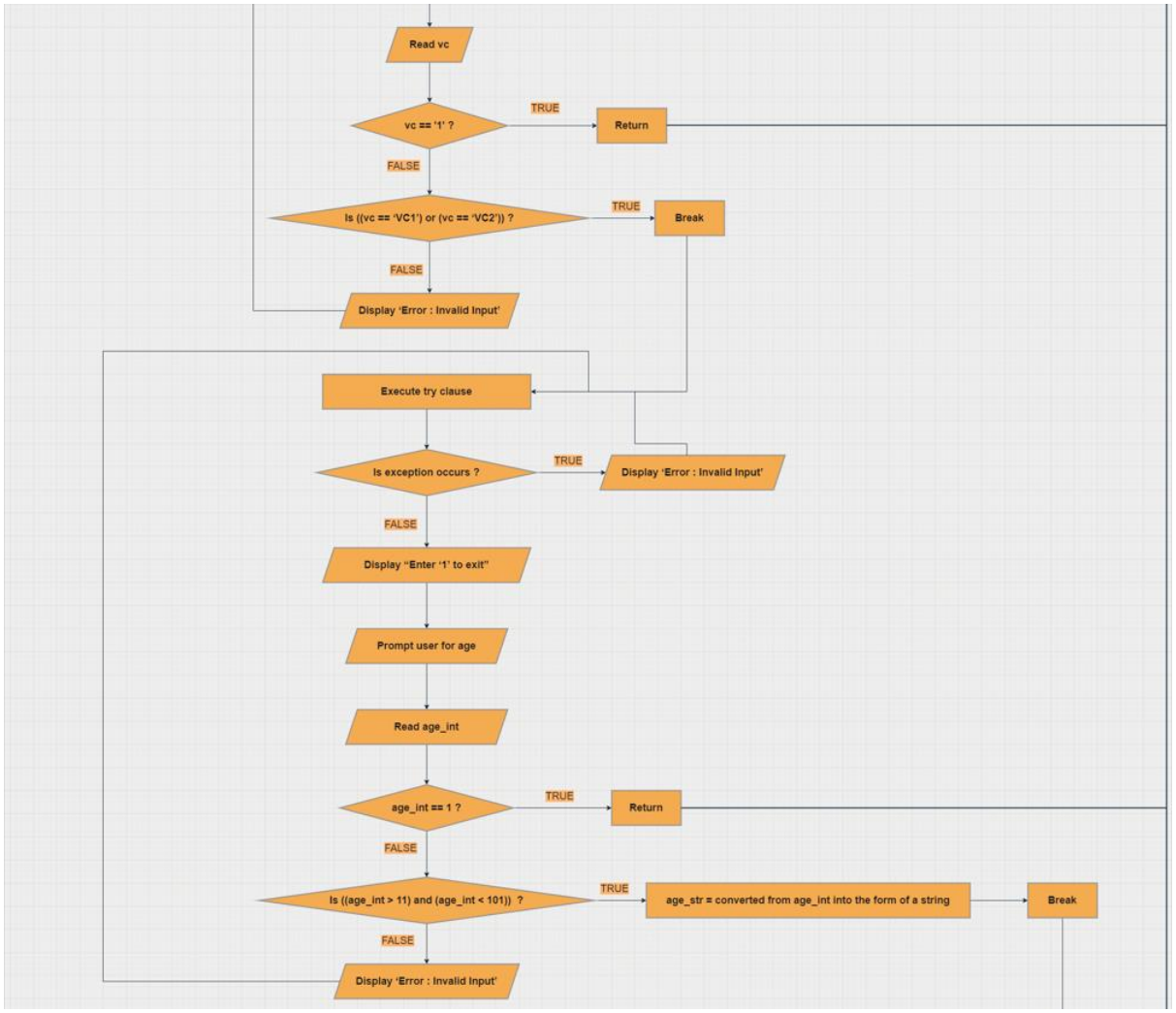

Start
new_user()
Declare name, a, pasw, f, flag, ic, lists, line, line_split, gender, address, vc, postcode, age_int, age_str, email, phone_num, illness, is_allergic, vaccine, pat_id
Display 'Please register your details'
Prompt user for name
Read name
a = Result of checking if all the characters in the name are alphaets
name == '1' ?
TRUE
Return
FALSE
a == True ?
TRUE
Break
FALSE
a == False ?
TRUE
Is whitespace is found in the name
TRUE
Break
FALSE
Display 'Error : Invalid Input'
passw = Returned value from validation_password()

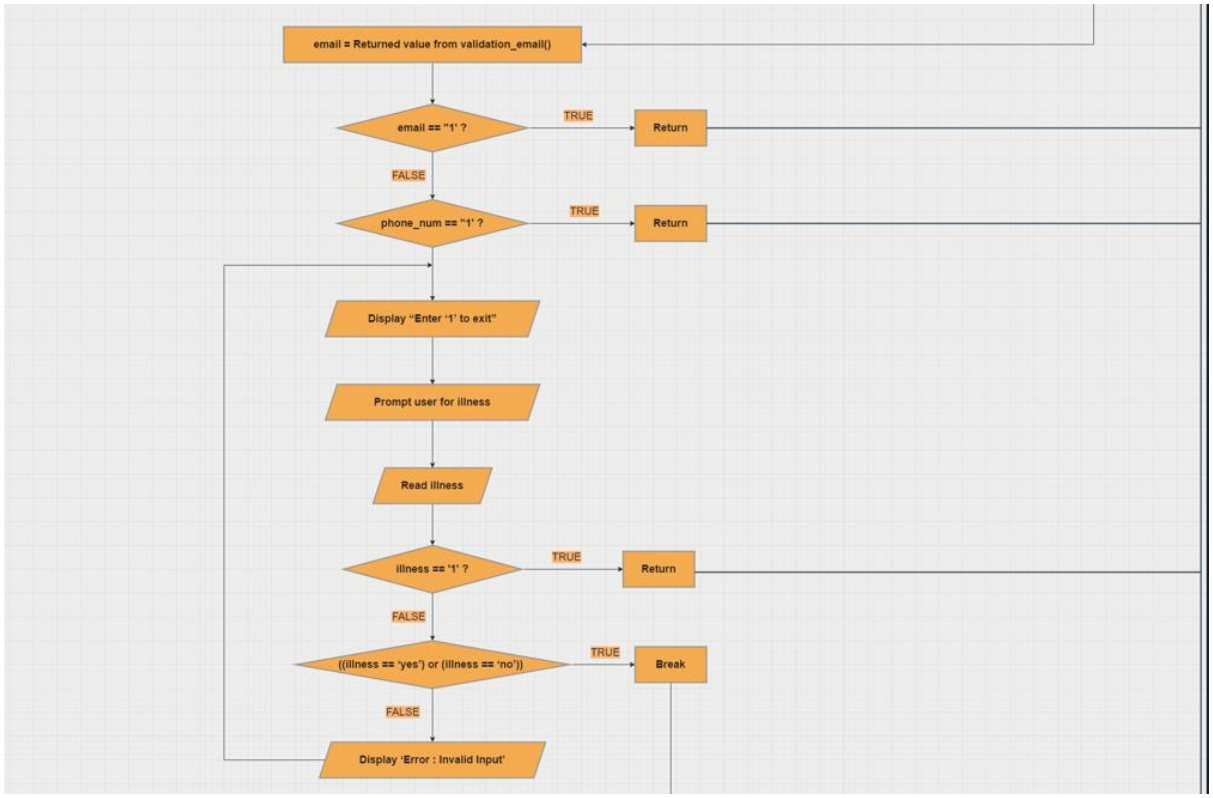

passw == '1' ?
TRUE
Return
FALSE
Open patients text file in read mode
flag = False
ic = Returned value from validation_ic()
ic == '1' ?
TRUE
Return
FALSE
lists = empty list
Is record line the last line ?
FALSE
Split the record in line by ','
Append the record into a list
Close patients text file


Open patients text file in read mode
Is the number of range of list length values the end ?
FALSE
Is (ic) in the list ?
flag = True
TRUE
Break
flag == True ?
TRUE
Display 'Error : IC exists, Please try again'
Continue
Close patient text file


Display "Enter '1' to exit"
Prompt user for gender
Read gender
Is ((gender == 'm') or (gender == 'f') or (gender == 'male') or (gender == 'female')) ?
TRUE
Break
FALSE
gender == '1' ?
TRUE
Return
FALSE
Display 'Error : Invalid input either male(m) or female(f)
Display "Enter '1' to exit"
Prompt user for address
Read address
address == '1' ?
TRUE
Return
FALSE
postcode = Returnted value from validation_postcode()


postcode == '1' ?
TRUE
Return
FALSE
Display "Enter '1' to exit"
Prompt user for vacciene centre


Read vc
vc == '1' ?
TRUE
Return
FALSE
Is ((vc == 'VC1') or (vc == 'VC2')) ?
TRUE
Break
FALSE
Display 'Error : Invalid Input'
Execute try clause
Is exception occurs ?
TRUE
Display 'Error : Invalid Input'
FALSE
Display "Enter '1' to exit"
Prompt user for age
Read age_int
age_int == 1 ?
TRUE
Return
FALSE
Is ((age_int > 11) and (age_int < 101)) ?
TRUE
age_str = converted from age_int into the form of a string
Break
FALSE
Display 'Error : Invalid Input'


email = Returned value from validation_email()
email == "1' ?
TRUE
Return
FALSE
phone_num == "1' ?
TRUE
Return
Display "Enter '1' to exit"
Prompt user for illness
Read illness
illness == '1' ?
TRUE
Return
FALSE
((illness == 'yes') or (illness == 'no'))
TRUE
Break
FALSE
Display 'Error : Invalid Input'

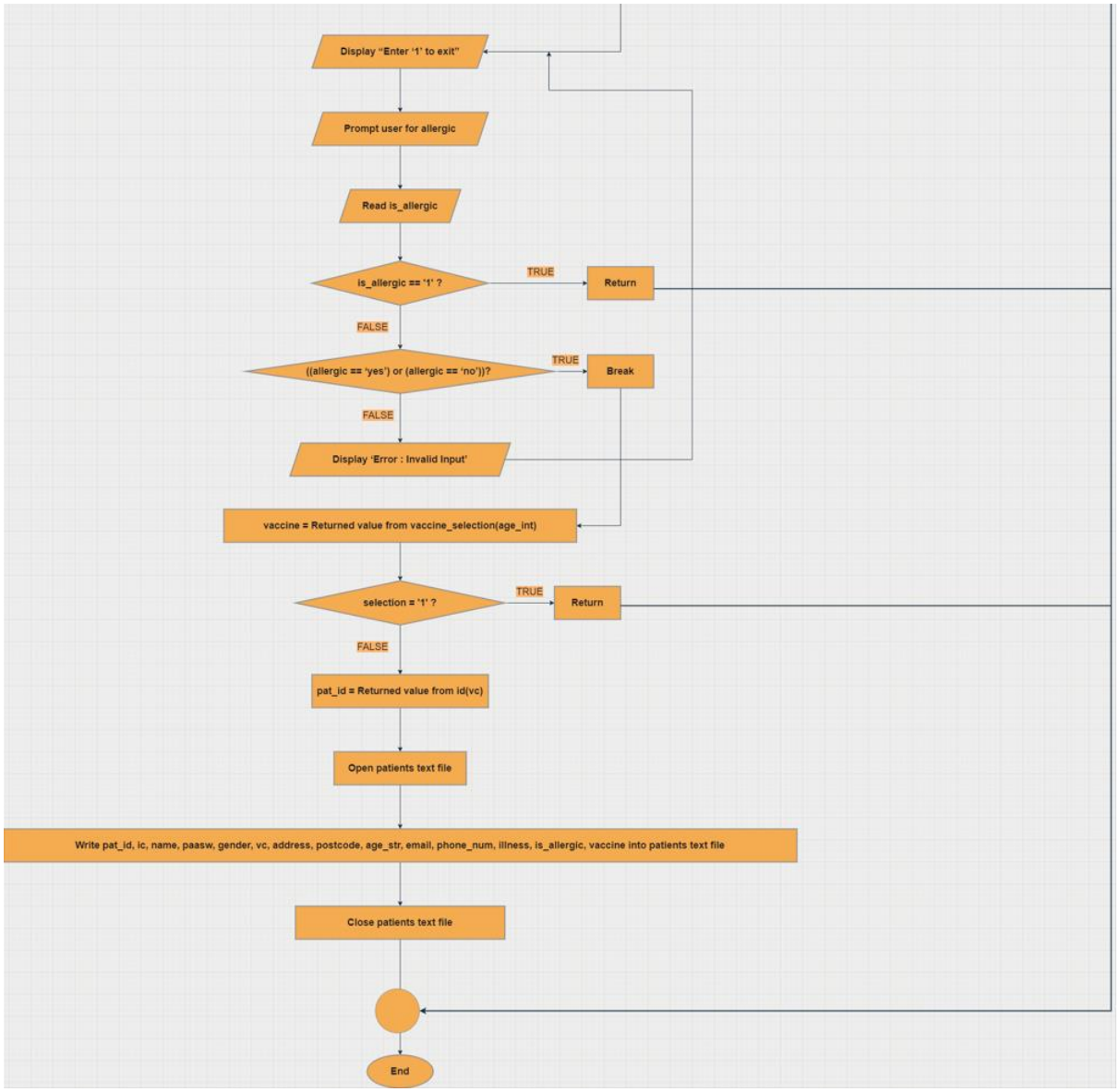


Figure 64: Flowchart new_user()

3.2.2 – HO FENG SHENG

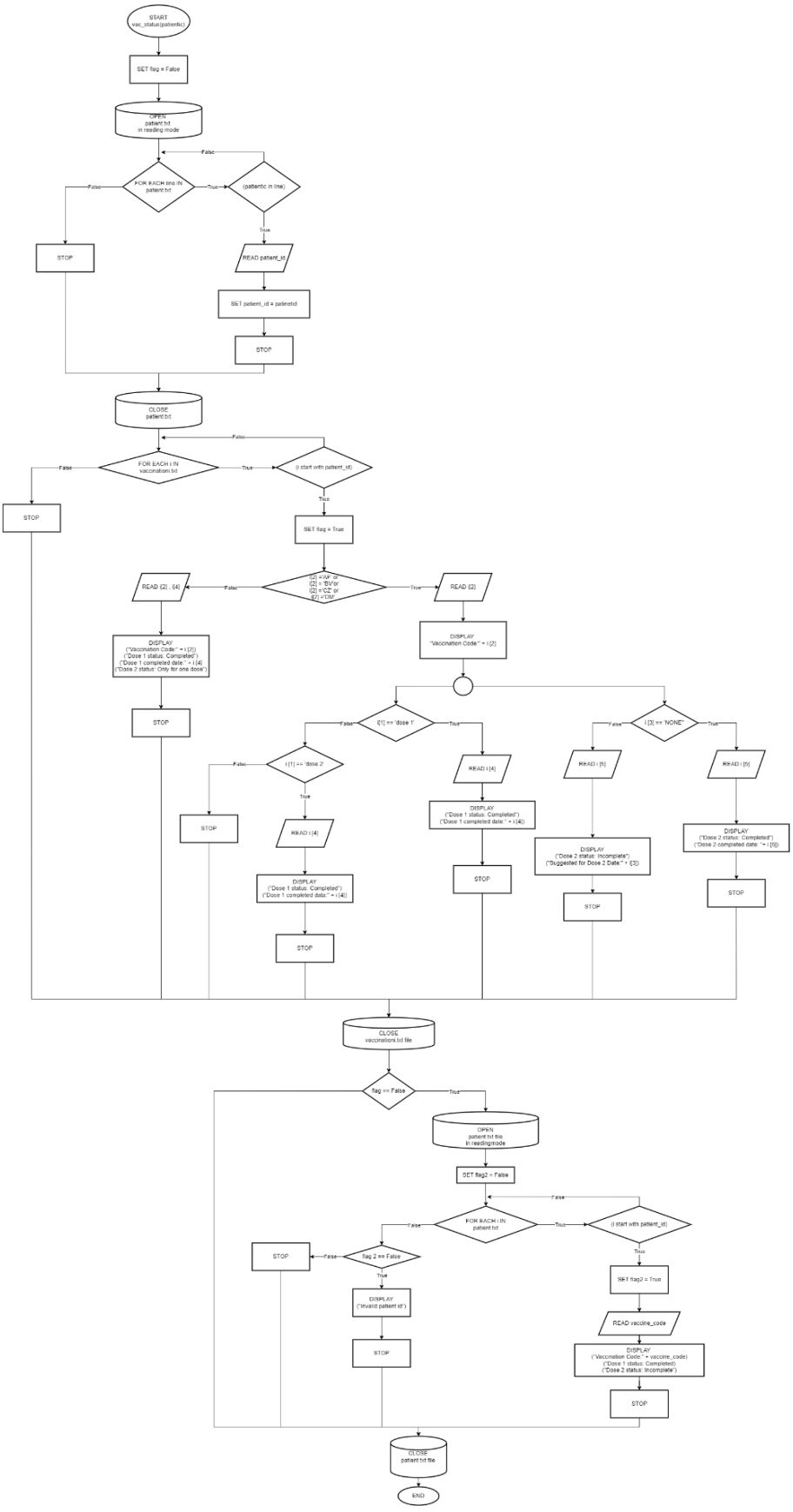


Figure 65: Flowchart of vac_status(patientic)

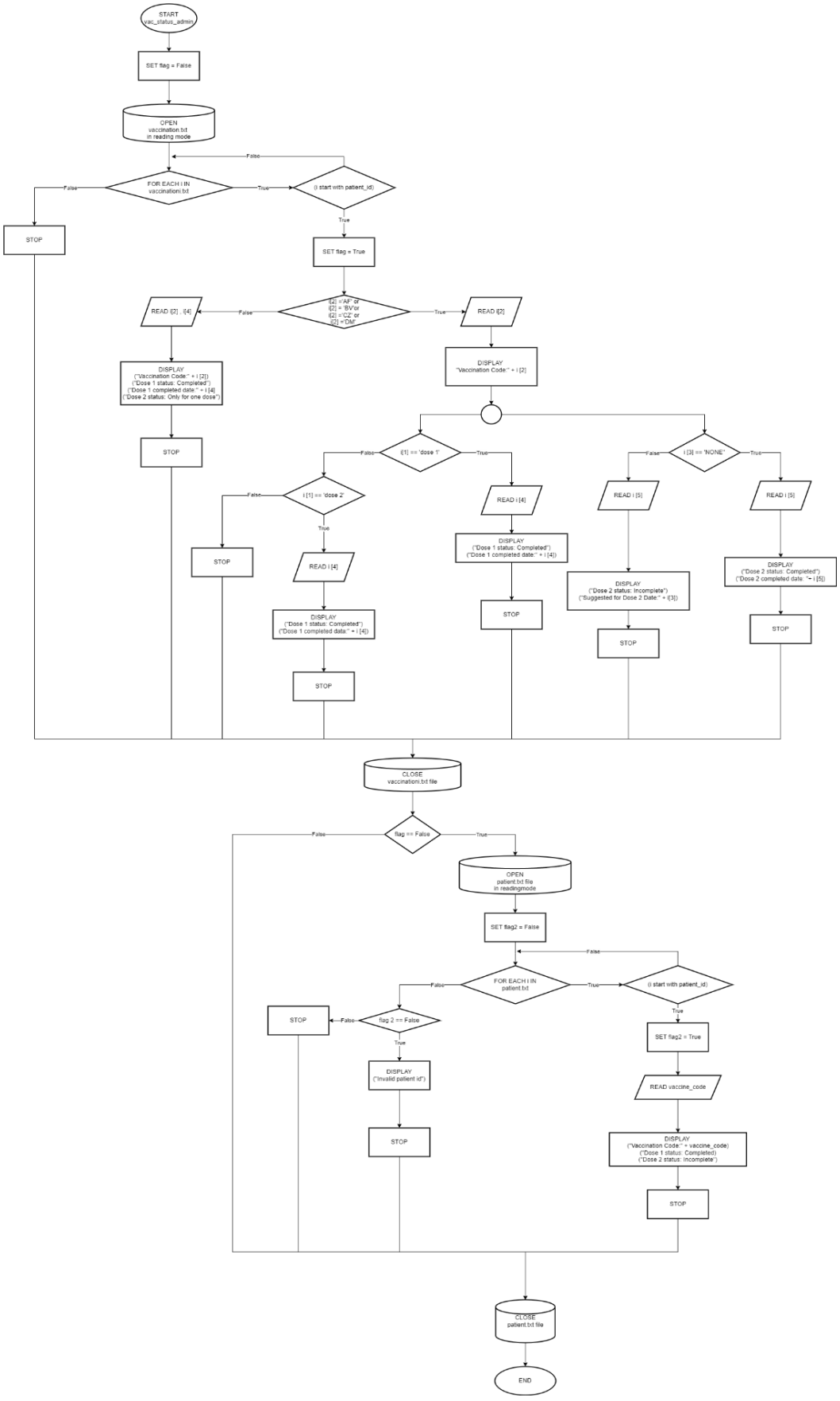


Figure 66: Flowchart of vac_status_admin

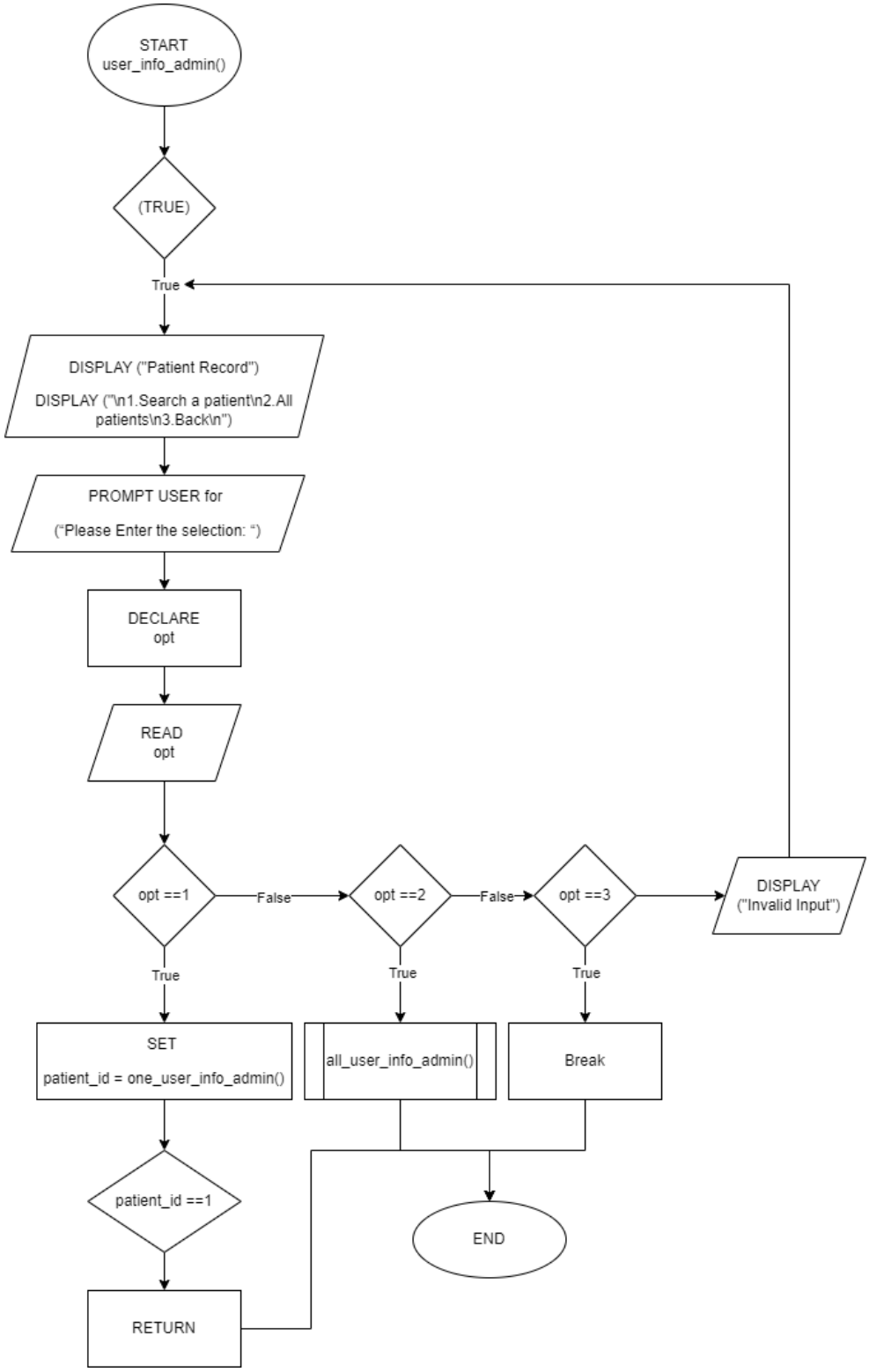


Figure 67: Flowchrrt of user_info_admin()

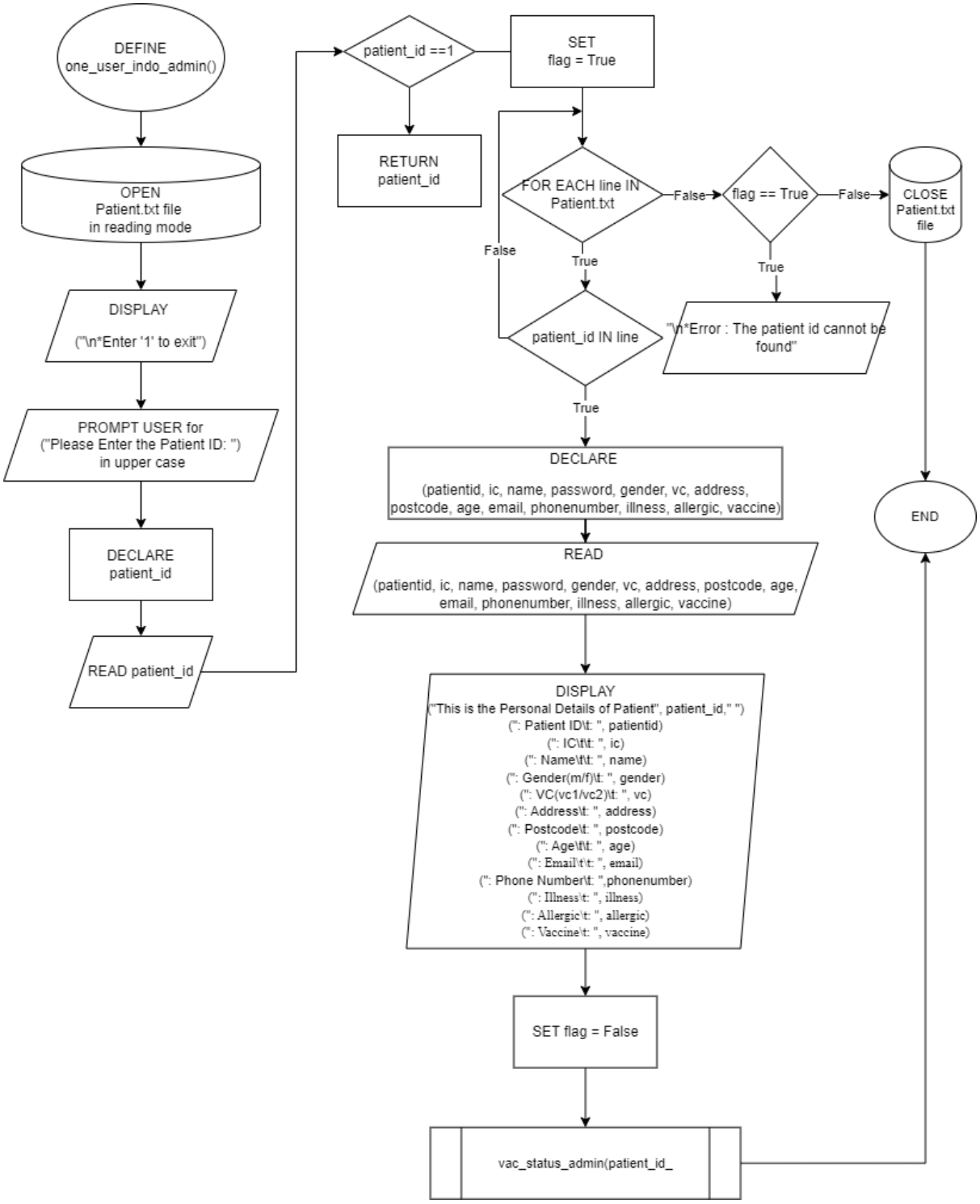


Figure 68: Flowchart of one_user_info_admin()

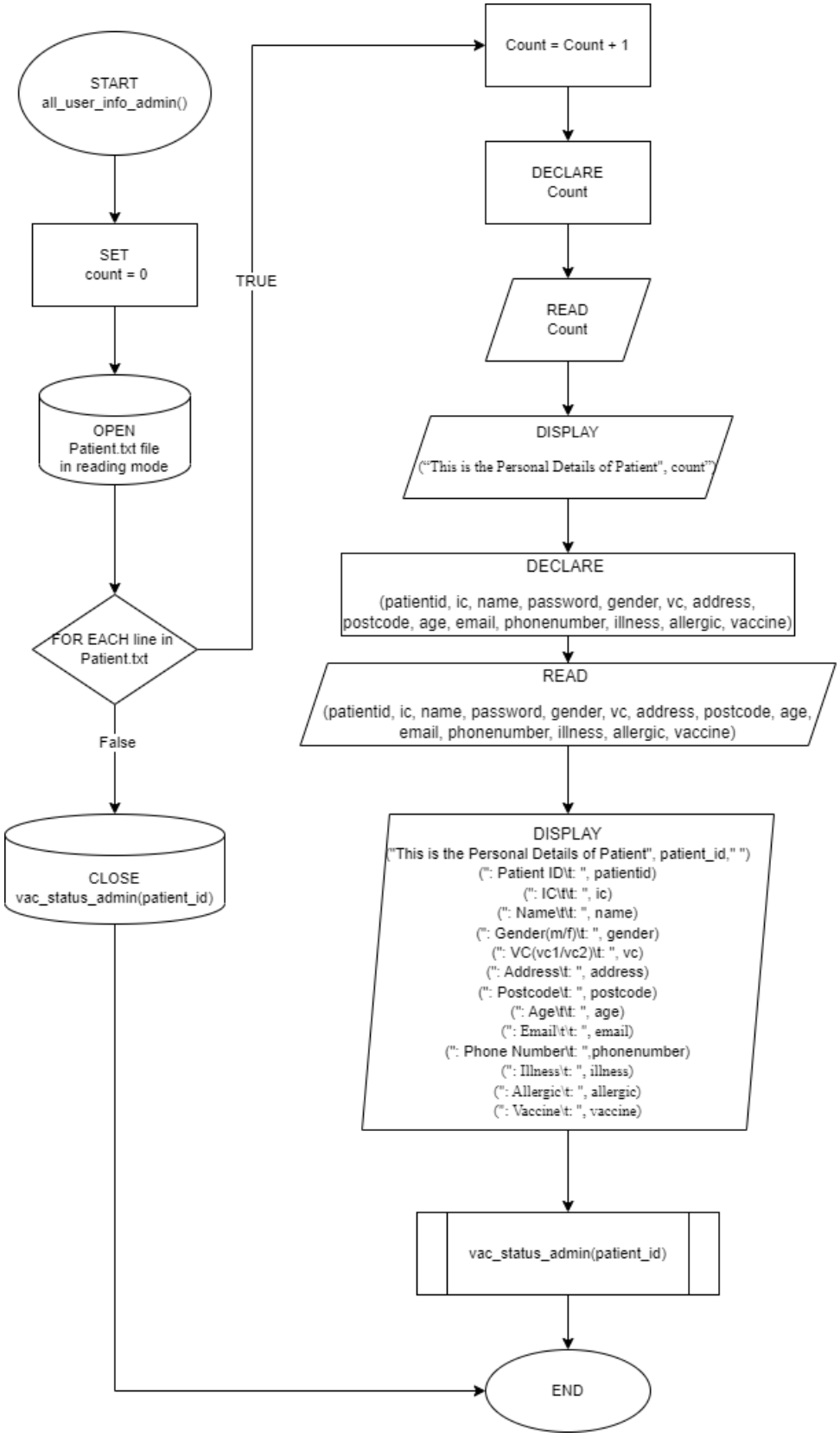


Figure 69: Flowchart of all_user_info_admin()

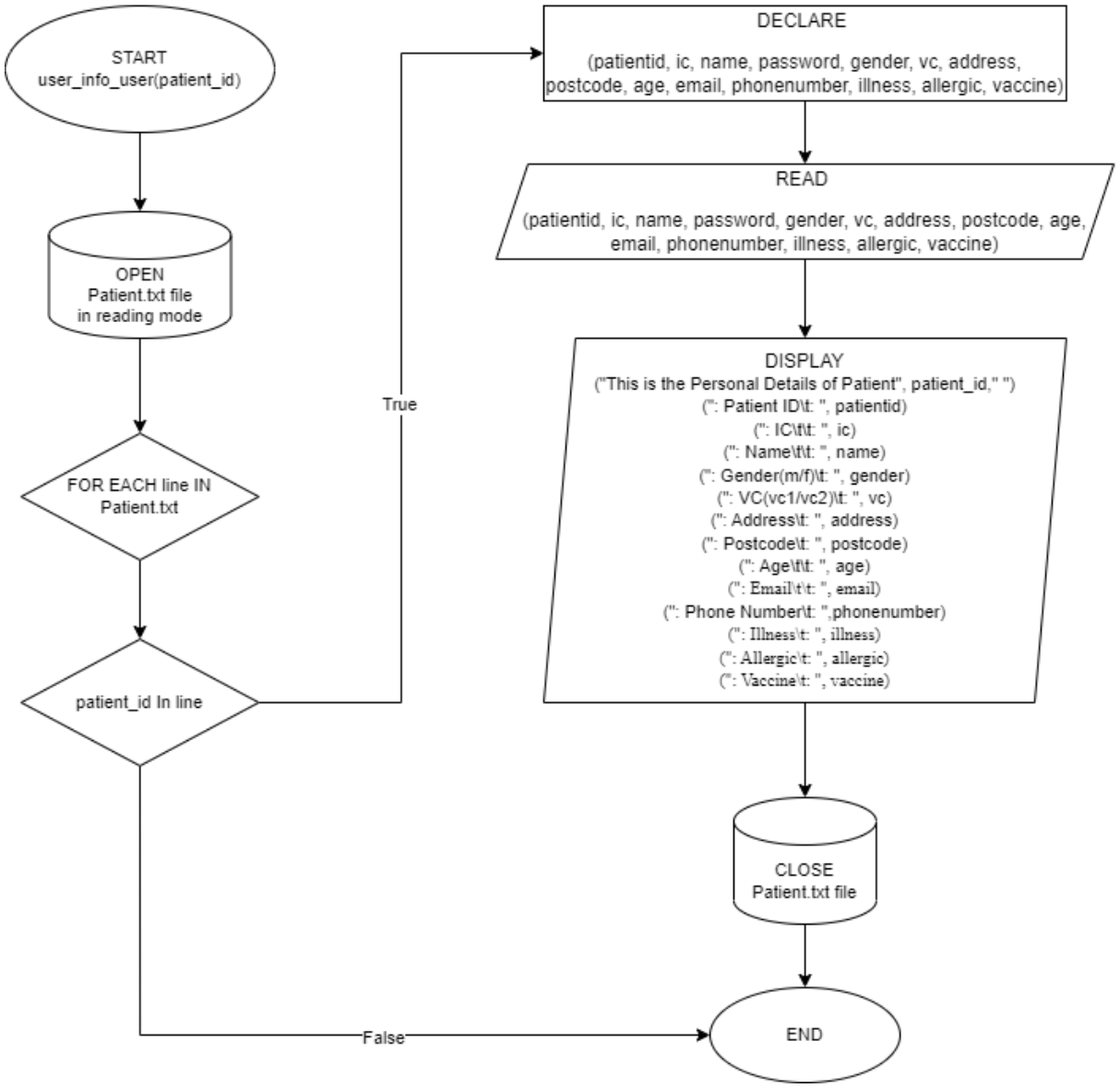


Figure 70: user_info_user(patient_id)

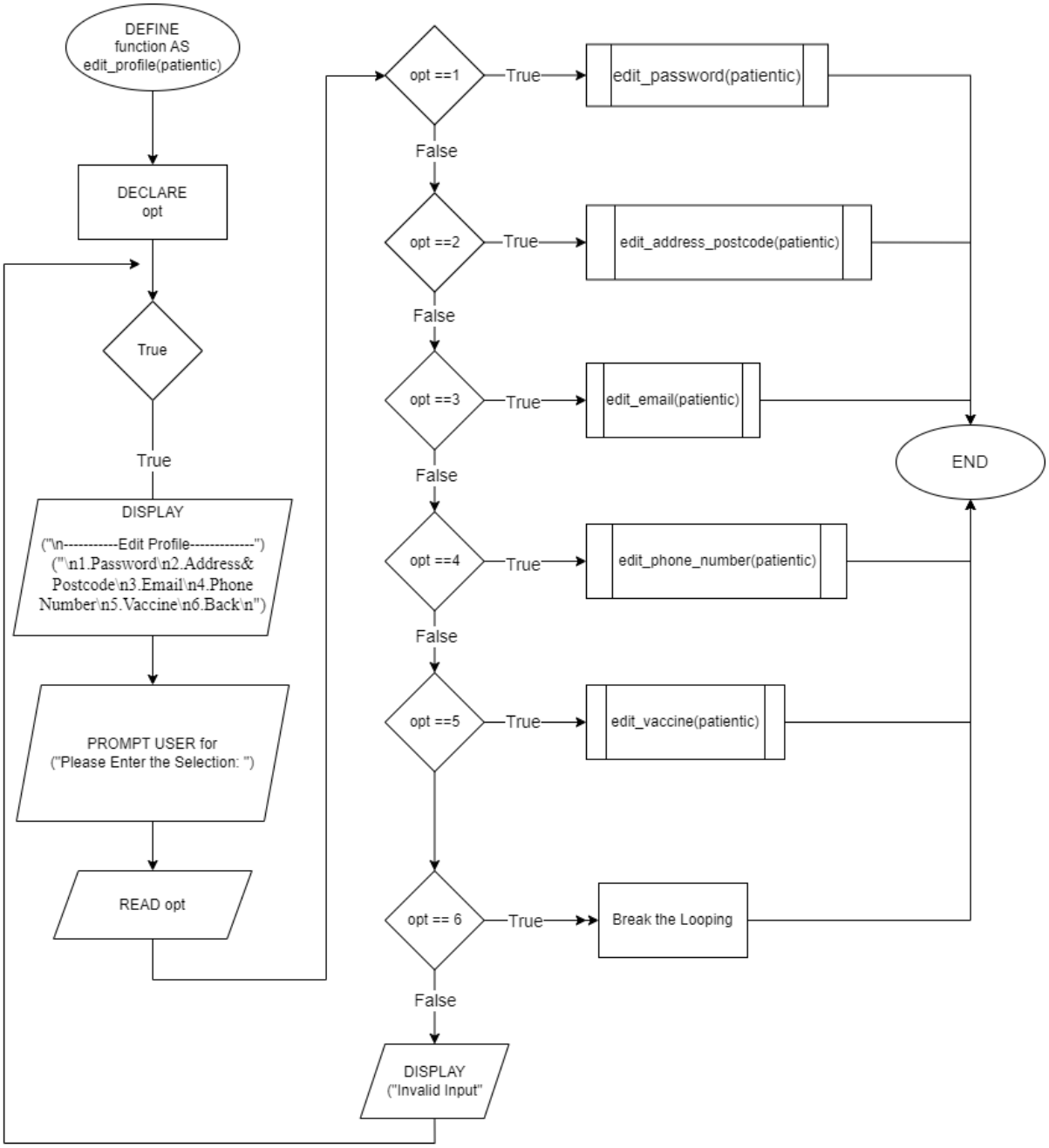


Figure 71: Flowchart of edit_profile(patientic)

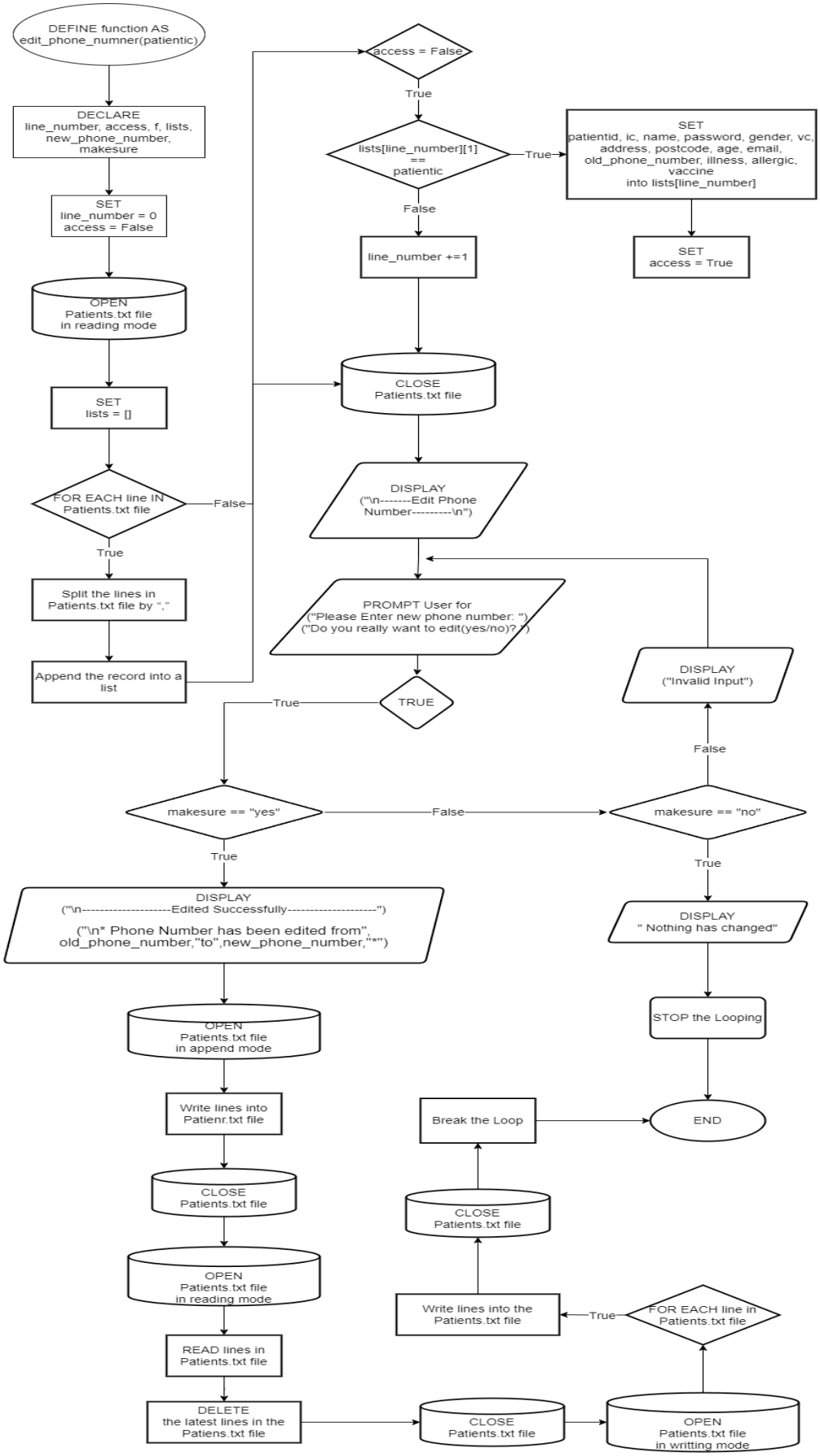


Figure 72:Flowchart of edit_phone_number(patientic)

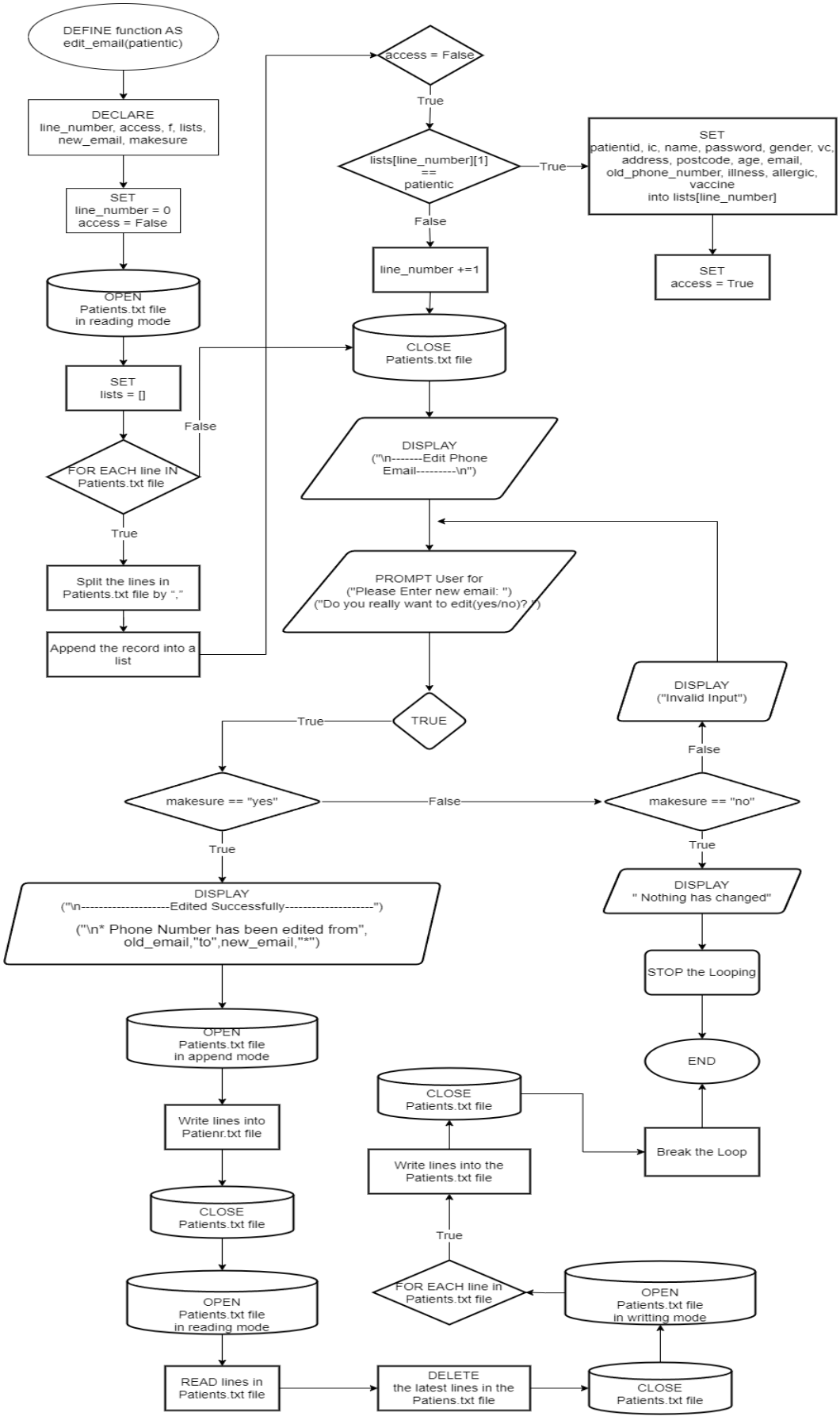


Figure 73: Flowchart of edit_email(patientic)

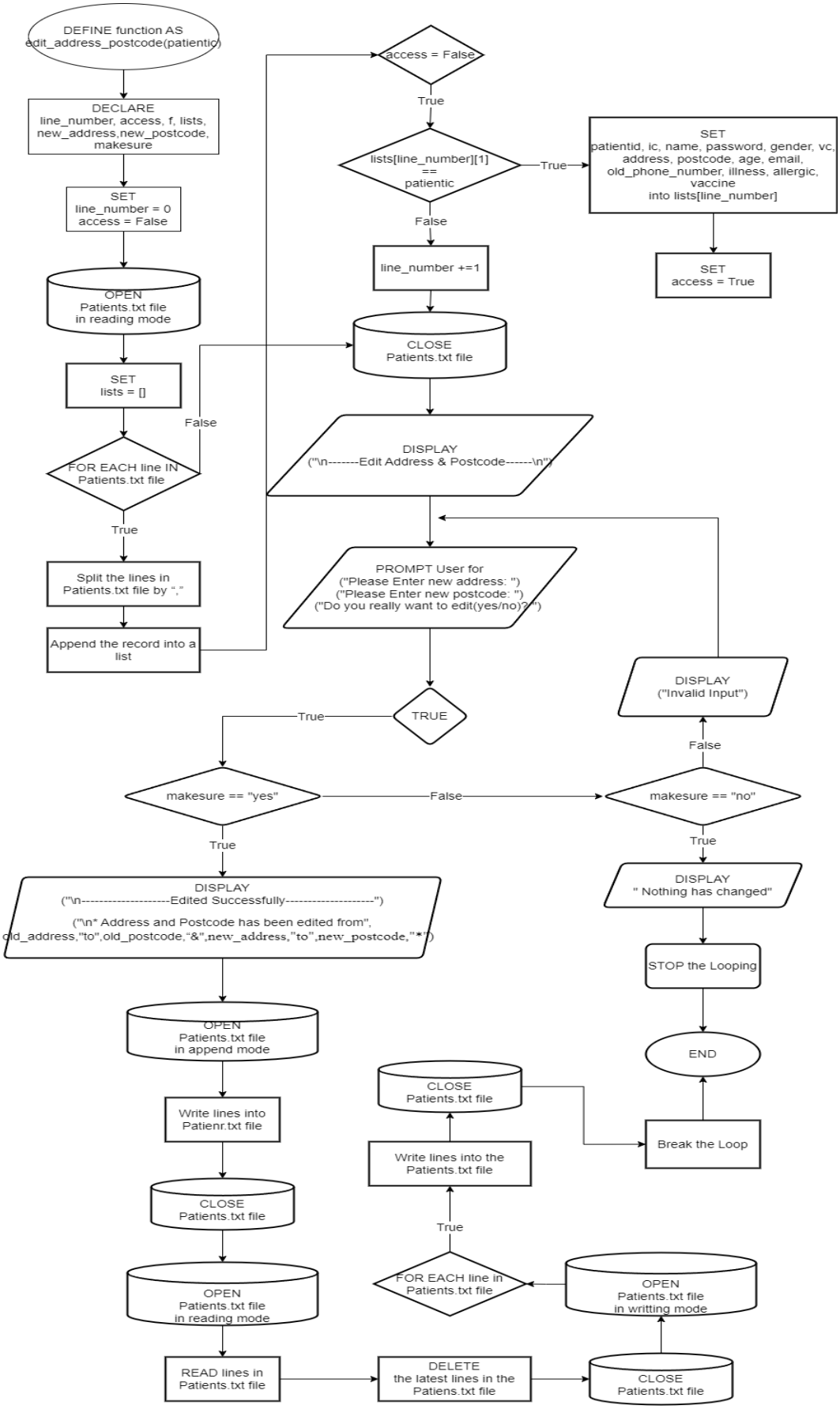


Figure 74: Flowchart of edit_address_postcode(patientic)

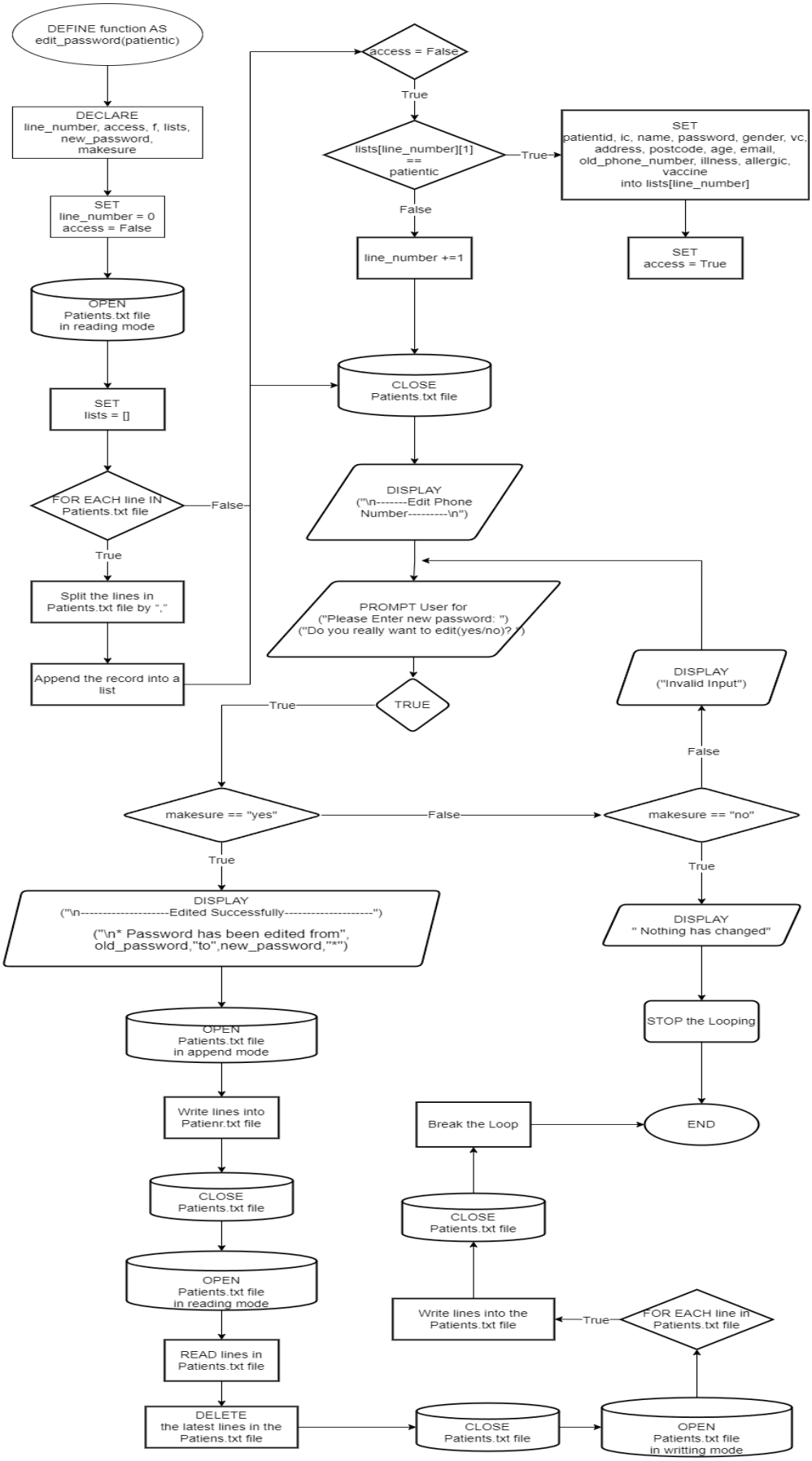


Figure 75:Flowchart of edit_password(patientic)

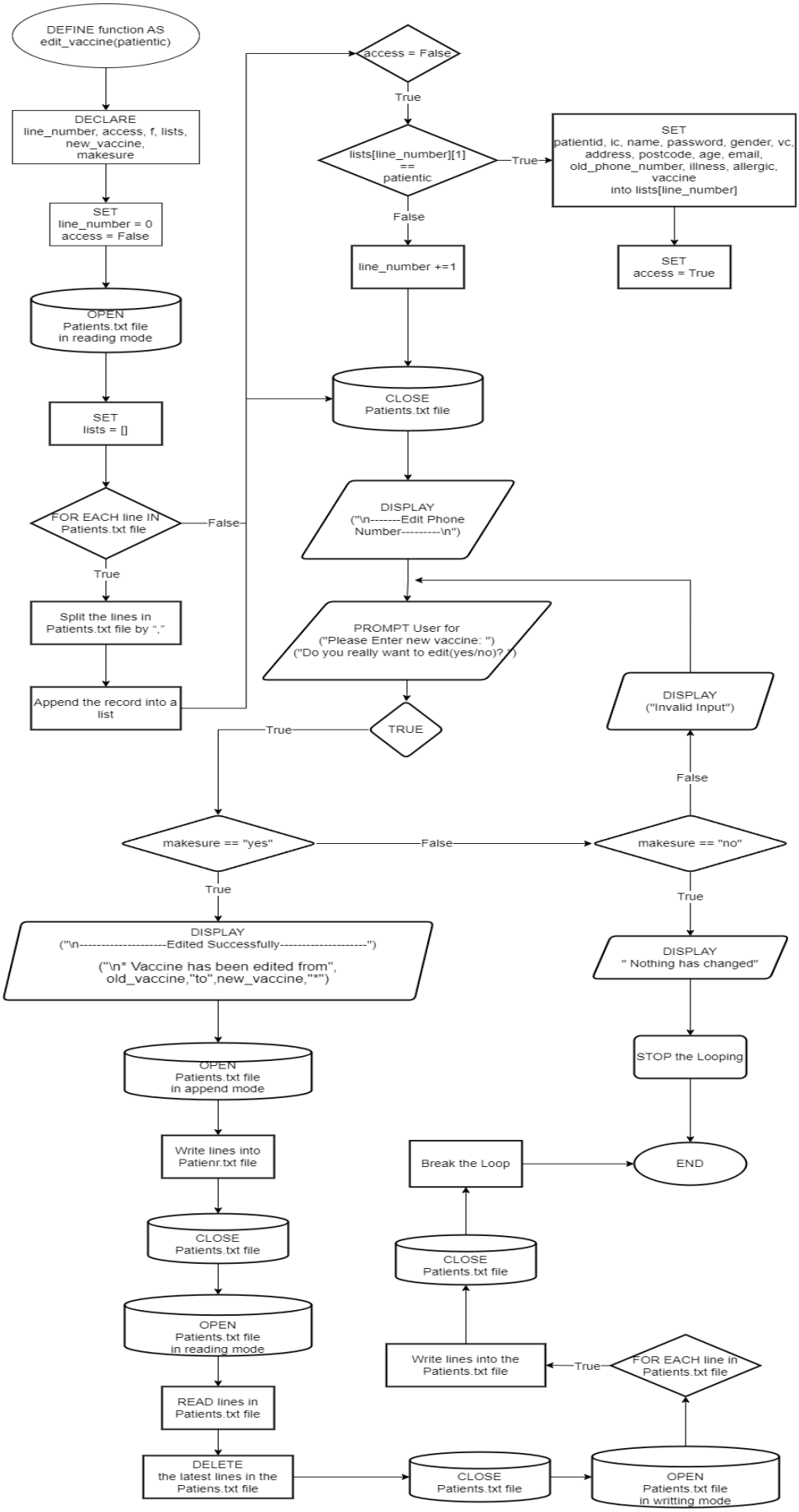


Figure 76: Flowchart of edit_vaccine(patientic)

3.2.3 – DYANIEL CHING CHEE XIONG

Make sure key in valid patient id

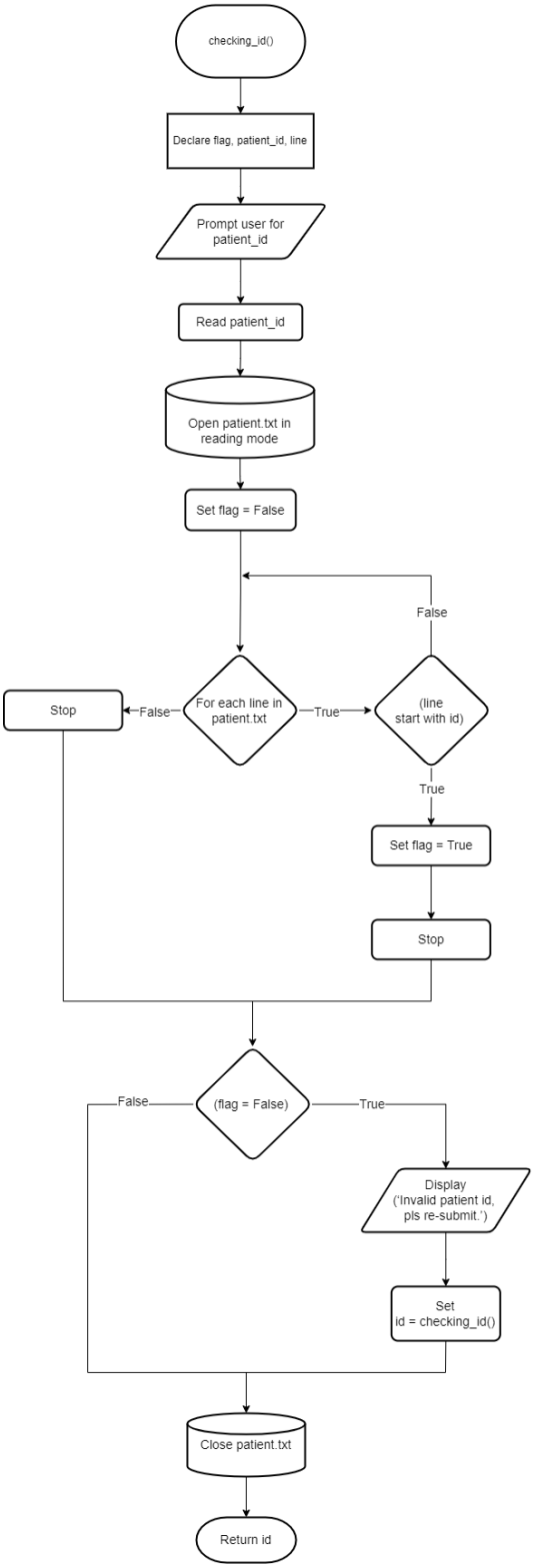


Figure: flowchart of checking_id() function

Determine patient age

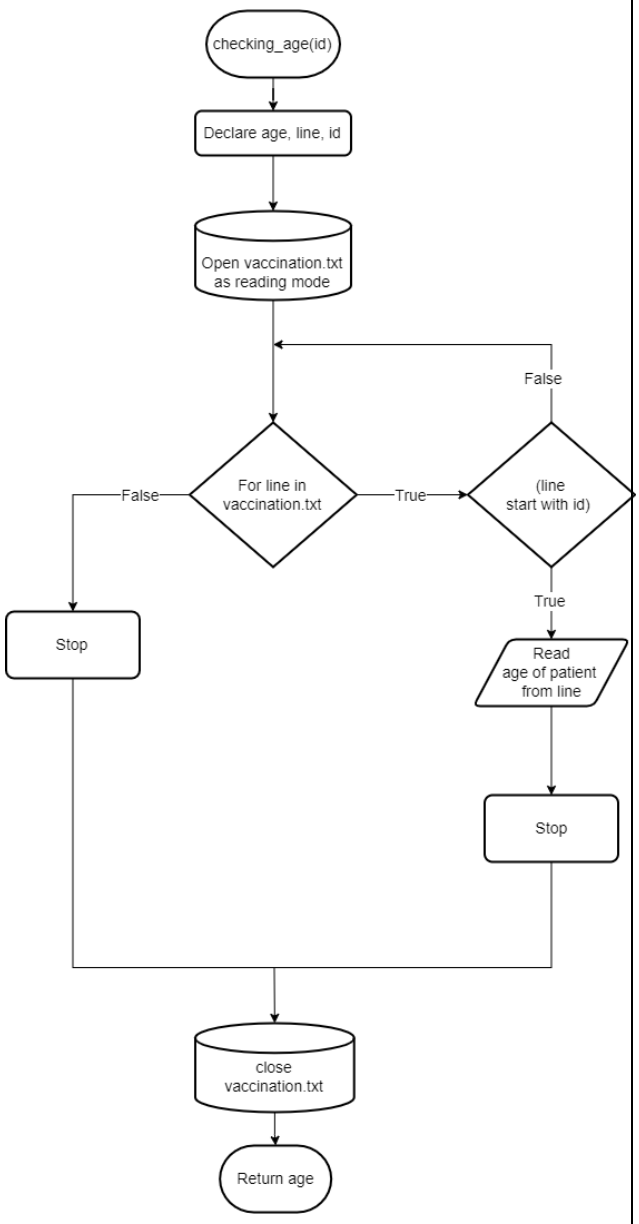


Figure: flowchart of checking_age(id) function

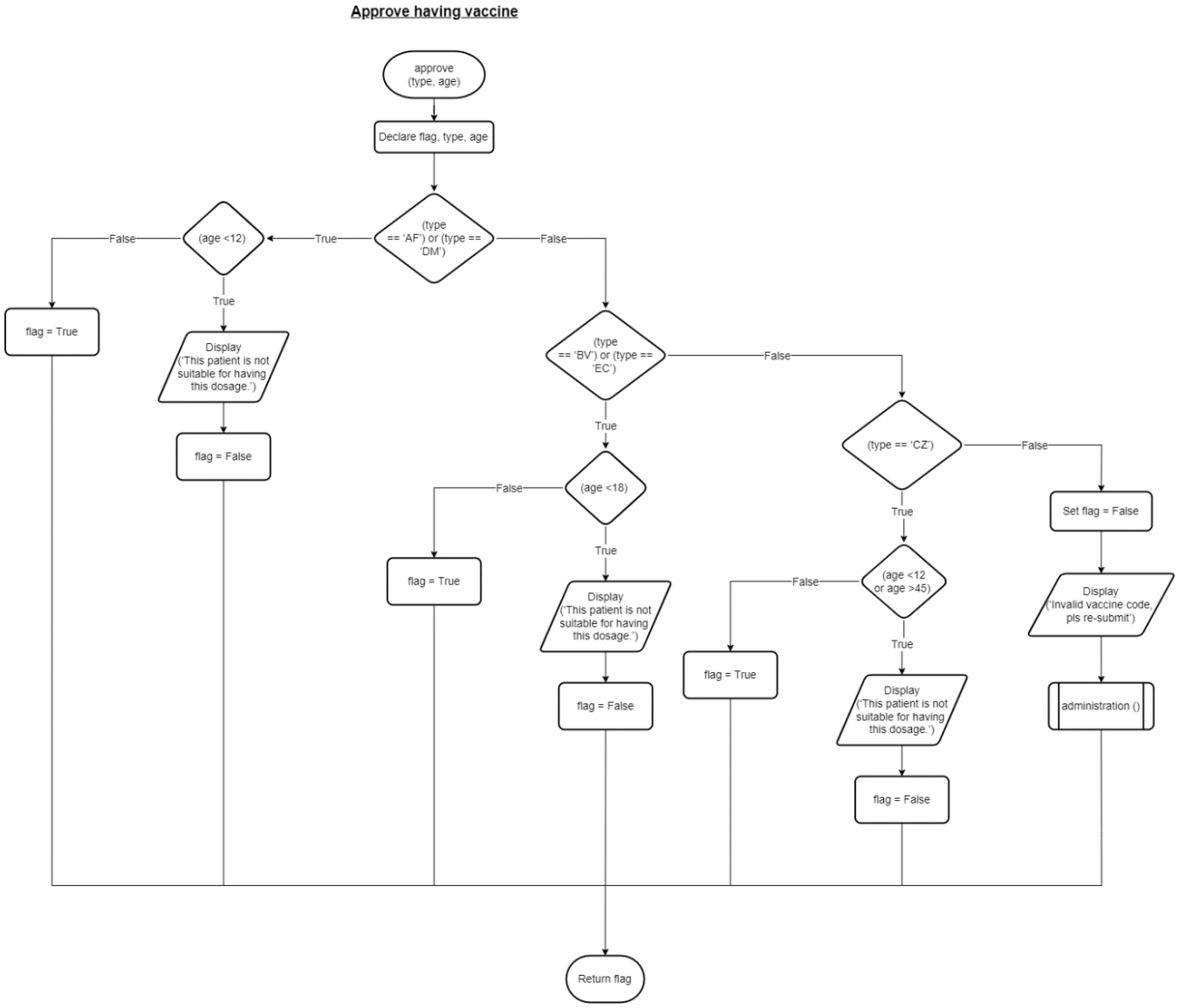


Figure: flowchart of approve(type, age) function

Check vaccine code key in same with vaccine code in Patients text file

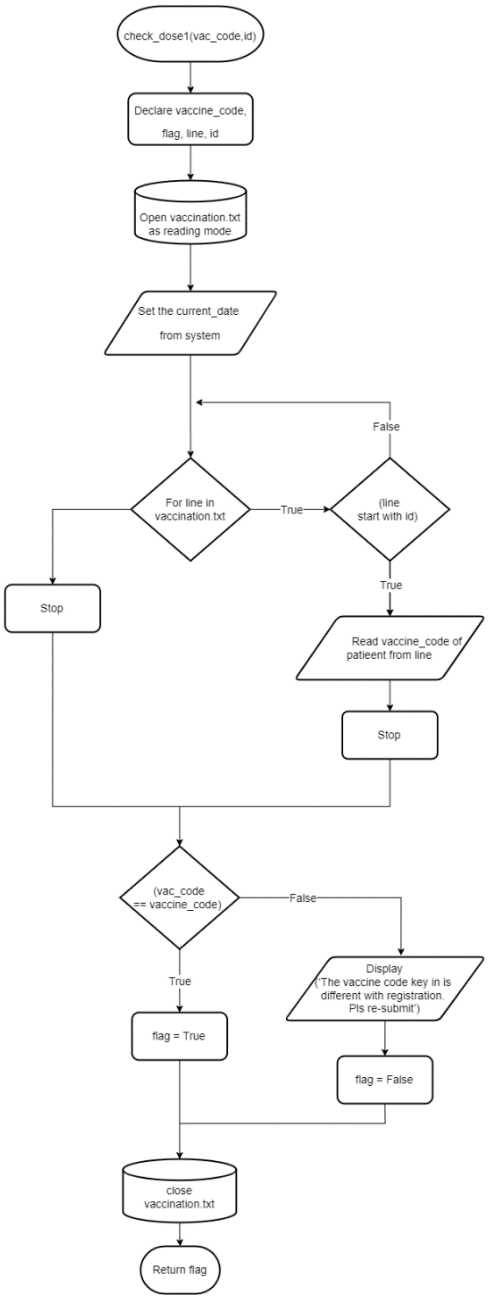


Figure: flowchart of check_dose1(vac_code,id)function

Confirm patient reach 2nd dose date

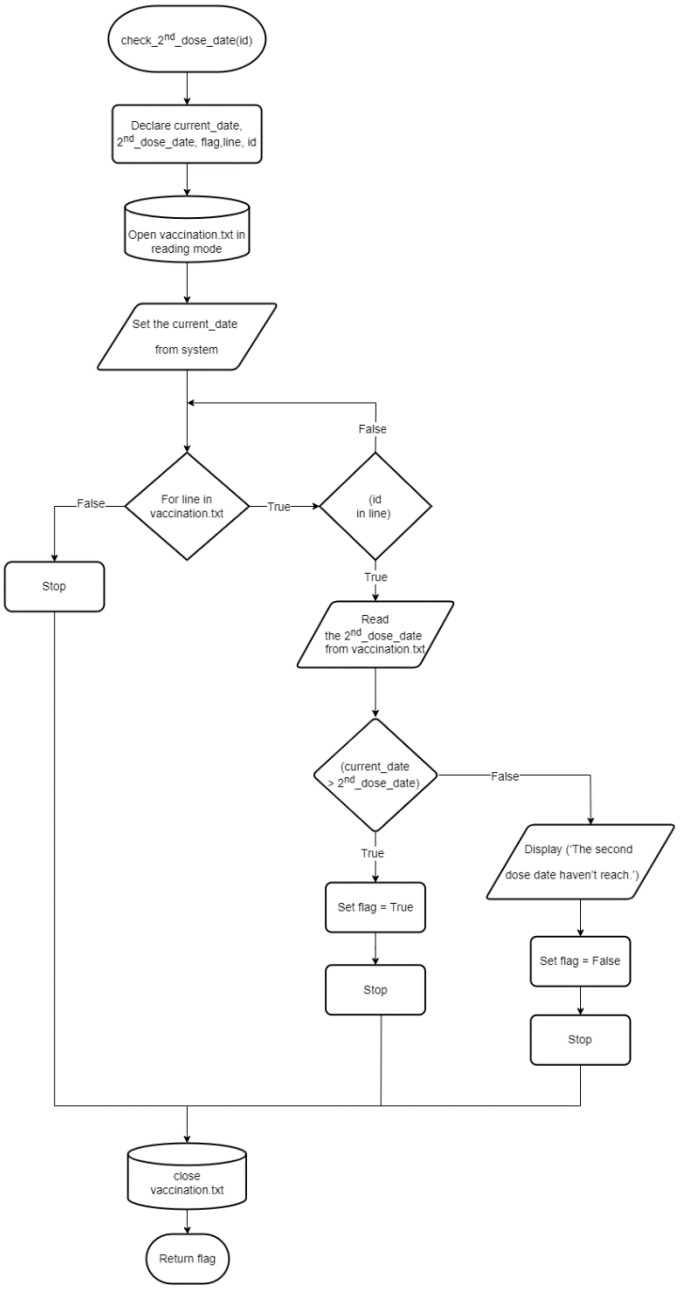


Figure: flowchart of check_2nd_dose_date(id) function

Check vaccine code of 2nd dose same with 1st dose

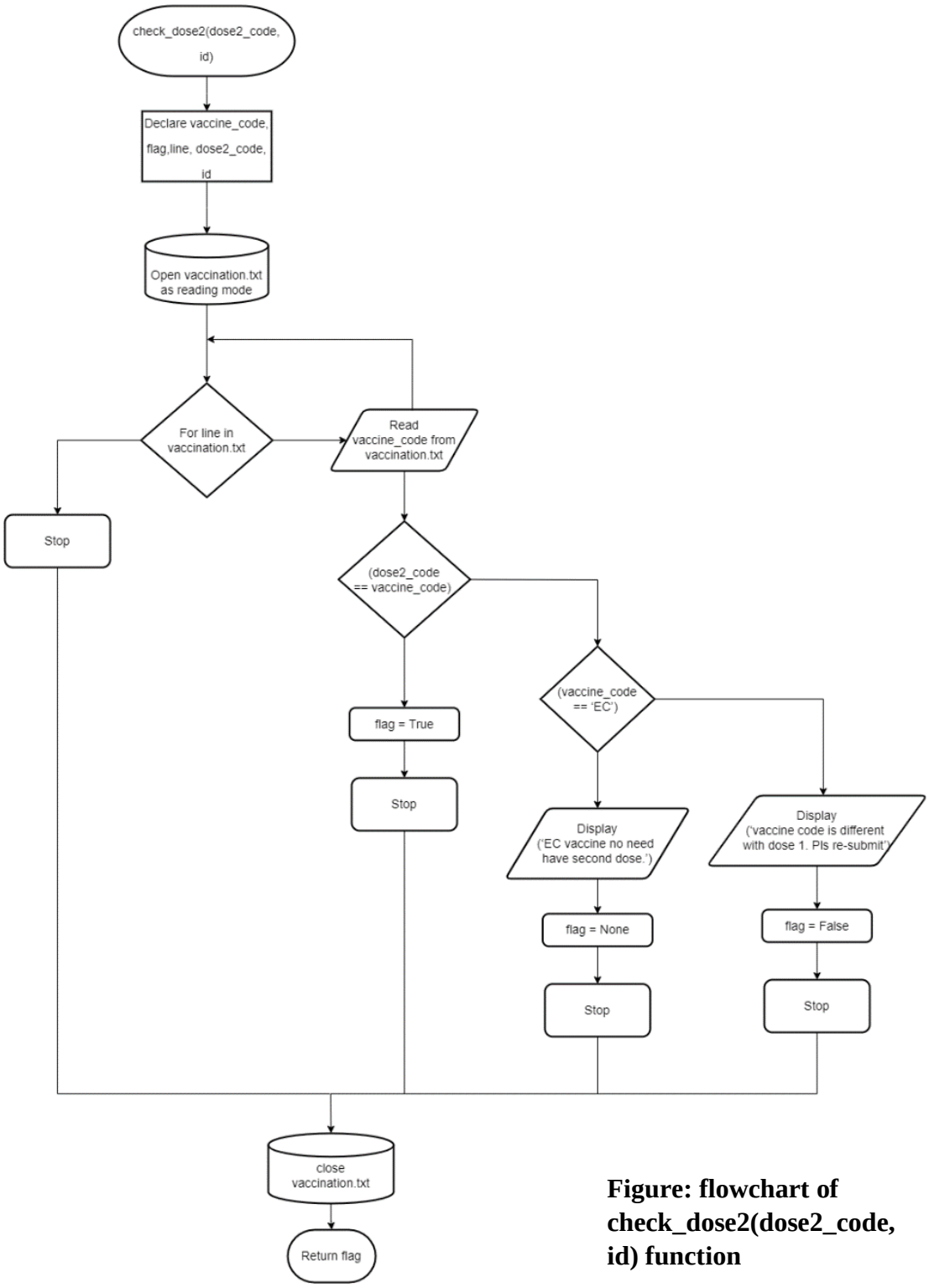


Figure: flowchart of check_dose2(dose2_code, id) function

Dose 1 step

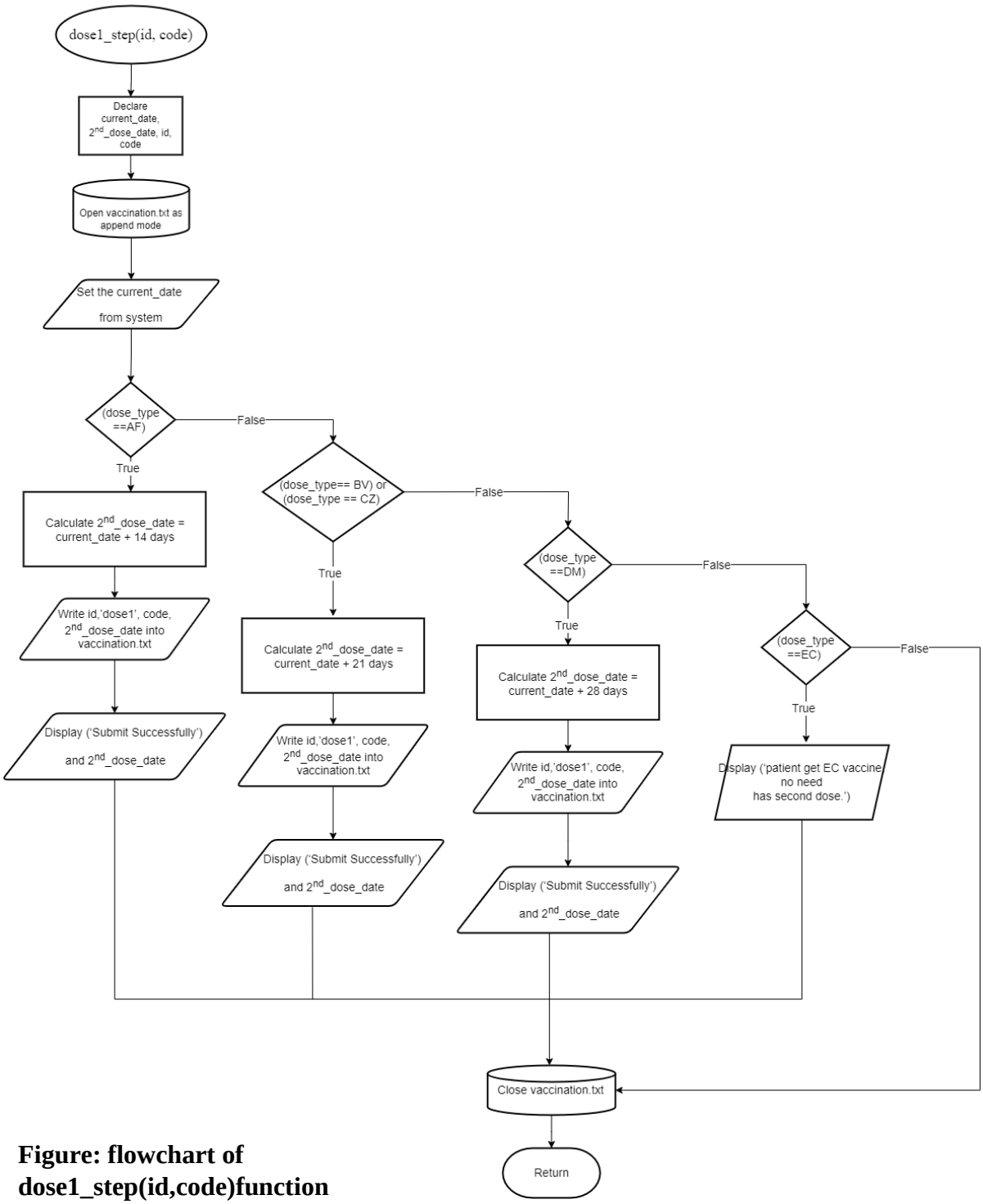


Figure: flowchart of dose1_step(id,code)function

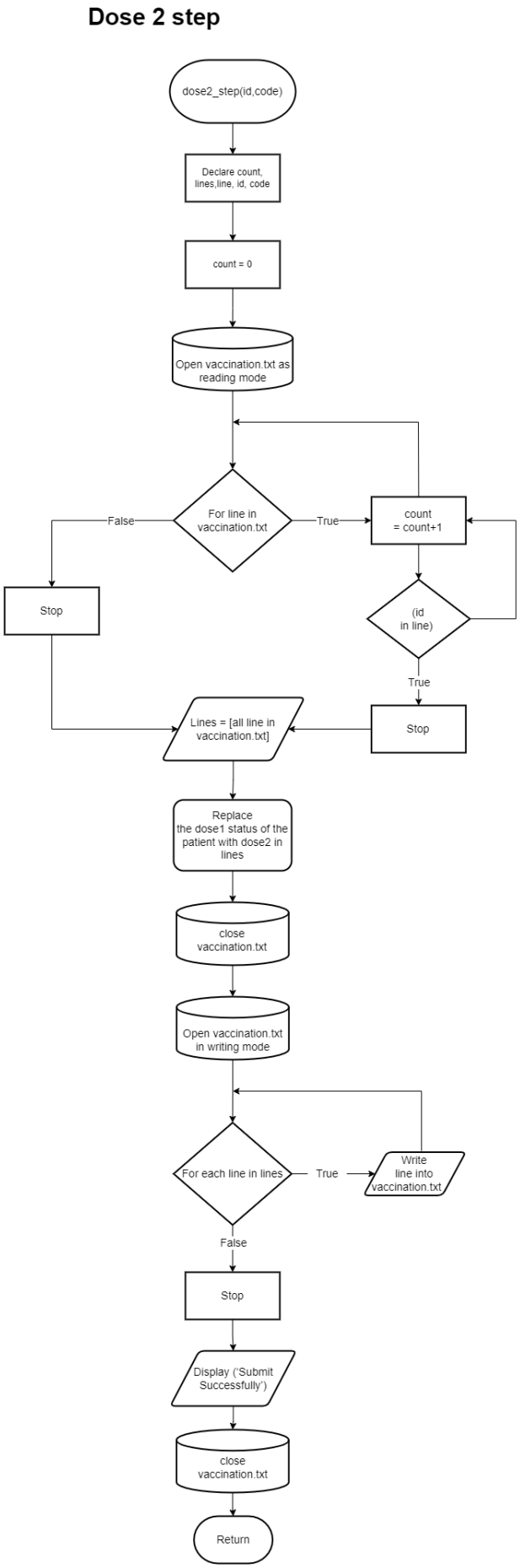


Figure: flowchart of dose2_step(id,code)function

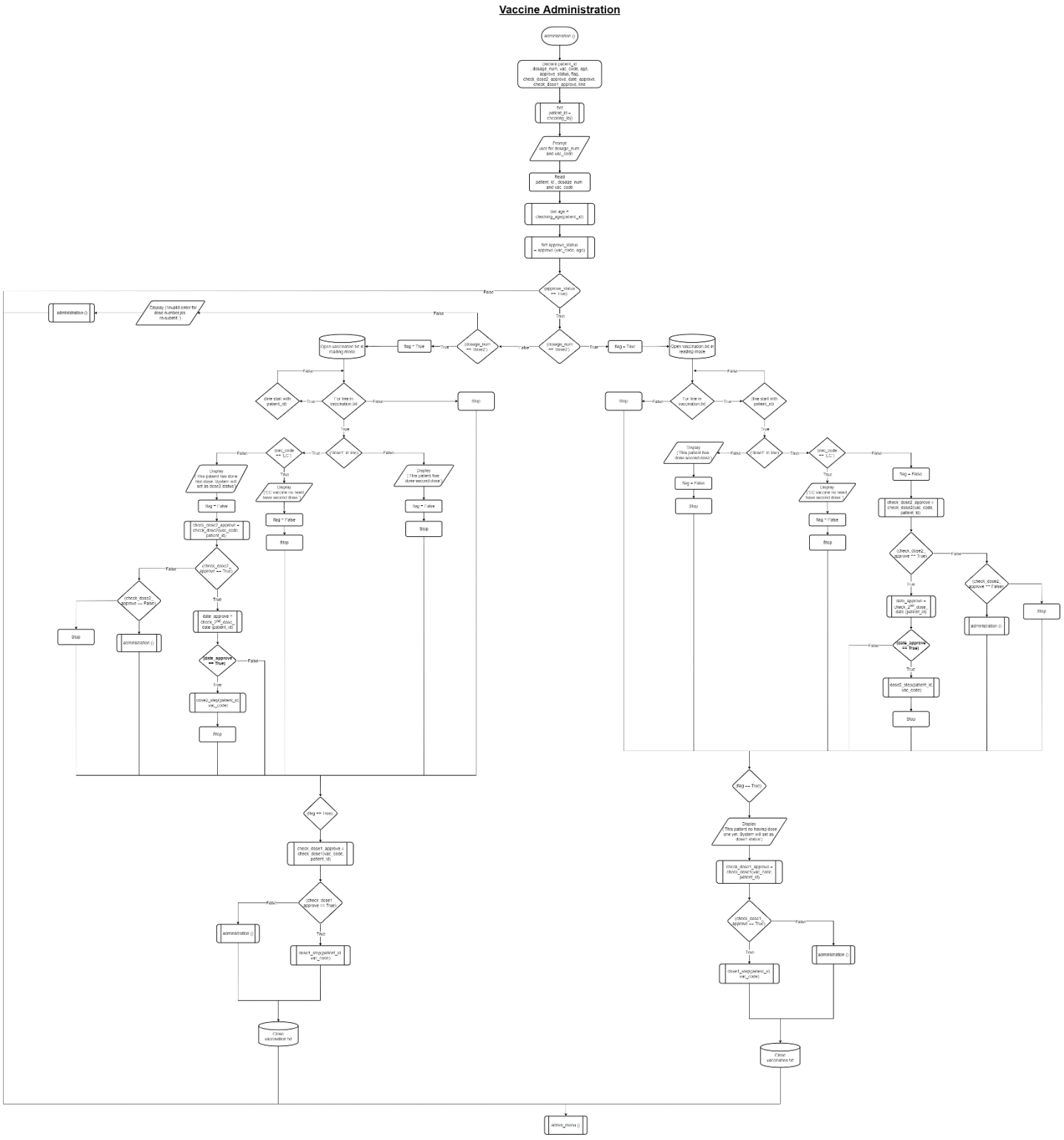


Figure: flowchart of administration() function

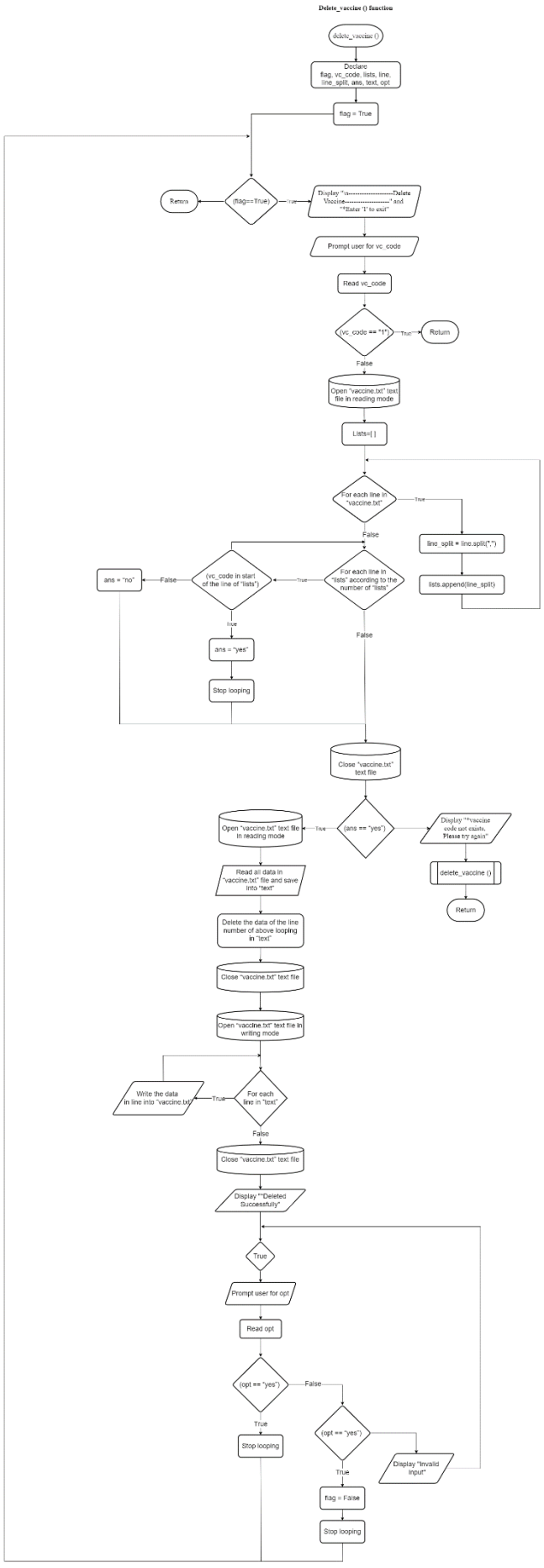


Figure: flowchart of delete_vaccine ()

4.0 Program Source Code & Explanation

4.1 - GAN MING HUI

```
1470 #DONE BY GAN MING HUI
1471 #Define a main menu for the system
1472
1473 def menu():
1474     print("\n--------------------Malaysia Vaccination System--------------------")
1475     print("\nWelcome Back :D\n")
1476     print("Do you want to enter as ? : \n")
1477     print("1.Patient\n2.Admin\n3.Close")
1478     selection = input("\nPls enter the selection (1-3) : ")
1479
1480     if(selection == "1"):
1481         type_user() #USER LOGIN / REGISTER
1482     elif(selection == "2"):
1483         admin_acc() #ADMIN LOGIN
1484     elif(selection == "3"):
1485         exit() #STOP ENTIRE PROGRAM
1486     else:
1487         print("\nError : Invalid Input")
```

Figure 77: menu( )

Variables / Functions in menu():

Variable / Function	Meaning
selection	A variable that saves the input of selection from the user (1-3)
type_user( )	A function that allows users to log in or register as a patient
admin_acc( )	A function that allows users to log in as an admin
exit( )	To terminate the entire program

Explanation for menu( ):

The **'menu'** function will be the main menu for the Malaysia vaccination system. The **main purpose** of this function is to enable users to choose whether they want to enter the system as a patient or as an admin. In this function, the users will be asked to select the '1' option to enter the system as a patient, select '2' to enter the system as an admin, or select '3' to exit the program.

To build this function, the 'Malaysia Vaccination System' and 'Welcome Back :D' will be printed in line **1474** and **1475** respectively. In addition, in lines **1476-1477** the users are prompted to enter a choice of what identity they want to enter the system as. In line **1478** the users' input selection will be read and stored in the 'selection' variable. In lines **1480-1487** if...elif...else statement is used, if the users enter '1', they will be taken to the 'type_user' function (refer to figure 78), if the users enter '2', they will be taken to the 'admin_acc' function

and if the users enter ‘3’, the entire program will be terminated, except for 1-3 inputs, ‘Invalid Input’ will be printed.

```
378 #DONE BY GAN MING HUI
379 #Define is new account or existing account
380
381 def type_user():
382     while(True):
383         print("\n--------------------Patient Account--------------------")
384         print("\n1.Create New Account\n2.Login into Existing Account\n3.Back\n")
385         user_acc = input("Pls enter the selection : ")
386
387         # if users enter 1, will lead them to do new registration and the patient id will be assigned to them
388         if (user_acc == "1"):
389             new_user()
390             break
391
392         # if users enter 2, will lead them to log in
393         elif (user_acc == "2"):
394             exist_user()
395             break
396
397         elif (user_acc == "3"):
398             return #STOP THE FUNCTION
399
400         else:
401             print("\n*Error : Invalid Input")
```

Figure 78: type_user( )

Variables / Functions in type_user( ):

Variable / Function	Meaning
user_acc	A variable that saves the input of selection from the user (1-3)
new_user( )	A function that allows patients to register a new account
exist_acc( )	A function that allows patients to login into an existing account
return	Used to stop the ‘type_user’ function

Explanation for type_user( ):

The **main purpose** of the 'type_user' function is to allow patients to create new accounts or login into their accounts. In this function, users will be given the option to create a new account by choosing ‘1’, log in to their account by choosing ‘2’, or go back to the previous section by choosing ‘3’.

To build this function, a while loop will be used on lines 382-401. Under the while loop, ‘Patient Account’ will be printed in line **383**. Also on lines **384**, users are prompted to enter an option to either create a new account or log into an existing account. The selection of users will be read and stored in the 'user_acc' variable on line **385**. In lines **388-401** if...elif statement is used, if the users enter '1', they will be taken to the 'new_user' function (refer to figure 81), if the users enter '2', they will be taken to the 'exist_user' function (refer to figure 108) and if the

users enter '3', they will return to the previous page, which is the interface to choose what identity to enter the system as.

```
#DONE BY GAN MING HUI
#Define a menu for user (what they can do after they login)

def user_menu(patientic):
    while (True):
        print("\n1.View Profile\n2.Edit Profile Details\n3.View Available Vaccines\n4.View Vaccination Status\n5.Log Out\n")
        opt = input("Pls enter the selection : ")
        print("")

        #link to each of the function
        if (opt == "1"):
            user_info_user(patientic)

        elif (opt == "2"):
            edit_profile_menu(patientic)

        elif (opt == "3"):
            vaccine_details()

        elif (opt == "4"):
            vac_status(patientic)

        elif (opt == "5"):
            print("*You have been successfully logged out.")
            break

        else:
            print("Invalid Input")
```

Figure 79: user_menu( )

Variables / Functions in user_menu( ):

Variable / Function	Meaning
opt	A variable that saves the input of selection from the user (1-5)
user_info_user(patientic)	'patientic' is an argument that passes the value back to the parameter of 'user_info_user' function. This function allows users to view the profile that contains their personal details.
edit_profile_menu(patientic)	'patientic' is an argument that passes the value back to the parameter of 'edit_profile_menu' function. This function allows users to edit their personal details
vaccine_details()	A function that allows users to view all the available vaccines
vac_status(patientic)	'patientic' is an argument that passes the value back to the parameter of 'vac_status' function. This function allows users to view their vaccination status

Explanation for user_menu( ):

The 'user_menu' function is a menu that contains five functions. Its **main purpose** is to let the user choose what they want to do when they are successfully logged in. In this function, the users will be asked to select the '1' option to view their profile, select '2' to edit their profile,

select ‘3’ to view all available vaccines, select ‘4’ to view their vaccination status or select ‘5’ to logout from their account.

To build this function, at the beginning of line **408** the while loop will be used. Under this loop which is in lines **409-431**, on lines **409**, the users are prompted to enter a choice of whether they want to see the profile, edit the profile, view available vaccines, view vaccination status, or log out. Next, the selection of users will be read and stored in the ‘opt’ variable on line **410**. In lines **414-431** if...elif...else statement is used, if the user enters '1', they will be taken to the 'user_info_user' function (refer to figure 107), if the users enter '2', they will be taken to the 'edit_profile_menu' function (refer to figure 87), if the users enter '3', they will be taken to the ‘vaccine_details’ function (refer to figure 100), if the users enter ‘4’, they will be taken to the ‘vac_status’ function (refer to figure 102), if users enter ‘5’, they will be successfully logged out from their account and if the input selection is not 1-5, 'Invalid Input' will be printed out and the loop will continue until the user enters the correct input (1-5).

```
436 #DONE BY GAN MING HUI
437 #Define a menu for admin(what they can do after they login)
438
439 def admin_menu():
440     while (True):
441         print("\n1.Vaccine Administration\n2.Search Patients Record\n3.Edit the Vaccines\n4.View Statistical Information\n5.Log Out\n")
442         opt = input("Pls enter the selection : ")
443
444         #link to each of the function
445         if (opt == "1"):
446             administration() #record all vaccines administered in vaccination.txt file
447
448         elif (opt == "2"):
449             user_info_admin() #a menu allows admins to choose either searching patients (one / all)
450
451         elif (opt == "3"):
452             vaccine_edit() #allow admins to add new vaccines or delete vaccines
453
454         elif (opt == "4"):
455             statistical_report() #reporting - Statistical Information(VC1/VC2)
456
457         elif (opt == "5"):
458             print("\n*You have been successfully logged out.")
459             break
460
461         else:
462             print("\n*Error : Invalid Input")
```

Figure 80: admin_menu( )

Variables / Functions in admin_menu( ):

Variable / Function	Meaning
opt	A variable that saves the input of selection from the user (1-5)
administration( )	A function that allows admins to record all vaccines administered by patients
user_info_admin( )	A function that prints a menu that allows admins to choose either searching for a specific patient by entering their patient id or searching for all patients at once
vaccine_edit( )	A function that prints a menu that allows the admins to edit either add new vaccines or delete vaccines
statistical_report( )	A function that prints a menu that allows admins to view statistical information for VC1 and VC2.

Explanation for admin_menu( ):

The 'admin_menu' function is a menu that contains five functions. The **main purpose** is to let the admins choose what they want to do when they are successfully logged in. In this function, the admins will be asked to select the ‘1’ option to record all vaccines received at VC1 and VC2 as well as the receiver of that vaccine, select ‘2’ to search for a specific patient's records by entering their patient id or all patients' records, select ‘3’ to edit the currently available vaccines either add new vaccines or deletes vaccines, select ‘4’ to view statistical information for VC1 and VC2. For example, the total number of patients, patients who have received the

vaccine, patients who are waiting for their second dose, and patients who have completed receiving all the doses. Finally, the admins can also select ‘5’ to log out from their accounts. To build this function, at the beginning of line **440** the while loop will be used. Under this loop which is in lines **440-462**, on lines **441**, the users are prompted to enter a choice of whether they want to do vaccine administration, search patients’ records, edit the vaccines, view statistical information, or log out. Next, the selection of users will be read and stored in the ‘opt’ variable on line **442**. In lines **445-462** if...elif...else statement is used, if the users enter '1', they will be taken to the 'administration' function (refer to figure 127), if the users enter '2', they will be taken to the 'user_info_admin' function (refer to figure 104), if the users enter '3', they will be taken to the ‘vaccine_edit’ function (refer to figure 97), if the users enter ‘4’, they will be taken to the ‘statistical_report’ function (refer to figure 95), if users enter ‘5’, they will be successfully logged out from their account and if the input selection is not 1-5, 'Invalid Input' will be printed out and the loop will continue until the users enter the correct input (1-5).

```python
#DONE BY GAN MING HUI
#Define new user registration

def new_user():
        print("\n--------------------Please register your details--------------------")

        #validation for name
        while(True):
            print("\n*Enter '1' to exit")
            name = input("Please enter your name (only letters are accepted) : ")

            #make sure only alphabet entered
            a = name.isalpha()

            if(name == "1"):
                return

            elif(a == True):
                break

            if(a == False):
                if " " in name:
                    break

            else:
                print("\n*Error : Invalid Input")



        #validation for password
        passw = validation_password()

        #exit the registration function
        if (passw == "1"):
            return

        #validation for IC
        f = open("Patients.txt", "r")
        while (True):
            flag = False
            ic = validation_ic()

            #exit the registration
            if(ic == "1"):
                return

            lists = []

            for line in f:
                line_split = line.split(",")
                lists.append(line_split)
            f.close()

            f = open("Patients.txt","r")
            for i in range(len(lists)):
                if (ic) in lists[i][1]:
                    flag = True

            if (flag == True):
                print("\n*Error : IC exists, Please try again")
                continue

            else:
                break


                    #return #to stop asking again the gender
            f.close()


        #validation for gender
        while(True):
            print("\n*Enter '1' to exit")
            gender = input("Please enter your gender(m/f) : ").lower()
            if((gender == "m") or (gender == "f") or (gender == "male") or (gender == "female")):
                    break

            elif(gender == "1"):
                return

            else:
                    print("\n*Error : Invalid Input either male(m) or female(f)")

        #Address
        print("\n*Enter '1' to exit")
        address = input("Please enter your Current Address : ")

        if(address == "1"):
            return

        #validation for postcode
        postcode = validation_postcode()

        #exit registration
        if(postcode == "1"):
            return
```

```
561
562         #validation for vc
563         while(True):
564             print("\n*Enter '1' to exit")
565             vc = input("Please enter the Vaccine Centre where you want to select (vc1/vc2)  : ").lower()
566
567             if(vc == "1"):
568                 return
569
570             if((vc == "vc1") or (vc == "vc2")):
571                 break
572             else:
573                 print("\n*Error : Invalid VC")
574
575         #validation for age
576         while(True):
577
578             #avoid user enter string, string cannot be converted to integer, avoid show red error message
579             try:
580                 print("\n*Enter '1' to exit")
581                 age_int = int(input("Please enter your age (minimum age & maximum age to register is 12 & 100) : "))
582
583                 if(age_int == 1):
584                     return
585
586                 if((age_int > 11) and (age_int < 101)):
587                     age_str = str(age_int)
588                     break
589
590                 else:
591                     print("\n*Error : Invalid age")
592             except:
593                 print("\n*Error : Invalid Input")
594
595
596         #validation for email
597         email = validation_email()
598
599         #exit registration
600         if(email == "1"):
601             return
602
603         #valdidation for phone number
604         phone_num = validation_phone()
605         if(phone_num == "1"):
606             return
607
608         #validation for illness
609         while(True):
610             print("\n*Enter '1' to exit")
611             illness = input("Do you have any illnesses (yes/no) : ").lower()
612
613             if(illness == "1"):
614                 return
615
616             if((illness == "yes") or (illness == "no")):
617                 break
618             else:
619                 print("\n*Error : Invalid Input*")
620
621     #validation for allergic
622     while(True):
623         print("\n*Enter '1' to exit")
624         is_allergic = input("Do you have allergic (yes/no) : ").lower()
625
626         if(is_allergic == "1"):
627             return
628
629         if((is_allergic == "yes") or (is_allergic == "no")):
630             break
631
632         else:
633             print("\n*Error : Invalid Input")
634
635     vaccine = vaccine_selection(age_int)
636     #user want to exit the registration
637     if(vaccine == "1"):
638         return
639
640     pat_id = id(vc)
641     f = open("Patients.txt", "a")
642     f.writelines([pat_id,",",ic,",",name,",", passw,",",gender,",", vc,",", address,",", postcode,",", age_str,",", email,",", phone_num,",", illness,",", is_allergic,",",vaccine,"\n"])
643     f.close()
```

Figure 81: new_user( )

Variables / Functions in new_user( ):

Variable / Function	Meaning
name	A variable that saves the input of the patient's name (1-5)
a	A variable that stores the result (True/False) of checking whether the 'name' variable input is a letter or not.
passw	A variable that stores the return result of the function 'validation_password'
validation_password( )	A function that validates the patient's password

f	A variable that stores the way to open the text file, the way to open the text file in this ‘new_user ‘ function will be read and append mode.
flag	A variable assigned a Boolean value to control the looping and stopping of the loop.
ic	A variable that stores the return result of the function ‘validation_ic’
validation_ic( )	A function that validates the patient’s IC numbers.
lists	An empty list is used to store the records in the text file.
gender	A variable that stores the input of the patient’s gender (m/f).
address	A variable that stores the input of the patient’s address.
postcode	A variable that stores the input of the patient’s postcode.
validation_postcode( )	A function that validates the patient’s postcode.
vc	A variable that stores the input for the selection of vaccine centre (vc1/vc2).
age_int	A variable that stores the input of the patient’s age in integer form.
age_str	A variable that stores the input of the patient’s age in string form.
email	A variable that stores the input of the patient’s email.
validation_email( )	A function that validates the patient’s email.
phone_num	A variable that stores the input of the patient’s phone number.
validation_phone( )	A function that validates the patient’s phone number.
illness	A variable that stores the input (yes/no) of whether the patient has an illness or not

is_allergic	A variable that stores the input (yes/no) of whether the patient is allergic or not
vaccine	A variable that stores the input of vaccine the patient chooses to receive.
vaccine_selection(age_int)	'age_int' is an argument that passes the value back to the parameter of 'vaccine_selection' function. This function lets the user view the options of vaccines and choose the vaccine they want to receive.
pat_id	A variable that stores the input of the patient's patient id.
id(vc)	'vc' is an argument that passes the value back to the parameter of 'id' function. This function will generate a patient id for each of the patients that completes registration.

Explanation for new_user( ):

The **main purpose** of 'new_user' function is to let the patients do their new user registration. In this function, the patients will be asked to enter their name, password, IC, gender, address, postcode, vaccine centre (vc1/vc2), age, email, phone number, illness, allergic, and selected vaccine to complete their registration.

To build this function, at the beginning of line **469** the 'Please register your details' will be printed. The while loop written in line **472** will be used for lines **472-490** until the users enter the correct input. The code in line **473** is to let the users know they can key in '1' if they want to exit the registration. In line **474** the users are prompted to enter their name and the input will be stored in the variable 'name'. In line **477,** to make sure the input of the patient's name is all letters, a function that checks if all inputs are letters will be used and the result will be stored in the variable 'a'. For lines **479-490** if...elif...else statement is used, if the users enter '1', they will exit from the registration, if the users enter a name with all the letters, will be able to continue filling out the next profile, if the users enter an invalid name, 'Invalid Input' will be printed, and the loop will continue until the users enter the correct input. To ensure that the user enters the name in all letters, but with spaces in it, the coding in lines **485-487** is written. This is because the space is not considered a letter. In line **495**, the 'validation_password'

function call is made and the returned value from the 'validation_password' will be stored in the variable 'passw'. In lines **498-499** the if statement is used, if the input of the password is '1', users will exit from the registration.

Lines **502-532**: IC

- Line 502: The 'Patients.txt' will be opened in reading mode.
- Line 503-532: A while loop is used until the correct input is gotten.
- Line 504: The variable 'flag' is used to control the looping and stopping of the while loop.
- Line 505: The 'validation_ic' function call is made and the returned value from the 'validation_ic' will be stored in the variable 'ic'.
- Line 508-509: The if statement is used, if the input of the ic is '1', users will exit from the registration.
- Line 511: A empty list called '**lists'** is used.
- Line 513: A for loop is used to loop every record in the 'Patients' text file.
- Line 514: Every input in every record that looped over in the 'Patients' text file will be split by ',' to make every record become a list.
- Line 515: Appending every list that converted from every record in 'Patients' text file into the empty list created in '511', so the empty list will become a nested list.
- Line 516: The 'Patients' text file will be closed.
- Line 518: The 'Patients' text file will be opened in reading mode.
- Line 519: A for loop is used to loop through lines 520-521, the number of loops is based on the range of lengths of 'list'.
- Line 520-521: Under the for loop in line **519**, the if statement is used, if the input of ic already exists in one of the lists in the nested list, the value of the variable 'flag' will change from 'False' to 'True' else will be the same.
- Line 523-528: The if...else statement is used. If the 'flag' is equal to 'True' it means the ic is already used by others. 'IC exists' will be printed. 'continue' keyword will be used to end the current iteration in the while loop and continues to the next iteration. Else, if this ic has not been used yet, then the while loop will be ended by the keyword 'break'. Users will be able to continue filling in their gender.
- Line **532**: The 'Patients' text file will be closed.

Lines 536-546: Gender

- Line 536-546: A while loop is used until the correct input is gotten.
- The code in line **537** is to let the users know they can key in '1' if they want to exit the registration.
- Line **538**: The users are prompted to enter their gender and the input will be stored in the variable 'gender'.
- Lines 539-546: if...elif...else statement is used. If the users enter 'm', 'f', 'male' or 'female', then the while loop will be ended by the keyword 'break', and the users can continue to fill in their address. If the users enter '1', they will exit from the registration. If the user enters an input other than the input above, 'Invalid Input either male(m) or female(f)' will be printed out and continue to loop.

Lines 549-553: Address

- Line 549: To let the users know they can key in '1' if they want to exit the registration.
- Line 550: The users are prompted to enter their address and the input will be stored in the variable 'address'.
- Lines 552-553: if statement is used. If the users enter '1', they will exit from the registration.

Lines 556-560: Postcode

- Line 556: The 'validation_postcode' function call is made and the returned value from the 'validation_postcode' will be stored in the variable 'postcode'.
- Lines 559-560: if statement is used. If the users enter '1', they will exit from the registration.

Lines 563-573: Vaccine Centre

- Line 563-573: A while loop is used until the correct input is gotten.
- Line 564: To let the users know they can key in '1' if they want to exit the registration.
- Line 565: Users will be prompted to choose the vaccination centre where they want to receive their selected vaccine and the input will be converted to uppercase letters which will be stored in the variable 'vc'.
- Lines 567-573: if...else statement is used. If the users enter '1', they will exit from the registration. If the variable 'vc' is equal to 'VC1' or 'VC2', then the while loop will be ended by the keyword 'break', and the users can continue to fill in their age. If the user

enters an input other than the input '1', 'VC1' or 'VC2', then the 'Invalid VC' will be printed and continue to loop.

Lines 576-593: Age

- Line 576-593: A while loop is used until the correct input is gotten.
- Line 579-593: Try and except is used. This is because if users enter a string, the string cannot be converted to an integer. As a result, the error message will be shown. To prevent that, an 'Invalid Input' message can be written in the except part. Once there is any error that occurs in the try part, the program will directly jump from try to except part and execute the coding below the except part.
- Line 580: To let the users know they can key in '1' if they want to exit the registration.
- Line 581: Users will be prompted to enter their age and the input will be converted to an integer which will be stored in the variable 'age_int'.
- Line 583-591: if...else statement is used. If the users enter '1', they will exit from the registration. If the variable 'age_int' is greater than 11 and smaller than 10, then the input of 'age_int' will be converted to a string and stored in the variable 'age_str'. Also, the while loop will be ended by the keyword 'break' and the users will able to continue to fill in their email. However, if the user enters an input other than the input '1', not greater than 11 and not smaller than 10, then the 'Invalid Input' will be printed and continue to loop.

Lines 597-601: Email

- Line 597: The 'validation_email' function call is made and the returned value from the 'validation_email' will be stored in the variable 'email'.
- Lines 600-601: if statement is used. If the users enter '1', they will exit from the registration.

Lines 609-619: Illness

- Line 609-619: A while loop is used until the correct input is gotten.
- The code in line **610** is to let the users know if they want to exit the registration they can key in '1'.
- In line **611** the users are prompted to enter whether they have an illness, and the input will be stored in the variable 'illness'.

- Lines 613-619: if...else statement is used. If the users enter '1', they will exit from the registration. If the variable 'illness' is equal to 'yes' or 'no', then the while loop will be ended by the keyword 'break', and the users will continue to fill in whether there is an allergic. If the user enters an input other than the input '1', 'yes' or 'no', then the 'Invalid Input' will be printed and continue to loop.

Lines 622-633: Allergic

- Line 622-633: A while loop is used until the correct input is gotten.
- The code in line **623** is to let the users know if they want to exit the registration they can key in '1'.
- In line **624** the users are prompted to enter whether they have an allergy, and the input will be stored in the variable 'allergic'.
- Lines 626-633: if...else statement is used. If the users enter '1', they will exit from the registration. If the variable 'allergic' is equal to 'yes' or 'no', then the while loop will be ended by the keyword 'break', and the users will continue to choose the vaccines they desire. If the user enters an input other than the input '1', 'yes' or 'no', then the 'Invalid Input' will be printed and continue to loop.

Lines 635-638: Vaccine selection

- Line 635: The 'vaccine_selection' function call with the 'age_int' argument is made and the returned value from the 'vaccine_selection' function will be stored in the variable 'vaccine'.
- Lines 637-638: if statement is used. If the users enter '1', they will exit from the registration.

Lines 640: Patient id

- Line 640: The 'id' function call with the 'vc' argument is made and the returned value from the 'id' function will be stored in the variable 'pat_id'.

Lines 641-643 Patients.txt

- Line 641: The 'Patients.txt' will be opened in append mode.
- Line 642: Patient id, IC, name, password, gender, vaccine centre, address, postcode, age in string form, email, phone number, is there any disease and allergic as well as the selected vaccine of the patient will be stored into 'Patients.txt'.

- Line 643: The ‘Patients.txt’ will be closed.

```
649 def validation_password():
650         regex_password = "[0-9a-zA-Z]{6,}"
651 
652         while (True):
653             a = False
654             b = False
655             print("\n*Enter '1' to exit")
656             #Wiil only store the correct password in text file
657             passw = input("Please enter your password (at least 6 characters / must contain only alpha and numbers) : ")
658 
659             if(passw == "1"):
660                 return passw
661 
662             if(re.search(regex_password,passw)):
663 
664                 #make sure contain alpha and numbers
665                 makesure_digit = any(char.isdigit() for char in passw)
666                 makesure_alpha = any(char.isalpha() for char in passw)
667 
668                 if(makesure_digit == True):
669                     a = True
670 
671                 else:
672                     print("\n*Error : Must contain digit")
673 
674                 if(makesure_alpha == True):
675                     b = True
676 
677                 else:
678                     print("\n*Error : Must contain alpha")
679 
680 
681                 if((a == True) and (b == True)):
682                     return passw
683 
684             else:
685                 print("\n*Error : minimum 6 characters")
```

Figure 82: validation_password( )

Variables / Functions in validation_password( ):

Variable / Function	Meaning
regex_password	A regular expression that determines the pattern of the password.
a	To make sure any digit is included. If digit is included, then a = True, else a = False.
b	To make sure any alpha is included. If alpha is included, then b = True, else b = False.
passw	A variable that stores the input of the patient's password.
makesure_digit	A variable that stores the result (True/False) of checking whether any of the digits is included in the password.
makesure_alpha	A variable that stores the result (True/False) of checking whether any of the alphas is included in the password.

Explanation for validation_password( ):

The **main purpose** of 'validation_password' function is to validate the password entered by the users. In this function, the patients will be asked to enter their password. The password entered must only contain numbers and letters and a minimum of six characters.

Line 650: Regular expression for password

- A regular expression named ‘regex_password’ is used to determine the pattern of the password.

Line 652-685: Password

- Line 652-685: A while loop is used until the correct input is gotten.
- Line 653: Set the variable ‘a’ equal to False.
- Line 654: Set the variable ‘b’ equal to False.
- Line 655: To let the users know they can key in ‘1’ if they want to exit the registration.
- Line 657: Users will be prompted to enter their password and the input will be stored in the variable 'passw'.
- Lines 659-685: if...else statement is used. If the users enter ‘1’, they will exit from the registration. If the password matches the pattern, then the while loop will be ended and the input will be returned to the ‘validation_password’ function. If the users do not match the password pattern or the password does not contain any numbers or the password does not contain any letters, then the error message will be printed and continue to loop.

```
688 #DONE BY GAN MING HUI
689 #Validation for IC
690 def validation_ic():
691         #Year-Month-Day
692         regex_ic = '(([0-9]{2})(0[1-9]|1[0-2])(0[1-9]|1[0-9]|2[0-9]|3[0-1]))-([0-9]{2})-([0-9]{4})'
693 
694 
695         while(True):
696             print("\n*Enter '1' to exit")
697             ic = input("Please enter your identity card (YYMMDD-99-9999) : ")
698 
699             if(ic == "1"):
700                 return ic
701 
702             if (len(ic) == 14):
703                     if(re.search(regex_ic,ic)):
704                             return ic
705                     else:
706                             print("\n*Error : Invalid IC")
707 
708             else:
709                     print("\n*Error : Invalid IC")
```

Figure 83: validation_ic( )

Variables / Functions in validation_ic( ):

Variable / Function	Meaning
regex_ic	A regular expression that determines the pattern of the IC.
ic	A variable that stores the input of the patient's IC.

Explanation for validation_ic( ):

The **main purpose** of 'validation_ic' function is to validate the IC entered by the users. In this function, the patients will be asked to enter their IC. The IC entered must only contain numbers and only 14 characters. The first two digits of IC can be any number from 0-9. The third digit of the IC can only be 0 or 1. The fourth digit of the IC needs to be based on the third digit, if the third digit is 0 then the fourth digit must be one of the digits 1-9 but if the third digit is 1 then the fourth digit must be 0,1 or 2. The fifth digit of IC can be one of 0-3. The sixth digit of IC is based on the fifth digit. If the fifth digit is 0 then the sixth digit can only be one of the digits 1-9. If the fifth digit is 1 then the sixth digit can only be one of the digits 0-9. If the fifth digit is 2 then the sixth digit can only be one of the digits 0-9. If the fifth number is 3, then the sixth number can only be 0 or 1. After entering the six numbers, the user must enter - first and then enter the seventh and eighth numbers. The seventh and eighth numbers can be one of the numbers 0-9. After entering the eighth number, the user must first enter - and then enter the remaining four numbers. The remaining four numbers can be one of the numbers from 0-9.

Line 692: Regular expression for IC

- A regular expression named 'regex_ic' is used to determine the pattern of the IC.

Line 695-709: IC

- Line 695-709: A while loop is used until the correct input is gotten.
- Line 696: To let the users know they can key in ‘1’ if they want to exit the registration.
- Line 697: Users will be prompted to enter their IC and the input will be stored in the variable 'ic'.
- Lines 699-709: if...else statement is used. If the users enter ‘1’, they will exit from the registration. If the IC matches the pattern, then the while loop will be ended and the input will be returned to the ‘validation_ic’ function. If the users do not match the password pattern, then the error message will be printed and continue to loop.

```
712 #DONE BY GAN MING HUI
713 #Validation for postcode
714
715 def validation_postcode():
716     while(True):
717         print("\n*Enter '1' to exit")
718         postcode = input("Please enter your postcode (only 5 numbers can be entered) : ")
719         check = postcode.isnumeric()
720
721         if(postcode == "1"):
722             return postcode
723
724         if(check == True) and (len(postcode) == 5 ):
725             return postcode
726
727         else:
728             print("\n*Error : Invalid Postcode")
```

Figure 84: validation_postcode( )

Variables / Functions in validation_postcode( ):

Variable / Function	Meaning
postcode	A variable that stores the input of the patient's postcode.
check	A variable that stores the result (True/False) of checking whether the 'postcode' variable input is all numeric or not.

Explanation for validation_postcode( ):

The **main purpose** of the 'validation_postcode' function is to validate the postcode entered by the users. In this function, the patients will be asked to enter their postcodes. The postcode must be only five digits.

Line 716-728: Postcode

- Line 716-728: A while loop is used until the correct input is gotten.
- Line 717: To let the users know they can key in '1' if they want to exit the registration.
- Line 718: Users will be prompted to enter their postcode and the input will be stored in the variable 'postcode'.
- Line 719: The postcode entered by the user will be checked to make sure it is all numeric.
- Lines 721-728: if...else statement is used. If the users enter '1', they will exit from the registration. If the IC only includes five numbers, then the while loop will be ended and the input will be returned to the 'validation_postcode' function. If the users do not match the postcode pattern, then the error message will be printed and continue to loop.

```
731 #DONE BY GAN MING HUI
732 #Validation for email
733
734 def validation_email():   #symbol + = 1 or more  (any) characters
735     regex_email = re.compile(r'([a-z0-9]+[._])*[a-z0-9]+@[a-z0-9-]+(\.[a-z]{2,})+')
736
737     while(True):
738         print("\n*Enter '1' to exit")
739         email = input("Please enter your Email : ")
740
741         if(email == "1"):
742             return email
743
744         if(re.search(regex_email,email)):
745             return email
746
747         else:
748             print("\n*Error : Invalid email")
```

Figure 85: validation_email( )

Variables / Functions in validation_email( ):

Variable / Function	Meaning
regex_email	A regular expression that determines the pattern of the email.
email	A variable that stores the input of the patient's email.

Explanation for validation_email( ):

The **main purpose** of the 'validation_email' function is to validate the email entered by the users. In this function, the patients will be asked to enter their email. We made sure that there is at least one alphanumeric character before and after each special character because they cannot immediately precede the @ symbol or begin the prefix.An email can have multiple top-level domains separated by a dot in terms of the domain.

Line 735: Regular expression for email

- A regular expression named 'regex_email' is used to determine the pattern of the email.

Line 737-748: Email

- Line 737-748: A while loop is used until the correct input is gotten.
- Line 738: To let the users know they can key in '1' if they want to exit the registration.
- Line 739: Users will be prompted to enter their email and the input will be stored in the variable 'email'.
- Lines 741-748: if...else statement is used. If the users enter '1', they will exit from the registration. If the email matches with the pattern, then the while loop will be ended and the input will be returned to the 'validation_email' function. If the users do not match the email pattern, then the error message will be printed and continue to loop.

```
751 #DONE BY GAN MING HUI
752 #Validation for phone number
753
754 def validation_phone():
755     regex_phone = '^(\+?6?01)[02-46-9][-][0-9]{7}$|^(\+?6?01)[1][-][0-9]{8}$'
756
757     while(True):
758         print("\n*Enter '1' to exit")
759         phone_num = input("Format : 01X-XXXXXXX or 011-XXXXXXXX,where X is 0, 2, 3, 4, 6, 7, 8, 9\nPlease enter your phone number in above format : ")
760
761         if(phone_num == "1"):
762             return phone_num
763
764         if(re.search(regex_phone,phone_num)):
765             return phone_num
766
767         else:
768             print("\n*Error : Invalid Phone Number, Please refer to the format")
```

Figure 86: validation_phone( )

Variables / Functions in validation_phone( ):

Variable / Function	Meaning
regex_phone	A regular expression that determines the pattern of the phone number.
phone_num	A variable that stores the input of the patient's phone number.

Explanation for validation_phone( ):

The **main purpose** of the 'validation_phone' function is to validate the phone number entered by the users. In this function, the patients will be asked to enter their phone numbers. The first two digits of the phone number pattern must be 01. The third digit can be 0-9 except for 5. This is because 015 is not a valid phone number. After entering the three digits, the user must enter '-' before the remaining digits can be entered. If the third digit of the phone number is 1 then there must be 8 more digits to enter but if the third digit of the phone number is other than 1 and 5 then there must be 7 more digits to enter.

Line 755: Regular expression for phone number

- A regular expression named 'regex_phone' is used to determine the pattern of the phone number.

Line 757-768: Phone number

- Line 757-768: A while loop is used until the correct input is gotten.
- Line 758: To let the users know they can key in '1' if they want to exit the registration.
- Line 759: Users will be prompted to enter their phone number and the input will be stored in the variable 'phone_num'.
- Lines 761-768: if...else statement is used. If the users enter '1', they will exit from the registration. If the phone number matches the pattern, then the while loop will be ended

and the input will be returned to the ‘validation_phone’ function. If the users do not match the phone number pattern, then the error message will be printed and continue to loop.

```python
#DONE BY GAN MING HUI
#Functions for user to edit profile

def edit_profile_menu(patientic):
    while (True):
        print("\n--------------------Edit Profile--------------------")
        print("\n1.Password\n2.Address & Postcode\n3.Email\n4.Phone Number\n5.Vaccine\n6.Back\n")

        opt = input("Please Enter the selection : ")

        if (opt == "1"):
            edit_password(patientic)

        elif (opt == "2"):
            edit_address_postcode(patientic)

        elif (opt == "3"):
            edit_email(patientic)

        elif (opt == "4"):
            edit_phone_number(patientic)

        elif (opt == "5"):
            edit_vaccine(patientic)

        elif (opt == "6"):
            break

        else:
            print("Invalid Input")
```

Figure 87: edit_profile_menu(patientic)

Variables / Functions in edit_profile_menu(patientic):

Variable / Function	Meaning
opt	A variable that saves the input of selection from the user (1-6)
edit_password(patientic)	‘patientic’ is an argument that passes the value back to the parameter of ‘edit_password’ function. This function allows users to edit their old password to a new password.
edit_address_postcode(patientic)	‘patientic’ is an argument that passes the value back to the parameter of ‘edit_address_postcode’ function. This function allows users to edit their old address and postcode to a new address and postcode.
edit_email(patientic)	‘patientic’ is an argument that passes the value back to the parameter of ‘edit_email’ function. This function allows users to edit their old email to a new email.
edit_phone_number(patientic)	‘patientic’ is an argument that passes the value back to the parameter of ‘edit_phone_number’ function. This

	function allows users to edit their old phone number to a new phone number.
edit_vaccine(patientic)	‘patientic’ is an argument that passes the value back to the parameter of ‘edit_vaccine’ function. This function allows users to edit their selected vaccine to another vaccine.

Explanation for edit_profile_menu(patientic):

The **main purpose** of this function is to present a menu for the user, also known as the patient, to select the information they want to modify. In this function, the users will be asked to select the ‘1’ option to edit the password, select ‘2’ to edit the address and postcode, select ‘3’ to edit the email, select ‘4’ to edit the phone number, select ‘5’ to edit the vaccine to another vaccine they want to receive or select ‘6’ to back to the previous page.

Line 776-801: Edit profile

- Line 776-801: A while loop is used until the correct input is gotten.
- Line 777: Title of ‘edit profile’ will be printed.
- Line 778: Users are prompted to enter a choice whether they want to edit password, address & postcode, email, phone number, vaccine or back to the previous page.
- Line 780: The selection of users will be read and stored in the ‘opt’ variable.

Line 782-801: if...elif…else statement is used. If the users enter ‘1’, they will be taken to the 'edit_password' function and the value of the ‘patientic’ argument will also be passed back to the parameter of ‘edit_password’ function. If the users enter ‘2’, they will be taken to the 'edit_address_postcode' function, and the value of the ‘patientic’ argument will also be passed back to the parameter of ‘edit_address_postcode’ function. If the users enter ‘3’, they will be taken to the 'edit_email' function, and the value of the ‘patientic’ argument will also be passed back to the parameter of ‘edit_email’ function. If the users enter ‘4’, they will be taken to the 'edit_phone_number' function, and the value of the ‘patientic’ argument will also be passed back to the parameter of ‘edit_phone_number’ function. If the users enter ‘5’, they will be taken to the 'edit_vaccine' function, and the value of the ‘patientic’ argument will also be passed back to the parameter of ‘edit_vaccine’ function. If users enter ‘6’, they will be successfully

back to the previous page. If the input selection is not 1-6, 'Invalid Input' will be printed out and the loop will continue until the users enter the correct input (1-6).

```
#DONE BY GAN MING HUI
#Edit phone number

def edit_phone_number(patientic):

    line_number = 0
    access = False

    f = open("Patients.txt","r")
    lists = []
    for line in f:
        line_split = line.split(",")
        lists.append(line_split)

    #To know the users info in which line
    while (access == False):
        if(lists[line_number][1] == patientic):
            patientid, ic, name, password, gender, vc, address, postcode, age, email, old_phone_number, illness, allergic, vaccine = lists[line_number]#get the old record
            access = True
        else:
            line_number += 1
    f.close()

    print("\n--------------------Edit Phone Number--------------------\n")
    new_phone_number = input("Please enter new phone number : ")
    makesure = input("Do you really want to edit (yes/no) ?\n> ").lower()
    while (True):
        if (makesure == "yes"):
            print("\n--------------------Edited Successfully--------------------")
            print("\n* Phone Number has been edited from", old_phone_number,"to",new_phone_number,"*")

            #To update the ONE new record to the file
            f = open("Patients.txt","a")
            #insert new record but haven deleted old record
            f.writelines([patientid,",",ic,",",name,",", password,",",gender,",", vc,",", address,",", postcode,",", age,",", email,",", new_phone_number,",", illness,",", allergic,",", vaccine])
            f.close()

            #delete old record and write new record into text file
            f = open("Patients.txt","r")
            lines = f.readlines()
            del lines[line_number]  #the lines contains the latest records(including the record which has been edited)
            f.close()

            f = open("Patients.txt","w")
            for line in lines:  #write all the latest record into text file
                f.write(line)
            f.close()
            break

        elif (makesure == "no"):
            print("*Nothing has changed*")
            break

        else:
            print("Invalid Input")
```

Figure 88: edit_phone_number(patientic)

Variables / Functions in edit_phone_number(patientic):

Variable / Function	Meaning
patientic	A parameter that receives the value from the argument in ‘edit_phone_number’ function call.
line_number	Used to know which row the desired user's profile is on.
access	To control the while loop looping or stopping.
f	A variable that stores the way to open the patients text file, the way to open the text file in this ‘edit_phone_number’ function will be read, append and write mode.
lists	An empty list is used to store the records in the text file.
line	A variable that reads every looped record from the ‘Patients’ text file, but only one record can be read, and every time a new record is read the old one will be removed. This variable is used for appending purposes.
line_split	A variable to store the result after splitting the record in line by ‘,’. This variable is used for data positioning purposes.
patientid	A variable to store the patient ID of the desired patient.

ic	A variable to store the IC of the desired patient.
name	A variable to store the name of the desired patient.
password	A variable to store the password of the desired patient.
gender	A variable to store the gender of the desired patient.
vc	A variable to store the vaccine centre of the desired patient.
address	A variable to store the address of the desired patient.
postcode	A variable to store the postcode of the desired patient.
age	A variable to store the age of the desired patient.
email	A variable to store the email of the desired patient.
old_phone_number	A variable to store the old phone number of the desired patient.
illness	A variable to store whether the desired patient has an illness or not.
allergic	A variable to store whether the desired patient is allergic or not.
vaccine	A variable to store the selected vaccine of the desired patient.
new_phone_number	A variable to store the new phone number of desired patient.
makesure	A variable to store the input (yes/no) of whether the patient really wants to edit or not.
lines	A variable to store all the record lines from patients text file.

Explanation for edit_phone_number(patientic):

The **main purpose** of this function is to let users edit their phone numbers. In this function, the users will be asked to enter their new phone number. After entering a new phone number, users will be asked whether the patients really want to edit or not. If the users enter ‘yes’, the old

phone number will be replaced with a new one. If the users enter ‘no’, no changes will be made to the phone number.

Line 809: The variable ‘line_number’ is set to 0.

Line 810: The variable ‘access’ is set to False.

Line 812: Open the ‘Patients.txt’ in read mode.

Line 813: Create an empty list named ‘lists’.

Lines 814-816: Using for loop to insert all the records in ‘Patients.txt’ to a list.

- Line 814: Using for loop to loop every record in the ‘Patients.txt’ text file.
- Line 815: Splitting the record in line in the ‘Patients.txt’ by ‘,’.
- Line 816: Appending the record into the ‘lists’ list. Therefore, the ‘lists’ will become a nested list.

Lines 819-824: Using the while loop to know the info of the desired users in which line.

- Line 819: A while loop is used until the info of the desired is gotten.
- Line 820-824: if…else statement is used. If the patient IC matches the patient IC in the ‘lists’ nested list, then the record line will be retrieved. Furthermore, the value for the access will also change to ‘True’. If the patient IC is not on the ‘lists’ nested list, the value for variable ‘line_number’ will increase by 1 to test the next list in the nested list.

Line 825: The ‘Patients.txt’ is closed.

Line 827: Title of ‘Edit Phone Number’ is printed.

Line 828: Users will be prompted to enter their new phone number and the input will be stored in the variable 'new_phone_number’.

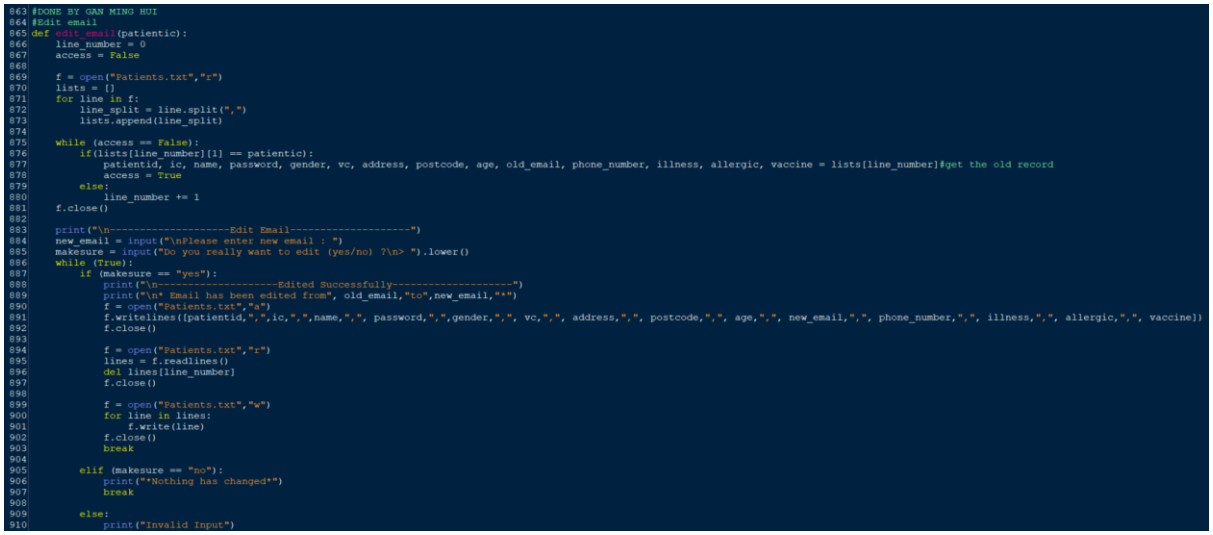
Line 829: Users will be prompted to enter do they really want to edit the phone number and the input will be converted to lowercase before being stored in the variable ‘makesure’.

Lines 830-858: Using the while loop to avoid re-enter the new phone number after answering an invalid entry

- Line 830: A while loop is used until the users choose ‘yes’ or ‘no’.
- Line 831-858: if…elif…else statement is used. If the users enter ‘yes’, then they will be able to successfully change the phone number. If the users enter ‘no’, then nothing

will be changed. If the users enter other than ‘yes’ or ‘no’, the ‘Invalid Input’ will be printed.

- Line 836-839: The ‘Patients.txt’ will be opened in append mode. The record with the new email will be inserted into ‘Patiens.txt’. The ‘Patients.txt’ will be closed.
- Line 842-845: The ‘Patients.txt’ will be opened in reading mode. All records in the text file will be read into the ‘lines’ variable. After that, the record with the old email will be deleted. The ‘Patients.txt’ will be closed.
- Line 847-850: The ‘Patients.txt’ will be opened in writing mode. The latest record will be rewritten into the ‘Patients’ text file. The ‘Patients.txt’ will be closed.



```
#DONE BY GAN MING HUI
#Edit email
def edit_email(patientic):
    line_number = 0
    access = False

    f = open("Patients.txt","r")
    lists = []
    for line in f:
        line_split = line.split(",")
        lists.append(line_split)

    while (access == False):
        if(lists[line_number][1] == patientic):
            patientid, ic, name, password, gender, vc, address, postcode, age, old_email, phone_number, illness, allergic, vaccine = lists[line_number]#get the old record
            access = True
        else:
            line_number += 1
    f.close()

    print("\n--------------------Edit Email--------------------")
    new_email = input("\nPlease enter new email : ")
    makesure = input("Do you really want to edit (yes/no) ?\n> ").lower()
    while (True):
        if (makesure == "yes"):
            print("\n--------------------Edited Successfully--------------------")
            print("\n* Email has been edited from", old_email,"to",new_email,"*")
            f = open("Patients.txt","a")
            f.writelines([patientid,",",ic,",",name,",", password,",",gender,",", vc,",", address,",", postcode,",", age,",", new_email,",", phone_number,",", illness,",", allergic,",", vaccine])
            f.close()

            f = open("Patients.txt","r")
            lines = f.readlines()
            del lines[line_number]
            f.close()

            f = open("Patients.txt","w")
            for line in lines:
                f.write(line)
            f.close()
            break

        elif (makesure == "no"):
            print("*Nothing has changed*")
            break

        else:
            print("Invalid Input")
```

Figure 89: edit_email(patientic)

Variables / Functions in edit_email(patientic):

Variable / Function	Meaning
patientic	A parameter that receives the value from the argument in ‘edit_email’ function call.
line_number	Used to know which row the desired user's profile is on.
access	To control the while loop looping or stopping.
f	A variable that stores the way to open the patients text file, the way to open the text file in this ‘edit_phone_number’ function will be read, append and write mode.
lists	An empty list is used to store the record lines in the text file.
line	A variable that reads every looped record line from the ‘Patients’ text file, but only one record can be read at a time, and every time a new record is read the old one will be removed. This variable is used for appending purposes.
line_split	A variable to store the result after splitting the record in line by ‘,’. This variable is used for data positioning purposes.
patientid	A variable to store the patient ID of the desired patient.
ic	A variable to store the IC of the desired patient.

name	A variable to store the name of the desired patient.
password	A variable to store the password of the desired patient.
gender	A variable to store the gender of the desired patient.
vc	A variable to store the vaccine centre of the desired patient.
address	A variable to store the address of the desired patient.
postcode	A variable to store the postcode of the desired patient.
age	A variable to store the age of the desired patient.
old_email	A variable to store the old email of the desired patient.
phone_number	A variable to store the phone number of the desired patient.
illness	A variable to store whether the desired patient has an illness or not.
allergic	A variable to store whether the desired patient is allergic or not.
vaccine	A variable to store the selected vaccine of the desired patient.
new_email	A variable to store the new email of the desired patient.
makesure	A variable to store the input (yes/no) of whether the patient really wants to edit or not.
lines	A variable to store all the record lines from patients text file.

Explanation for edit_email(patientic):

The **main purpose** of this function is to let users edit their email. In this function, the users will be asked to enter their new email. After entering a new email, users will be asked whether the patients really want to edit or not. If the users enter ‘yes’, the old email will be replaced with a new one. If the users enter ‘no’, no changes will be made to the email.

Line 866: The variable ‘line_number’ is set to 0.

Line 867: The variable ‘access’ is set to False.

Line 869: Open the ‘Patients.txt’ in read mode.

Line 870: Create an empty list named ‘lists’.

Line 871-873: Using for loop to convert the records in text file into a list and add the list into another list

- Line 871: Using for loop to loop every record in the ‘Patients’ text file.
- Line 872: Splitting the record in line in the ‘Patients.txt’ by ‘,’.
- Line 873: Appending the record into the ‘lists’ list.

Lines 875-880: Using the while loop to know the info of the desired users and in which line.

- Line 875: A while loop is used until the info of the desired patient is gotten.
- Line 876-880: if…else statement is used. If the patient IC matches the patient IC in the ‘lists’ nested list, then the record line will be retrieved. Furthermore, the value for the access will also change to ‘True’. If the patient IC is not on the ‘lists’ nested list, the value for variable ‘line_number’ will increase by 1 to test the next list in the nested list.

Line 881: The ‘Patients.txt’ is closed.

Line 883: Title of ‘Edit Email’ is printed.

Line 884: Users will be prompted to enter their new email and the input will be stored in the variable ‘new_email’.

Line 885: Users will be prompted to enter do they really want to edit the email and the input will be converted to lowercase before being stored in the variable ‘makesure’.

Lines 886-910: Using the while loop to avoid re-enter the new email after answering an invalid entry

- Line 886: A while loop is used until the users choose ‘yes’ or ‘no’.
- Line 887-910: if…elif…else statement is used. If the users enter ‘yes’, then they will be able to successfully change the email. If the users enter ‘no’, then nothing will be changed. If the users enter other than ‘yes’ or ‘no’, the ‘Invalid Input’ will be printed.
- Line 890-892: The ‘Patients.txt’ will be opened in append mode. The record with the new email will be inserted into ‘Patiens.txt’. The ‘Patients.txt’ will be closed.

- Line 894-897: The ‘Patients.txt’ will be opened in reading mode. All records in the text file will be read into the ‘lines’ variable. After that, the record with the old email will be deleted. The ‘Patients.txt’ will be closed.
- Line 899-902: The ‘Patients.txt’ will be opened in writing mode. The latest record will be rewritten into the ‘Patients’ text file. The ‘Patients.txt’ will be closed.

```
#DONE BY GAN MING HUI
#Edit address_postcode
def edit_address_postcode(patientic):
    line_number = 0
    access = False

    f = open("Patients.txt","r")
    lists = []
    for line in f:
        line_split = line.split(",")
        lists.append(line_split)

    while (access == False):
        if(lists[line_number][1] == patientic):
            patientid, ic, name, password, gender, vc, old_address, old_postcode, age, email, phone_number, illness, allergic, vaccine = lists[line_number]#get the old record
            access = True
        else:
            line_number += 1
    f.close()

    print("\n--------------------Edit Address & Postcode--------------------")
    new_address = input("\nPlease enter new address : ")
    new_postcode = input("Please enter new postcode : ")
    makesure = input("Do you really want to edit (yes/no) ?\n> ").lower()
    while (True):
        if (makesure == "yes"):
            print("\n--------------------Edited Successfully--------------------")
            print("\n* Address and Postcode has been edited from", old_address,"&",old_postcode,"to",new_address,"&",new_postcode,"*")
            f = open("Patients.txt","a")
            f.writelines([patientid,",",ic,",",name,",", password,",",gender,",", vc,",", new_address,",", new_postcode,",", age,",", email,",", phone_number,",", illness,",", allergic,",", vaccine])
            f.close()

            f = open("Patients.txt","r")
            lines = f.readlines()
            del lines[line_number]
            f.close()

            f = open("Patients.txt","w")
            for line in lines:
                f.write(line)
            f.close()
            break

        elif (makesure == "no"):
            print("*Nothing has changed*")
            break

        else:
            print("Invalid Input")
```

Figure 90: edit_address_postcode(patientic)

Variables / Functions in edit_address_postcode(patientic):

Variable / Function	Meaning
patientic	A parameter that receives the value from the argument in ‘edit_address_postcode’ function call.
line_number	Used to know which row the desired user's profile is on.
access	To control the while loop looping or stopping.
f	A variable that stores the way to open the patients text file, the way to open the text file in this ‘edit_phone_number’ function will be read, append and write mode.
lists	An empty list is used to store the record lines in the text file.
line	A variable that reads every looped record line from the ‘Patients’ text file, but only one record can be read at a time, and every time a new record is read the old one will be removed. This variable is used for appending purposes.
line_split	A variable to store the result after splitting the record in line by ‘,’. This variable is used for data positioning purposes.
patientid	A variable to store the patient ID of the desired patient.
ic	A variable to store the IC of the desired patient.

name	A variable to store the name of the desired patient.
password	A variable to store the password of the desired patient.
gender	A variable to store the gender of the desired patient.
vc	A variable to store the vaccine centre of the desired patient.
old_address	A variable to store the old address of the desired patient.
old_postcode	A variable to store the old postcode of the desired patient.
age	A variable to store the age of the desired patient.
email	A variable to store the email of the desired patient.
phone_number	A variable to store the phone number of the desired patient.
illness	A variable to store whether the desired patient has an illness or not.
allergic	A variable to store whether the desired patient is allergic or not.
vaccine	A variable to store the selected vaccine of the desired patient.
new_address	A variable to store the new address of the desired patient.
new_postcode	A variable to store the new postcode of the desired patient.
makesure	A variable to store the input (yes/no) of whether the patient really wants to edit or not.
lines	A variable to store all the record lines from patients text file.

Explanation for edit_address_postcode(patientic):

The **main purpose** of this function is to let users edit their address and postcode. In this function, the users will be asked to enter their new address and postcode. After entering a new address and postcode, users will be asked whether the patients really want to edit or not. If the users enter ‘yes’, the old address and postcode will be replaced with a new one. If the users enter ‘no’, no changes will be made to the address and postcode.

Line 918: The variable ‘line_number’ is set to 0.

Line 919: The variable ‘access’ is set to False.

Line 921: Open the ‘Patients.txt’ in read mode.

Line 922: Create an empty list named ‘lists’.

Lines 923-925: Using for loop to insert all the records in ‘Patients.txt’ to a list.

- Line 923: Using for loop to loop every record in the ‘Patients.txt’ text file.
- Line 924: Splitting the record in line in the ‘Patients.txt’ by ‘,’.
- Line 925: Appending the record into the ‘lists’ list. Therefore, the ‘lists’ will become a nested list.

Lines 927-932: Using the while loop to know the info of the desired users and in which line.

- Line 927: A while loop is used until the info of the desired is gotten.
- Line 928-932: if…else statement is used. If the patient IC matches the patient IC in the ‘lists’ nested list, then the record line will be retrieved. Furthermore, the value for the access will also change to ‘True’. If the patient IC is not on the ‘lists’ nested list, the value for variable ‘line_number’ will increase by 1 to test the next list in the nested list.

Line 933: The ‘Patients.txt’ is closed.

Line 935: Title of ‘Edit Address & Postcode’ is printed.

Line 936: Users will be prompted to enter their new address and the input will be stored in the variable ‘new_address’.

Line 937: Users will be prompted to enter their new postcode and the input will be stored in the variable ‘new_postcode’.

Line 938: Users will be prompted to enter do they really want to edit the address and postcode and the input will be converted to lowercase before being stored in the variable ‘makesure’.

Lines 939-963: Using the while loop to avoid re-enter the new address and postcode after answering an invalid entry

- Line 939: A while loop is used until the users choose ‘yes’ or ‘no’.
- Line 940-963: if…elif…else statement is used. If the users enter ‘yes’, then they will be able to successfully change the phone number. If the users enter ‘no’, then nothing

will be changed. If the users enter other than ‘yes’ or ‘no’, the ‘Invalid Input’ will be printed.

- Line 943-945: The ‘Patients.txt’ will be opened in append mode. The record with the new email will be inserted into ‘Patiens.txt’. The ‘Patients.txt’ will be closed.
- Line 947-950: The ‘Patients.txt’ will be opened in reading mode. All records in the text file will be read into the ‘lines’ variable. After that, the record with the old email will be deleted. The ‘Patients.txt’ will be closed.
- Line 952-955: The ‘Patients.txt’ will be opened in writing mode. The latest record will be rewritten into the ‘Patients’ text file. The ‘Patients.txt’ will be closed.

```python
#DONE BY GAN MING HUI
#Edit password
def edit_password(patientic):

    line_number = 0
    access = False

    f = open("Patients.txt","r")
    lists = []
    for line in f:
        line_split = line.split(",")
        lists.append(line_split)

    #To know the users info in which line
    while (access == False):
        if(lists[line_number][1] == patientic):
            patientid, ic, name, old_password, gender, vc, address, postcode, age, email, phone_number, illness, allergic, vaccine = lists[line_number]#get the old record
            access = True
        else:
            line_number += 1
    f.close()

    print("\n--------------------Edit Password--------------------\n")
    while (True):
        comfirm = input("Please enter old password before edit your password : ")
        if (lists[line_number][3] == comfirm):
            new_password = input("Please enter new password : ")
            makesure = input("Do you really want to edit (yes/no) ?\n> ").lower()
            break
        else:
            print("\n* The password you entered is incorrect. *")
            confirm2 = input("Do you really want to edit your password (yes/no) ? \n>").lower()
            if (confirm2 == "yes"):
                continue
            else:
                makesure = "no"
                break

    while (True):
        if (makesure == "yes"):
            print("\n--------------------Edited Successfully--------------------")
            print("\n* Password has been edited from", old_password,"to",new_password,"*")
                #To update the new record to the file
            f = open("Patients.txt","a")
            #insert new record but haven deleted old record
            f.writelines([patientid,",",ic,",",name,",", new_password,",",gender,",", vc,",", address,",", postcode,",", age,",", email,",", phone_number,",", illness,",", allergic,",", vaccine])
            f.close()

            #delete old record and write new record into text file
            f = open("Patients.txt","r")
            lines = f.readlines()
            del lines[line_number]  #the lines contains the latest records(including the record which has been edited)
            f.close()

            f = open("Patients.txt","w")
            for line in lines:  #write all the latest record into text file
                f.write(line)
            f.close()
            break
        elif (makesure == "no"):
            print("\n*Nothing has changed*")
            break
        else:
            print("Invalid Input")
```

Figure 91: edit_password(patientic)

Variables / Functions in edit_password(patientic):

Variable / Function	Meaning
patientic	A parameter that receives the value from the argument in ‘edit_address_postcode’ function call.
line_number	Used to know which row the desired user's profile is on.
access	To control the while loop looping or stopping.
f	A variable that stores the way to open the patients text file, the way to open the text file in this ‘edit_phone_number ‘ function will be read, append and write mode.
lists	An empty list is used to store the record lines in the text file.
line	A variable that reads every looped record line from the ‘Patients’ text file, but only one record can be read at a time, and every time a new record is read the old one will be removed. This variable is used for appending purposes.

line_split	A variable to store the result after splitting the record in line by ‘,’. This variable is used for data positioning purposes.
patientid	A variable to store the patient ID of the desired patient.
ic	A variable to store the IC of the desired patient.
name	A variable to store the name of the desired patient.
old_password	A variable to store the old password of the desired patient.
gender	A variable to store the gender of the desired patient.
vc	A variable to store the vaccine centre of the desired patient.
address	A variable to store the address of the desired patient.
postcode	A variable to store the postcode of the desired patient.
age	A variable to store the age of the desired patient.
email	A variable to store the email of the desired patient.
phone_number	A variable to store the phone number of the desired patient.
illness	A variable to store whether the desired patient has an illness or not.
allergic	A variable to store whether the desired patient is allergic or not.
vaccine	A variable to store the selected vaccine of the desired patient.
confirm	A variable that stores the input of the old password to confirm users know the old password before they change to a new password.
new_password	A variable to store the new password of the desired patient.
makesure	A variable to store the input (yes/no) of whether the patient really wants to edit or not.
lines	A variable to store all the record lines from patients text file.

Explanation for edit_password(patientic):

The **main purpose** of this function is to let users edit their password. In this function, the users will be asked to enter their new password. After entering a new password, users will be asked whether the patients really want to edit or not. If the users enter 'yes', the old password will be replaced with a new one. If the users enter 'no', no changes will be made to the password.

Line 974: The variable 'line_number' is set to 0.

Line 975: The variable 'access' is set to False.

Line 977: Open the 'Patients.txt' in read mode.

Line 978: Create an empty list named 'lists'.

Lines 979-981: Using for loop to insert all the records in 'Patients.txt' to a list.

- Line 979: Using for loop to loop every record in the 'Patients.txt' text file.
- Line 980: Splitting the record in line in the 'Patients.txt' by ','.
- Line 981: Appending the record into the 'lists' list. Therefore, the 'lists' will become a nested list.

Lines 984-989: Using the while loop to know the info of the desired users and in which line.

- Line 984: A while loop is used until the info of the desired is gotten.
- Line 985-989: if…else statement is used. If the patient IC matches the patient IC in the 'lists' nested list, then the record line will be retrieved. Furthermore, the value for the access will also change to 'True'. If the patient IC is not on the 'lists' nested list, the value for variable 'line_number' will increase by 1 to test the next list in the nested list.

Line 990: The 'Patients.txt' is closed.

Line 992: Title of 'Edit Password' is printed.

Lines 993-1006 Using the while loop to know the info of the desired users in which line.

Line 993: A while loop is used until to get the valid input from user.

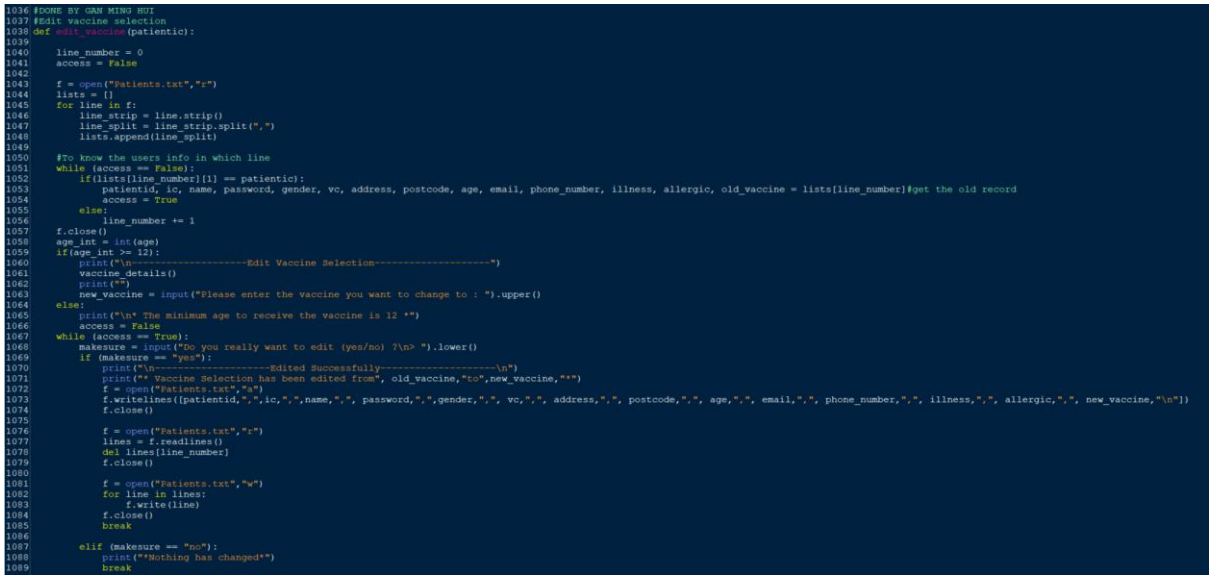
Line 994: Users will be prompted to enter their old password and the input will be stored in the variable 'confirm'.

Line 995-1006: if…else statement is used. Users will only be allowed to set a new password if the old one they entered is correct. Users will be prompted to enter their new password and the

input will be stored in the variable ‘new_password’. If the user enters an incorrect old password, they will not be able to set a password. Users will be prompted do they really want to change their password (yes/no) and the input will be converted to lowercase before being stored in the variable ‘makesure’.

Lines 1008-1033: Using the while loop to avoid re-enter the new password after answering an invalid entry

- Line 1008: A while loop is used until the users choose ‘yes’ or ‘no’.
- Line 1009-1033: if…elif…else statement is used. If the users enter ‘yes’, then they will be able to successfully change the phone number. If the users enter ‘no’, then nothing will be changed. If the users enter other than ‘yes’ or ‘no’, the ‘Invalid Input’ will be printed.
- Line 1013-1016: The ‘Patients.txt’ will be opened in append mode. The record with the new email will be inserted into ‘Patiens.txt’. The ‘Patients.txt’ will be closed.
- Line 1019-1022: The ‘Patients.txt’ will be opened in reading mode. All records in the text file will be read into the ‘lines’ variable. After that, the record with the old email will be deleted. The ‘Patients.txt’ will be closed.
- Line 1024-: The ‘Patients.txt’ will be opened in writing mode. The latest record will be rewritten into the ‘Patients’ text file. The ‘Patients.txt’ will be closed.



```
#DONE BY GAN MING HUI
#Edit vaccine selection
def edit_vaccine(patientic):

    line_number = 0
    access = False

    f = open("Patients.txt","r")
    lists = []
    for line in f:
        line_strip = line.strip()
        line_split = line_strip.split(",")
        lists.append(line_split)

    #To know the users info in which line
    while (access == False):
        if(lists[line_number][1] == patientic):
            patientid, ic, name, password, gender, vc, address, postcode, age, email, phone_number, illness, allergic, old_vaccine = lists[line_number]#get the old record
            access = True
        else:
            line_number += 1
    f.close()
    age_int = int(age)
    if(age_int >= 12):
        print("\n----------Edit Vaccine Selection----------")
        vaccine_details()
        print("")
        new_vaccine = input("Please enter the vaccine you want to change to : ").upper()
    else:
        print("\n* The minimum age to receive the vaccine is 12 *")
        access = False
    while (access == True):
        makesure = input("Do you really want to edit (yes/no) ?\n> ").lower()
        if (makesure == "yes"):
            print("\n----------Edited Successfully----------\n")
            print("* Vaccine Selection has been edited from", old_vaccine,"to",new_vaccine,"*")
            f = open("Patients.txt","a")
            f.writelines([patientid,",",ic,",",name,",", password,",",gender,",", vc,",", address,",", postcode,",", age,",", email,",", phone_number,",", illness,",", allergic,",", new_vaccine,"\n"])
            f.close()

            f = open("Patients.txt","r")
            lines = f.readlines()
            del lines[line_number]
            f.close()

            f = open("Patients.txt","w")
            for line in lines:
                f.write(line)
            f.close()
            break

        elif (makesure == "no"):
            print("*Nothing has changed*")
            break
```

Figure 92: edit_vaccine(patientic)

Variables / Functions in edit_vaccine(patientic):

patientic	A parameter that receives the value from the argument in ‘edit_address_postcode’ function call.
line_number	Used to know which row the desired user's profile is on.
access	To control the while loop looping or stopping.
f	A variable that stores the way to open the patients text file, the way to open the text file in this ‘edit_phone_number ‘ function will be read, append and write mode.
lists	An empty list is used to store the record lines in the text file.
line	A variable that reads every looped record line from the ‘Patients’ text file, but only one record can be read at a time, and every time a new record is read the old one will be removed. This variable is used for appending purposes.
line_strip	A variable to store the result after removing the spaces at the beginning and at the end of the record lines.
line_split	A variable to store the result after splitting the record in line by ‘,’. This variable is used for data positioning purposes.

patientid	A variable to store the patient ID of the desired patient.
ic	A variable to store the IC of the desired patient.
name	A variable to store the name of the desired patient.
password	A variable to store the password of the desired patient.
gender	A variable to store the gender of the desired patient.
vc	A variable to store the vaccine centre of the desired patient.
address	A variable to store the address of the desired patient.
postcode	A variable to store the postcode of the desired patient.
age	A variable to store the age of the desired patient.
email	A variable to store the email of the desired patient.
phone_number	A variable to store the phone number of the desired patient.
illness	A variable to store whether the desired patient has an illness or not.
allergic	A variable to store whether the desired patient is allergic or not.
old_vaccine	A variable to store the old selected vaccine of the desired patient.
age_int	A variable to store the age of the desired patient in form of integer.
vaccine_details( )	A function that prints out a menu with all the vaccines. The purpose of this variable is primarily to allow patients to see what vaccines are available when changing vaccinations.
new_vaccine	A variable to store the new vaccine selection of the desired patient.
makesure	A variable to store the input (yes/no) of whether the patient really wants to edit or not.

lines	A variable to store all the record lines from patients text file.

Explanation for edit_vaccine(patientic):

The **main purpose** of this function is to let users edit their selected vaccine. In this function, the users will be asked to enter the vaccine they want to change to. After entering a new vaccine, users will be asked whether the patients really want to edit or not. If the users enter 'yes', the old selected vaccine will be replaced with a new one. If the users enter 'no', no changes will be made to the selected vaccine.

Line 1040: The variable 'line_number' is set to 0.

Line 1041: The variable 'access' is set to False.

Line 1043: Open the 'Patients.txt' in read mode.

Line 1044: Create an empty list named 'lists'.

Lines 1045-1048: Using for loop to insert all the records in 'Patients.txt' to a list.

- Line 1045: Using for loop to loop every record in the 'Patients.txt' text file.
- Line 1046: Removing the spaces at the beginning and at the end of the record
- Line 1047: Splitting the record in line in the 'Patients.txt' by ','.
- Line 1048: Appending the record into the 'lists' list. Therefore, the 'lists' will become a nested list.

Lines 1051-1056: Using the while loop to know the info of the desired users and in which line.

- Line 1051: A while loop is used until the info of the desired is gotten.
- Line 1052-1056: if…else statement is used. If the patient IC matches the patient IC in the 'lists' nested list, then the record line will be retrieved. Furthermore, the value for the access will also change to 'True'. If the patient IC is not on the 'lists' nested list, the value for variable 'line_number' will increase by 1 to test the next list in the nested list.

Line 1057: The 'Patients.txt' is closed.

Line 1058: Title of 'Edit Vaccine Selection' is printed.

Lines 1059-1066: Using if…else statement to avoid people too young to receive the vaccine.

- Line 1059-1063: If the user is 12 years old or older, they will be allowed to change their vaccine.
- Line 1060: 'Edit Vaccine Selection' will be printed.
- Line 1061: The 'vaccine_details' function will be called (refer to figure 100).
- Line 1063 : Users will be prompted to enter their new selection of vaccine and the input will be stored in the variable 'new_vaccine'.
- Line 1064-1066: If the user's age is too young, 'Minimum age to receive the vaccine is 12' will be printed and the value for the 'access' value will become False.

Lines 1067-1089: Using the while loop to avoid re-enter the new vaccine selection after answering an invalid entry

- Line 1067: A while loop is used until the users choose 'yes' or 'no'.
- Line 1068: Users will be prompted do they really want to change their vaccine selection (yes/no) and the input will be converted to lowercase before being stored in the variable 'makesure'.
- Line 1069-1089: if…elif…else statement is used. If the users enter 'yes', then they will be able to successfully change the vaccine selection. If the users enter 'no', then nothing will be changed. If the users enter other than 'yes' or 'no', the 'Invalid Input' will be printed.
- Line 1072-1074: The 'Patients.txt' will be opened in append mode. The record with the new email will be inserted into 'Patiens.txt'. The 'Patients.txt' will be closed.
- Line 1076-1079: The 'Patients.txt' will be opened in reading mode. All records in the text file will be read into the 'lines' variable. After that, the record with the old email will be deleted. The 'Patients.txt' will be closed.
- Line 1081-1084: The 'Patients.txt' will be opened in writing mode. The latest record will be rewritten into the 'Patients' text file. The 'Patients.txt' will be closed.

```python
#*************************************************************************************************************************************************
#DONE BY GAN MING HUI
#Define a function to let the user view the options of vaccines and choose the vaccine they want to receive

def vaccine_selection(age_int):
    access = False

    while (access == False):
        print("\n--------------------Depending on the age of you, the following vaccines are available to you--------------------\n")

        #retrieve minimum and maximum age from vaccine.txt text file
        f = open("vaccine.txt","r")
        lists = []
        lists1 = []
        above = True
        n = 0
        a = 1
        for line in f:
            line_strip = line.strip()
            line_split = line_strip.split(",")
            lists.append(line_split)

            #Check which vc meets the age
            #prevent the max age in age group is the word "above"
            try:
                min_age_str = lists[n][3]
                min_age_int = int(min_age_str)
                max_age_str = lists[n][4]
                max_age_int = int(max_age_str)

            except:
                max_age_int = 0

        #Not allow the too young & old people to receive vaccine (12 & 80)
        if(age_int < 18):
            print(f"\n*Notice : Your age is {age_int} and haven reach the minimum age ")
            return("ineligible")

        elif(age_int > 80):
            print(f"\n*Notice : Your age is {age_int} and already reach the maximum age ")
            return("ineligible")

        #if the patient age greather than minimum age and smaller than maximum age of the vaccine, then print the vaccine details
        if((age_int >= min_age_int) and (age_int <= max_age_int)):
            print(f'{a}.{lists[n][0]} -- {lists[n][1]} | {lists[n][2]} | {lists[n][3]} - {lists[n][4]} age group')
            access = True

            #store all the vaccine which available to patient into a list (validation for the selection--make sure selection is within the available vaccine)
            lists1.append(lists[n][0])

            #numbering
            a = a + 1

        else:
            above = False


        #print the vaccine available for the word "above"
        if(above == False):
            if((age_int >= min_age_int) and (max_age_int == 0)):
                print(f'{a}.{lists[n][0]} -- {lists[n][1]} | {lists[n][2]} | {lists[n][3]} - {lists[n][4]} age group')
                access = True
                lists1.append(lists[n][0])
                a = a + 1
                n = n + 1

            else:
                n = n + 1
        else:
            n = n + 1
    #for age not meet the age group requirement
    if(access == False):
        print(f"\n*Notice : You are not eligible to receive vaccines because your age is only {age_int} ")
        return ("ineligible")

    while(True):
        print("\n*Enter '1' to exit")
        selection = input("Please enter the vaccine code you want to receive (only the VC code is accepted) : ").upper()

        if(selection == "1"):
            return selection

        #import current date time
        import datetime
        vac_date = datetime.date.today()

        #recommendate the patient receive first dose after 10 days registration
        vac_date = vac_date + datetime.timedelta(days= 10)

        #make sure user only select the vaccine which available to him
        if(selection in lists1):
            print(f"\n--------------------Registration Completed You can receive your {selection} vaccination on {vac_date}--------------------")
            return selection
        else:
            print("\n*Error : Invalid Input")

    f.close()
```

Figure 93: vaccine_selection(age_int)

Variables / Functions in vaccine_selection(age_int):

age_int	A parameter that receives the value from the argument in 'vaccine_selection' function call.
access	To control the while loop looping or stopping.

f	A variable that stores the way to open the vaccine text file, the way to open the text file in this 'vaccine_selection' function will be read mode.
lists	An empty list that uses to store the record lines in the text file.
lists1	An empty list that uses to store all the vaccines available to the patient.
above	A variable that uses to print the vaccine available to the patient, if the maximum age of the age group is 'above'.
n	A variable that represents the record line in the vaccine text file.
a	A variable that represents the numbering.
line	A variable that reads every looped record line from the 'vaccine' text file, but only one record can be read at a time, and every time a new record is read the old one will be removed. This variable is used for appending purposes.
line_strip	A variable to store the result after removing the spaces at the beginning and at the end of the record lines.
line_split	A variable to store the result after splitting the record in line by ','. This variable is used for data positioning purposes.
min_age_str	A variable to store the minimum age of the age group in form of string.
min_age_int	A variable to store the minimum age of the age group in form of integer.
max_age_str	A variable to store the maximum age of the age group in form of string.
max_age_int	A variable to store the maximum age of the age group in form of integer.

selection	A variable that stores the input of selection from the user.
vac_date	A variable that uses to store the current date time.

The **main purpose** of this function is to let the users view the options of vaccines and choose the vaccine they want to receive. In this function, the users will be asked to enter the vaccine they want to receive. After entering the vaccine, users will be recommended to receive the first dose of the vaccine after 10 days of registration.

Line 1097: The variable 'access' is set to False.

Lines 1099-1189: Using while loop to loop the entire function of 'vaccine_selection'

- Line 1099: Using while loop to loop the entire function of 'vaccine_selection'
- Line 1100: 'Depending on the age of you, the following vaccines are available to you' will be printed.
- Line 1103: Open the 'vaccine.txt' in read mode
- Line 1104: Create an empty list name 'lists'
- Line 1105: Create an empty list name 'lists1'
- The variable 'above' is set to True
- The variable 'n' is set to 0
- The variable 'a' is set to 1

Lines 1109-1161: Using for loop to insert all the records in 'vaccine.txt' into a list

- Line 1109: Using for loop to insert all the records in 'vaccine.txt' into a list
- Line 1110: Removing the spaces at the beginning and at the end of the record
- Line 1111: Splitting the record in line in the 'vaccine.txt' by ','.
- Line 1112: Appending the record into the 'lists' list. Therefore, the 'lists' will become a nested list.

Lines: 1116-1123: Using try and except to prevent the input of maximum age in the age group is the word 'above'.

- Line 1116: 'try' is used. If there is any error occurs in the try section, it will stop running the code in the try section.
- Line 1117: Retrieve the minimum age group from the nested list and store it in the 'min_age_str' variable.

- Line 1118: Convert the ‘min_age_str’ into the form of integer and store the result in the variable ‘min_age_int’.
- Line 1119: Retrieve the maximum age group from the nested list and store it in the ‘max_age_str’ variable.
- Line 1120: Convert the ‘max_age_str’ into the form of integer and store the result in the variable ‘max_age_int’.
- Line 1122: ‘except’ is used. If there is any error that occurs in the try section and it will jump to the except section.
- Line 1123: The variable ‘max_age_int’ will be set to 0.

Lines 1126-1132: Using if…elif statement to not allow the too young and old people to receive vaccine

- Lines 1126-1132: If the age of the users is below 18, then the users will be informed they are too young and cannot receive any of the vaccines. Or if the age of the users is greater than 80, then the users will be informed they are too old and cannot receive any of the vaccines.

Lines 1134-1146: Using if…else statement to confirm the age of the users is eligible to receive vaccine.

- Lines 1135-1146: If the age of the users is greater than the minimum age and smaller than the maximum age of the vaccine age group, then the vaccines that are available to the patient will be printed. All the vaccines available to the patient will also be stored in a list to make sure the selection made by the users is within the selection the system provided. Otherwise, nothing will be printed.

Lines 1150-1161: Using if…else statement to print the available vaccine if the maximum age of the age group is ‘above’

- Lines 1151-1159: If…else statement is used. If the age of the users is greater than the minimum age of the age group, then the available vaccines to them will be printed.

Lines 1164-1166: Using the if statement for age does not meet the age group requirement

- If the age of the users does not meet the age group requirement, they will be informed that they are not eligible to receive vaccine.

Lines 1168-1187: Using the while loop to loop again if the users enter an invalid input.

- Line 1169: The users will be prompted if they want to receive the registration they can enter ‘1’.
- Line 1170: The users are prompted to enter a choice of what vaccine they want to receive, and the input will be stored in the variable ‘selection’.
- Lines 1172-1173: ‘If statement’ is used. If the users enter 1 as the input, users will exit the registration.

Lines 1176-1180: Import current date time

- Line 1177: The variable ‘vac_date’ is used to store the current date time.
- Line 1180: The variable ‘vac_date’ will be added ‘10’ as a recommended date for the patients to receive the first dose.

Lines 1183-1187: If…else statement is used to make sure users only select the vaccine available to them.

- If the selection made by the users is within the ‘lists’ nested list, then ‘registration completed’ will be printed. Else, ‘invalid input’ will be printed.

Lines 1189: The ‘vaccine.txt’ will be closed.

```
1193 #DONE BY GAN MING HUI
1194 #Define Giving Patient ID
1195
1196 def id(vc):
1197     if (vc == "VC1"):
1198         #CALL THE LINE FUNCTION AND ASSIGN THE VALUE OF LINE TO VARIABLE NUM1
1199         num1 = line(vc)
1200         print(f"\nPatient ID of the patient is : VC1-{num1}")
1201         print("*You Can Start To Login Into Your Account")
1202         pat_id = (f"VC1-{num1}")
1203         #RETURN THE PATIENT ID TO THE id()FUNCTION
1204         return pat_id
1205
1206     elif (vc == "VC2"):
1207         #CALL THE LINE FUNCTION AND ASSIGN THE VALUE OF LINE TO VARIABLE NUM2
1208         num2 = line(vc)
1209         print(f"\nPatient ID of the patient is : VC2-{num2}")
1210         print("*You Can Start To Login Into Your Account")
1211         pat_id = (f"VC2-{num2}")
1212         #RETURN THE PATIENT ID TO THE id()FUNCTION
1213         return pat_id
```

Figure 94: id(vc)

Variables / Functions in id(vc):

Variable / Function	Meaning
vc	A parameter that receives the value from the argument in a function call.
num1	A variable that stores the return result of the function ‘line’.
line(vc)	‘vc’ is an argument that passes the value back to the parameter of ‘line’ function. This function will count how many times vc1 and vc2 appear in the text file respectively.
pat_id	A variable that stores the patient id.
num2	A variable that stores the return result of the function ‘line’.

Explanation for user id(vc):

The **main purpose** of the ‘id’ function is to generate patient ID for patients based on the VC they have selected. In this function, the patient's VC is first obtained, and once the VC is known, the 'line' function is called to find out how many patients are already in the text file. A new patient ID calculated from the ‘line’ function is then given to the patient who has just registered.

In lines 1197-1213: Patient ID

- Lines 1197-1213: if…else statement is used. If the patient's VC is VC1, then their ID will start with VC1 and be followed by the number after adding one to the total number of patients in VC1. If the patient's VC is VC2, then their ID will start with VC2 and be followed by the number after adding one to the total number of patients in VC2. This will be the method of giving patient id to newly registered patients.

```python
#DONE BY GAN MING HUI
#a.print the total number of the patients according to their center
def statistical_report():
    flag = True
    while(flag == True):
        #Show a menu
        print("\n--------------------Statistical Information--------------------")
        print("\n1.View Statistical Report\n2.Back\n")
        opt = input("Please enter the selection : ")

        #Choose the VC
        while(True):
            if(opt == "1"):
                print("\n--------------------View Statistical Report--------------------")
                print("\n1.Ststistics VC1\n2.Statistics VC2\n3.Back\n")
                selection = input("Please enter the selection : ")

                #VC1
                if(selection == "1"):
                    vc = "VC1"
                    vc_menu(vc)

                #VC2
                elif(selection == "2"):
                    vc = "VC2"
                    vc_menu(vc)

                #Back to choose vc menu
                elif(selection == "3"):
                    break

                else:
                    print("\n*Error : Invalid Input")

            #Back to statistics main menu and stop the process
            elif(opt == "2"):
                flag = False
                break

            else:
                print("\n*Error : Invalid Input")
                break
```

Figure 95: statistical_report( )

Variables / Functions in statistical_report()

Variable / Function	Meaning
flag	To control the while loop looping or stopping.
opt	A variable that stores the input of selection from the user (1-3).
selection	A variable that stores the input of selection from the user (1-3).
vc	A variable that stores the vaccine centre.
vc_menu(vc)	‘vc’ is an argument that passes the value back to the parameter of ‘vc_menu’ function. This function will print a menu with five statistical options.

The **main purpose** of the ‘statistical_report’ function is to let the admin choose whether to see the VC1 or VC2 statistics report. Each center's report contains five statistical options. That is ‘total patient’, ‘total patients vaccinated’, ‘patients who are waiting for dose 2’ and ‘patients who have completed receiving all the doses’.

Line 1483: The variable ‘flag’ is set to ‘True’.

Lines 1484-1521: The while loop is used until getting the valid input in the ‘Statistical Information’ page.

Lines 1486-1488: The users are prompted to enter a choice and the input will be read by the variable ‘opt’.

Lines 1491-1521: The while loop is used until getting the valid input in the ‘View Statistical Report’ page.

Lines 1492-1521: The if…elif….else statement is used. If the users enter ‘1’ as their input, they will be taken to the ‘View Statistical Report’ page. If the users enter ‘2’ as their input, they will back to admin main menu. Other than the input ‘1-2’, the ‘invalid input’ will be printed.

Liens 1498-1512: The if…elif….else statement is used. If the users enter ‘1’ as their input, the variable ‘vc’ will equal ‘VC1’ and the argument of the ‘vc_menu’ function call will be ‘VC1’. If the users enter ‘2’ as their input, the variable ‘vc’ will equal ‘VC2’ and the argument of the ‘vc_menu’ function will be ‘VC2’. If the users enter ‘3’ as their input, they will be taken back to the ‘statistical Information’ page. Other than the input ‘1-3’, the ‘invalid input’ will be printed.

```
#DONE BY GAN MING HUI
#define a menu for vc statistics report

def vc_menu(vc):
    if(vc == "VC1"):
        print("\n-------------------VC1 Statistics Report------------------")
        total_patients(vc)
        vaccinated(vc)
        wait_dose2(vc)
        completed(vc)


    elif(vc == "VC2"):
        print("\n-------------------VC2 Statistics Report------------------")
        total_patients(vc)
        vaccinated(vc)
        wait_dose2(vc)
        completed(vc)
```

Figure 96: vc_menu(vc)

Variables / Functions in vc_menu(vc):

Variable / Function	Meaning
vc	A parameter that receives the value from the argument in 'vc_menu' function call.
total_patients(vc)	'vc' is an argument that passes the value back to the parameter of 'total_patients' function. This function allows admins to know the total number of patients in VC1 and VC2.
vaccinated(vc)	'vc' is an argument that passes the value back to the parameter of 'vaccinated' function. This function allows admins to know the total number of patients vaccinated.
wait_dose2(vc)	'vc' is an argument that passes the value back to the parameter of 'wait_dose2' function. This function allows admins to know the total number of patients who are waiting for dose 2.
completed(vc)	'vc' is an argument that passes the value back to the parameter of 'completed' function. This function allows admins to know the total number of patients who have completed receiving all the doses.

The **main purpose** of the ‘vc_menu’ function is to print the statistics report of vaccine centre. The statistics report consists of five statistical options.

Lines 1528-1542: The if…else statement is used. If the value passed back to the parameter by the argument of the ‘vc_menu’ function call is ‘1’, the statistics report of VC1 will be showed. Or if the value passed back to the parameter by the argument of the ‘vc_menu’ function call is ‘2’, the statistics report of VC2 will be showed.

```
17 #DONE BY GAN MING HUI
18 #Let admin edit vaccine
19
20 def vaccine_edit():
21
22     while(True):
23         vaccine_details()
24         print("\n--------------------Edit Vaccination--------------------")
25         print("\n1.Add New Vaccines\n2.Delete Existing Vaccines\n3.Back\n")
26
27         opt = input("Please Enter the selection : ")
28
29         if(opt == "1"):
30             add_vaccine()
31
32         elif(opt == "2"):
33             delete_vaccine()
34
35         elif(opt == "3"):
36             break
37
38         else:
39             print("\n*Error : Invalid Input")
```

Figure 97: vaccine_edit( )

Variables / Functions in vaccine_edit()

vaccine_detail( )	A function that allows users to view all the vaccines available in the vaccine centre.
opt	A variable that saves the input of selection from the user (1-3).
add_vaccine( )	A function that allows admins to add new vaccines to the vaccine centre.
delete_vaccine( )	A function that allows admins to delete vaccines from the vaccine centre.

The **main purpose** of this function is to let admins edit the vaccine in the vaccine centre. In this function, the users will be asked to select ‘1’ to add a new vaccine, select ‘2’ to delete the vaccine, or select ‘3’ to back to the previous page. If input entered by the users is other than 1-3, the ‘invalid input’ will be printed.

Lines 22-39: A while loop is used to get a valid input.

Line 23: The ‘vaccine_details’ function is called.

Lines 24-27: Users are prompted to enter an option to add new vaccines, delete existing vaccines, or back to the previous page. The selection of users will be read and stored in the ‘opt’ variable on line 27.

Lines 29-39: If…elif…else statement is used. If the users enter ‘1’ as their input, the ‘add_vaccine’ function will be called (refer to figure 129), if the users enter ‘2’ as their input, the ‘delete_vaccine’ function will be called (refer to ?), or if the users enter ‘3’ as their input, the users will be backed to the previous page. If the user enters an input other than the input ‘1’, ‘2’ or ‘3’, then the ‘Invalid Input’ will be printed and continue to loop.

```
112 #DONE BY GAN MING HUI
113 #Validation for dosage required
114
115 def validation_dosage():
116     regex_dosage = '([1-3]{1})'
117
118     while(True):
119         print("\n*Enter 'exit' to exit")
120         dosage_required = input("Please enter the number of dosage required (Only 1-3 are accepted) : ")
121
122         if(dosage_required == "exit"):
123             return dosage_required
124
125         if(len(dosage_required) == 1):
126
127             if(re.search(regex_dosage,dosage_required)):
128                 return dosage_required
129
130             else:
131                 print("\n*Error : Invalid dosage required")
132
133         else:
134             print("\n*Error : Invalid dosage required")
```

Figure 98: validation_dosage( )

Variables / Functions in validation_dosage( ):

Variable / Function	Meaning
regex_dosage	A regular expression that determines the pattern of the dosage required.
Dosage_required	A variable that stores the input of dosage required.

Explanation for validation_dosage( ):

The **main purpose** of 'validation_dosage' function is to validate the dosage entered by the users. In this function, the admins will be asked to enter the dosage required for the new vaccines. The dosage required entered must be between 1-3. Otherwise, 'Invalid dosage required' will be printed.

Line 116: Regular expression for dosage, means the dosage required entered by the users must be a number between 1-3.

Lines 118-134: A while loop is used until users enter a valid input.

Line 119: Users will be prompted they can enter '1' to exit the current page.

Line 120: Users will be prompted to enter the dosage required for the new vaccine and the input will be stored in the variable 'dosage_required'.

Lines 122-123: A if statement will be used. If the users enter 'exit' as their input, they will exit from the current page and back to the main menu of the vaccination system.

Lines 125-134: A if...else statement is used. If the length of the input entered by the users is 1, then will be a valid input. Otherwise, ‘Invalid dosage required’ will be printed.

Lines 127-131: A if…else statement is used. If the input entered by the users is meet the pattern of the dosage required regular expression, then the input is valid input. Otherwise, ‘Invalid dosage required’ will be printed.

```
137 #DONE BY GAN MING HUI
138 #Validation for interval
139
140 def validation_interval():
141     regex_interval = '([1-4]{1})'
142
143     while(True):
144         print("\n*Enter 'exit' to exit")
145         interval = input("Please enter the interval between doses in weeks (Only 1-4 or 'no' are accepted) : ")
146
147         if(interval == "exit"):
148             return interval
149
150
151         if(len(interval) == 1):
152
153             if(re.search(regex_interval,interval)):
154                 return interval
155
156             else:
157                 print("\n*Error : Invalid interval")
158
159         elif(interval == "no"):
160             return interval
161
162         else:
163             print("\n*Error : Invalid interval")
```

Figure 99: validation_interval( )

Variables / Functions in validation_interval( ):

Variable / Function	Meaning
regex_interval	A regular expression that determines the pattern of the interval.
interval	A variable that stores the input of interval.

Explanation for validation_interval( ):

The **main purpose** of 'validation_interval' function is to validate the interval entered by the users. In this function, the admins will be asked to enter the interval for the new vaccines. The interval entered must be between 1-4. Otherwise, 'Invalid interval' will be printed.

Line 141: Regular expression for dosage, means the interval entered by the users must be a number between 1-4.

Lines 143-163: A while loop is used until users enter a valid input.

Line 144: Users will be prompted they can enter '1' to exit the current page.

Line 145: Users will be prompted to enter the interval for the new vaccine and the input will be stored in the variable 'interval'.

Lines 147-148: A if statement will be used. If the users enter 'exit' as their input, they will exit from the current page and back to the main menu of the vaccination system.

Lines 151-163: A if...elif…else statement is used. If the length of the input entered by the users is 1 or ‘no’, then will be a valid input. Otherwise, ‘Invalid dosage required’ will be printed.

Lines 153-157: A if…else statement is used. If the input entered by the users is meet the pattern of the interval regular expression, then the input is valid input. Otherwise, ‘Invalid interval’ will be printed.

```
300 #DONE BY GAN MING HUI
301 #Let users view Vaccine details
302
303 def vaccine_details():
304     print("\n--------------------Vaccine Details--------------------\n")
305     f = open("vaccine.txt","r")
306     n = 0
307     a = 1
308     lists = []
309     for line in f:
310         line_strip = line.strip()
311         line_split = line_strip.split(",")
312         lists.append(line_split)
313
314         #print the available vaccines
315         print(f'{a}.{lists[n][0]} -- {lists[n][1]} | {lists[n][2]} | {lists[n][3]} - {lists[n][4]} age group')
316         n = n + 1
317         a = a + 1
318     f.close()
```

Figure 100: vaccine_details( )

f	A variable that stores the way to open the vaccine text file, the way to open the vaccine text file in this 'vaccine_details' function will be read mode.
n	A variable that represents the record line in the vaccine text file.
a	A variable that represents the numbering.
lists	An empty list is used to store the record lines in the text file.
line	A variable that reads every looped record line from the 'vaccine' text file, but only one record can be read at a time, and every time a new record is read the old one will be removed. This variable is used for appending purposes.
line_strip	A variable to store the result after removing the spaces at the beginning and at the end of the record lines.
line_split	A variable to store the result after splitting the record in line by ','. This variable is used for data positioning purposes.

The **main purpose** of the 'vaccine_details' function is to allow users to view all the vaccines in the vaccine centre. Once this function is called, a menu with all the vaccines details will be printed.

Line 304: The title 'Vaccine Details' is printed.

Line 305: The 'vaccine.txt' is opened in read mode.

Line 306: The variable 'n' is set to 0.

Line 307: The variable ‘a’ is set to 1.

Line 308: Create an empty list named ‘lists’.

Lines 309-317: A for loop is used to insert all the record lines in the vaccine text file into a list. After inserting all the record lines into the empty lists, all the vaccines in the vaccine centre will be printed by retrieving the details from the list.

Line 318: The vaccine text file will be closed.

```
2100 #DONE BY GAN MING HUI
2101 #Main Program
2102 import re #import regular expression
2103
2104 flag = True
2105 while (flag):
2106     menu()
```

Figure 101: Main Program

As shown in figure 46, the main program of our vaccine system will start with importing the regular expression. In lines 2104-2106 the while loop is used; the flag will be set to 'True' to allow the program to loop. The function of the main menu for this program will be called in the main program.

4.2 - HO FENG SHENG

```
def vac_status(patientic):
    flag = False
    f = open("Patients.txt","r")
    for line in f:
        if (patientic) in line:
            patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine = line.split(",")
            patient_id = patientid
    f.close()

    f = open("vaccination.txt", "r")
    for i in f:
        if i.startswith(patient_id):
            i = i.replace("\n","").split(",")
            flag = True
            if i[2] == 'AF'or i[2] == 'BV'or i[2] =='CZ'or i[2] =='DM':
                print("Vaccination Code :" + i[2])
                if i[1] == 'dose1':
                    print("Dose 1 status             :Completed")
                    print("Dose 1 completed date     :"+i[4])
                elif i[1] == 'dose2':
                    print("Dose 1 status             :Completed")
                    print("Dose 1 completed date     :"+i[4])
                if i[3]=='NONE':
                    print("Dose 2 status             :Completed")
                    print("Dose 2 completed date     :"+i[5])
                else:
                    print("Dose 2 status             :Incompleted")
                    print("Suggested for Dose 2 date :"+i[3])
            elif i[2] == 'EC':
                print("Vaccination Code :" + i[2])
                print("Dose 1 status             :Completed")
                print("Dose 1 completed date     :"+i[4])
                print("Dose 2 status             :Only for one dose")
    f.close()
    if flag == False:
        #print("Invalid Patient ID")
        f = open("Patients.txt","r")
        flag2 = False
        for i in f :
            if i.startswith(patient_id):
                flag2 = True
                ID,ic,nm,password,gender,VC,add,pc,age,email,no_ph,date_dose1,date_dose2,vaccine_code = i.replace("\n","").split(',')
                print("Vaccination Code :" + vaccine_code)
                print("Dose 1 status             :Incompleted")
                print("Dose 2 status             :Incompleted")

        if flag2 == False:
            print("Invalid patient id")

        f.close()
```

Figure 102: Code of vac_status() function

This function is designed to allow users to check the status of their vaccinations. First, there is a flag variable that will be set to False as a condition of the vaccination status function. After that, the "Patients.txt" file will be opened in reading mode and the system will read the data line by line. When the system reads a line starting with "patientic", it will divide all the data into 14 variables with ",". The "patientid" variable will then be set to "patient_id" in order to extract the data in the following code. After use, the "Patients.txt" file will be closed.

Next, the system will open the "Vaccination.txt" file in reading mode. It will read each line in the text file one by one, and if the data starts with "patient_id", it will stop and replace the new line with "" (blank) and split all variables with ",". If it meets the condition, the flag variable will change to "True". Next, the system will check if the vaccination code of i[2] in the vaccination.txt file is "AF", "BV", "CZ" or "DM". If it meets the condition, it will display the user's vaccination code. And if i [1] is the same as "Dose 1" and "Dose 2", the system will display the user's Dose 1 status and the time they completed Dose 1 based on the vaccination.txt file. If i [3] in the vaccination.txt file is "NONE", it will display the "Dose 2" status as

completed and the time when "Dose 2" was completed. On the contrary, if i [3] in the text file is not "NONE", it will show the Dose 2 status as incomplete and show the suggested date of Dose 2 calculated by the function of check_dose2. In addition, if i[2] is the same as "EC", it will display the inoculation code as EC and show the status of Dose 1 and the completion time. The Dose 2 status section will also remind the user that EC only receives one dose. After that, the vaccination.txt file will be closed and will go to the next section.

In the last paragraph, it is used to check if the user has completed registration but has not completed any doses yet. First, if the flag variable is False, the system will open the Patients.txt file in reading mode and set another conditional variable flag2 to False. After that, it will read each line of the text file one by one and if the data starts with "patient_id", it will stop and replace the new line with " " and split all variables with ",". Next, the vaccine code variables will be read and displayed. Also, the status of dose1 and dose2 will be displayed as incomplete. And if the flag2 variable is "False", the "Invalid Patient ID" will be displayed to the user. Finally, the Patients.txt file will be closed again by the system.

```
def vac_status_admin(patient_id):

    flag = False
    f = open("vaccination.txt", "r")
    for i in f:
        if i.startswith(patient_id):
            i = i.replace("\n","").split(",")
            flag = True
            if i[2] == 'AF'or i[2] == 'BV'or i[2] =='CZ'or i[2] =='DM':
                print("Vaccination Code :" + i[2])
                if i[1] == 'dose1':
                    print("Dose 1 status             :Completed")
                    print("Dose 1 completed date     :"+i[4])
                elif i[1] == 'dose2':
                    print("Dose 1 status             :Completed")
                    print("Dose 1 completed date     :"+i[4])
                if i[3]=='NONE':
                    print("Dose 2 status             :Completed")
                    print("Dose 2 completed date     :"+i[5])
                else:
                    print("Dose 2 status             :Incompleted")
                    print("Suggested for Dose 2 date :"+i[3])
            elif i[2] == 'EC':
                print("Vaccination Code :" + i[2])
                print("Dose 1 status             :Completed")
                print("Dose 1 completed date     :"+i[4])
                print("Dose 2 status             :Only for one dose")
    f.close()
    if flag == False:
        #print("Invalid Patient ID")
        f = open("Patients.txt","r")
        flag2 = False
        for i in f :
            if i.startswith(patient_id):
                flag2 = True
                ID,ic,nm,password,gender,VC,add,pc,age,email,no_ph,date_dose1,date_dose2,vaccine_code = i.replace("\n","").split(',')
                print("Vaccination Code :" + vaccine_code)
                print("Dose 1 status             :Incompleted")
                print("Dose 2 status             :Incompleted")

        if flag2 == False:
            print("Invalid patient id")

        f.close()
```

Figure 103: Code of vac_status_admin(patient_id) function

This vac_status_admin function is not much different from the previous vac_status function. In this vac_status_admin function, it will be displayed in the administration interface, while vac_status will be displayed in the user interface. The only difference between the two is that the vac_status_admin function starts by opening the vaccination.txt file directly, while vac_status must open the Patients.txt file and set the "patientid" variable to "patient_id" before moving on to the next section.

```
def user_info_admin():
    while (True):
        print("\n--------------------Patient Record--------------------")
        print("\n1.Search a patient\n2.All patients\n3.Back\n")

        opt = input("Pls enter the selection : ")
        if (opt == "1"):
            patient_id = one_user_info_admin()
            if(patient_id == "1"):
                return

        elif (opt == "2"):
            all_user_info_admin()

        elif (opt == "3"):
            break

        else:
            print("Invalid Input")
```

Figure 104: Code of user_info_admin() function

This function is designed for the patient record menu of the admin interface. The second line of code sets the system to display the patient record and menu if the condition is "true", which are "1. Search for a patient", "2. All patients", and "3. Back". Next, the administrator will be prompted to enter a selection. If the administrator's selection is "1", it will set the one_user_info_admin() function to patient_id, and if patient_id is "1", it will return the patient_id data to the one_user_info_admin() function and direct the administrator to the one_user _info_admin page. In addition, if the administrator enters "2" in the selection, the system will run the all_user_info_admin() function, and if the administrator enters "3" in the selection, it will break the loop and go back to the previous function, which is the admin_menu(). Finally, if the administrator enters a number that is not 1 to 3, the system will display "Invalid Input" and loop back to the menu, and the administrator will be prompted to enter the selection again.

```
def one_user_info_admin():
    f = open("Patients.txt","r")
    print("\n*Enter '1' to exit")
    patient_id = input("Please Enter the Patient ID : ").upper()

    if(patient_id == "1"):
        return patient_id

    #link to user_info for user
    flag = True
    for line in f:

        #SHOW THE USER INFO IF AND ONLY IF THE PATIENT ID EXIST IN TEXT FILE
        if patient_id in line:

            #Giving a variable to the data in text file in order
            patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine = line.split(",")
            print("\n--------------------This is the Personal Details of Patient", patient_id,"--------------------\n")
            print(": Patient ID\t: ", patientid)
            print(": IC\t\t: ", ic)
            print(": Name\t\t: ", name)
            print(": Gender(m/f)\t: ", gender)
            print(": VC(vc1/vc2)\t: ", vc)
            print(": Address\t: ", address)
            print(": Postcode\t: ", postcode)
            print(": Age\t\t: ", age)
            print(": Email\t\t: ", email)
            print(": Phone Number\t: ", phonenumber)
            print(": Illness\t: ", illness)
            print(": Allergic\t: ", allergic)
            print(": Vaccine\t: ", vaccine)
            flag = False

            vac_status_admin(patient_id)


    if (flag == True):
            print("\n*Error : The patient id cannot be found")

    f.close()
```

Figure 105: Code of one_user_info_admin() function

In this function, it is used to display the records of a specific patient. First, the system will open the Patients.txt file in read mode and display "Enter '1' to exit". Then, the administrator will be prompted to enter the patient ID he wants to search for, and the information entered by the administrator will be generated in uppercase letters. After that, if the administrator enters '1', the data will be returned, and the page will be exited back to the administrator's main menu. After this, the system will set the flag variable to True, and then the system will read the data in the Patients.txt file line by line. When the system reads the same line as the entered patient_id, it will divide all the data into 14 variables with ",". After that, the personal details of all specific patients will be displayed according to the variables saved in the text file. Next, the flag variable will be set to "false" and the vac_status_admin(patient_id) function will be run, which will display the vaccination status of the patient the administrator wants to search for. And if the flag variable is "true", it will display "Error: The patient id could not be found" on the admin page. Finally, the Patients.txt file will be closed after use.

```
def all_user_info_admin():
    count = 0
    f = open("Patients.txt","r")
    for line in f:
        count = count + 1 #INCREMENT -- Will show all the patients info one by one
        print("\n--------------------This is the Personal Details of Patient", count,"--------------------\n")
        #Givin a variable to the data in text file in order
        patient_id, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine = line.split(",")
        print(": Patient ID\t: ", patient_id)
        print(": IC\t\t: ", ic)
        print(": Name\t\t: ", name)
        print(": Gender(m/f)\t: ", gender)
        print(": VC(vc1/vc2)\t: ", vc)
        print(": Address\t: ", address)
        print(": Postcode\t: ", postcode)
        print(": Age\t\t: ", age)
        print(": Email\t\t: ", email)
        print(": Phone Number\t: ", phonenumber)
        print(": Illness\t: ", illness)
        print(": Allergic\t: ", allergic)
        print(": Vaccine\t: ", vaccine)
        print(" ")
        #add on what vc the patient take + vc date....

        vac_status_admin(patient_id)
    f.close()
```

Figure 106: all_user_info_admin() function

This function, the administrator will use this to view the information of all registered patients. First, the count variable will be set to 0 for the counting purpose in the following codes. Next, the Patients.txt file will be opened in read mode and then, the system will read it line by line. When the system reads a line of data in Patients.txt, it will increase the count variable by 1. Then, it will display the patient details and show the number of patients registered in the text file and it will divide all the data into 14 variables with ",". After that, all the patient details will be displayed according to the variables saved in the text file. At the same time, the system will run the vac_status_admin(patient_id) function to display the vaccination status of the patient. Finally, Patients.txt will be closed after use.

```
def user_info_user(patient_id): #get the patient id from other function
    f = open("Patients.txt","r")

    #loop every line in text file, will get the patient details when patient id exist in any of the line
    for line in f:
        if patient_id in line:
            #Divide the data of a record in the text file into their respective variables
            patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine = line.split(",")
            print("\n--------------------Personal Details--------------------\n")
            print(": Patient ID\t: ", patientid)
            print(": IC\t\t: ", ic)
            print(": Name\t\t: ", name)
            print(": Gender(m/f)\t: ", gender)
            print(": VC(vc1/vc2)\t: ", vc)
            print(": Address\t: ", address)
            print(": Postcode\t: ", postcode)
            print(": Age\t\t: ", age)
            print(": Email\t\t: ", email)
            print(": Phone Number\t: ", phonenumber)
            print(": Illness\t: ", illness)
            print(": Allergic\t: ", allergic)
            print(": Vaccine\t: ", vaccine)
    f.close()
```

Figure 107: user_info_user(patient_id) function

This function is not much different from the previous functions one_user_info_admin() and all_user_info_admin(). The previous functions are designed for the administration interface, while this function is designed for the user interface. First, the Patients.txt file will be opened in read mode and the system will read it line by line. If patient_id (obtained from other functions) is on that line, a data record in the text file will be put into the corresponding variable and the details of that patient will be displayed. Finally, the Patient.txt file will be closed after use.

```
def exist_user():
    access1 = False
    access2 = False
    a = False
    b = False

    while (access1 == False):
        print("\n--------------------Please login to your personal account--------------------")


        #login ic
        print("\n*Enter '1' to exit")
        ic = input("Please enter your IC : ")

        #exit login
        if(ic == "1"):
            return

        #login password
        print("\n*Enter '1' to exit")
        passw = input("Please enter your password : ")

        #exit login
        if(passw == "1"):
            return

        i = 0

        f = open("Patients.txt", "r")
        list1 = []
        for line in f:
            line_strip = line.strip()
            line_split = line_strip.split(",")
            list1.append(line_split)

            #matching the user input with the text file
            if((list1[i][1] == ic) and (list1[i][3] == passw)):
                print("\n--------------------Login Successfully--------------------")
                print("\n--------------------Welcome Back--------------------")
                user_menu(ic) #successfully enter into the acc
                access1 = True #avoid enter the first WHILE LOOP again / avoid enter the second WHILE Loop
                break #break the for loop
            elif((list1[i][1] != ic) or (list1[i][3] != passw)):
                i+=1

            else:
                #PREVENT INVALID INPUT
                access1 = False
                access2 = False

        #ask whether want to continue login if the login is failed
        if(access1 == False):
            print("\n*Error : Invalid Account")

            while (access2 == False):
                again = input("Do you still want to log in (yes/no) : ").lower()  #MAKE SURE THE INPUT IS ALL LOWER CASE
                if (again == "yes"):
                    break

                elif (again == "no"):
                    access1 = True
                    break

                else:
                    print("\n*Error : Invalid Input")

        f.close()
```

Figure 108: Code of exist_user() function

This function is designed as a login function for those patients who have completed registration. First, the system will set 4 variables to "False", namely access1, access2, a, and b. When the access1 variable is in the "False" state, the system will display "Please log in to your personal account" and "Enter '1' to exit" on the next line. At the same time, the system will prompt the user to enter their IC number to log in. After that, the system will display "Enter '1' to exit" again on the login password page and prompt the user to enter their password. If the user enters "1" on the login IC page or the password page, they will exit the login and return to the previous function menu().

Next, the system will set the "i" variable to 0 and open the patient.txt file in read mode. In addition, the "list1" variable will be set to a list variable. The system will read the patient.txt file line by line and if the IC and password entered by the user match with list[i][1] and [i][3] in list1, it will display "Login Successfully" and "Welcome back" on the user interface and run the user_menu(ic) function. After that, it will set the access1 variable to True and break the

loop. If the ic and password entered by the user do not match [i][1] and [i][3] in list 1, the "i" variable will be increased by 1. Otherwise, the access1 and access2 variables will be set to False condition to prevent invalid input.

If the access1 variable is in "False" condition, it will display "Error: Invalid Account" on the user interface . And if the access2 variable is "False", it will prompt the user if they still want to continue logging in, with all answers in lowercase. If the user enters "yes", it will break the loop and go back to the login page as its access1 variable is still in the "False" state, while if the user enters "no", the system will set the access1 variable to "True" and break the loop. If the user's answer is not "yes" or "no", it will display "Error. Invalid Input" is displayed on the user interface. The patient.txt file will be closed after the user finishes logging in.

```
def line(vc):
    f = open("Patients.txt","r")
    count1 = 101
    for line in f:
        if line.startswith(vc):
            count1 = count1 + 1
    f.close()
    return count1
```

Figure 109: Code of line(vc) function

This function is used to calculate the number of lines in the patient.txt file where vc1 or vc2 is present. First, the system will open the patient.txt file in read mode and set the count1 variable to a value of 101. The system will read the patient.txt file line by line and if the system finds a line starting with VC, the count1 variable will be incremented by 1. After that, patient.txt will be closed and the new value will be saved and returned to the count1 variable.

```
def name_list():
    f = open("Patients.txt","r")
    print("\n--------------------Patient list--------------------")
    a = 1
    for line in f:
        patientid, ic, name, password, gender, vc, address, postcode, age, email, phonenumber, illness, allergic, vaccine = line.split(",")
        print(f"\n{a}. Name\t:",name, "\t\tID :",patientid)
        a = a + 1
```

Figure 110: Code of name_list() function

This function is used for administrators to know each individual before searching for a specific patient. First, the system will open the patients.txt file in read mode and display the "patient list". After that, the 'a' variable will be set to 1. The system will start reading the patients.txt file line by line and split each record element in 'patients.txt' into different variables by ','. The details of the registered patient will be displayed, for example, the name and patient ID based on the values of the name and patientid variables, and then the 'a' variable will be incremented by 1.

```
def total_patients(vc):
    #print("\n--------------------Total Patients-------------------")
    count = 0
    f = open("Patients.txt","r")
    for line in f:
        if line.startswith(vc):
            count = count + 1
    print("\nTotal Number of Patients\t\t\t: ", count)
    f.close()
```

Figure 111: Code of total_patients(vc) function

This function is used to calculate the total number of patients in the vaccination center. First, the count variable will be set to 0. Then the Patients.txt file will be opened in read mode. The system will read the text file line by line, and if it finds a line starting with "vc", the count variable will be increased by 1. Finally, it will display the total number of patients based on the value of the count variable and close the text file.

```
def vaccinated(vc):
    #print("\n--------------------Patients Vaccinated-------------------")
    count = 0
    f = open("vaccination.txt","r")
    for line in f:
        if ((line.startswith(vc)) and ("dose" in line)):
            count = count + 1
    print("\nTotal Number of Vaccinated Patients\t\t: ", count)
    f.close()
```

Figure 112: Code of vaccinated(vc) function

This function is used to calculate the total number of vaccinated patients in the vaccination center. First, the count variable will be set to 0. Then the Patients.txt file will be opened in read mode. The system will read this text file line by line and if it finds a line starting with "vc" and "dose" in that line, the count variable will be increased by 1. Finally, it will display the total number of vaccinated patients based on the value of the count variable and close the text file.

```
def wait_dose2(vc):
    #print("\n--------------------Patients Waiting Dose 2-------------------")
    count = 0
    f = open("vaccination.txt","r")
    for line in f:
        if ((line.startswith(vc)) and ("dose1" in line)):
            count = count + 1
    print("\nTotal Number of Patients who Waiting for Dose 2 : ", count)
    f.close()
```

Figure 113: Code of wait_dose2(vc) function

This function is used to calculate the total number of patients waiting for a second dose at the vaccination center. First, the count variable will be set to 0. Then the Patients.txt file will be opened in read mode. The system will read this text file line by line and if it finds lines starting with "vc" and "dose1", the count variable will be increased by 1. Finally, it will display the total number of patients waiting for the second dose based on the value of the count variable and close the text file.

```
def completed(vc):
    #print("\n--------------------Patients Complete Vaccine-------------------")
    count = 0
    f = open("vaccination.txt","r")
    for line in f:
        if ((line.startswith(vc)) and ("dose2" in line)):
            count = count + 1
    print("\nTotal Number of Patients who completed their vaccine : ", count)
    f.close()
```

Figure 114: Code of completed(vc) function

This function is used to calculate the total number of patients who have completed their vaccinations at the vaccination center. First, the count variable will be set to 0. Then, the Patients.txt file will be opened in read mode. The system will read this text file line by line, and if it finds lines starting with "vc" and "dose2", the count variable will be increased by 1. Finally, it will display the total number of patients who completed vaccinations based on the value of the count variable and close the text file.

4.3 - DYANIEL CHING CHEE XIONG

checking_id()

```
'''MAKE SURE KEY IN VALID PATIENT ID'''
def checking_id():
    print("\n*Enter '1' to exit")
    id = input('Pls enter patient id: ').upper()
    if(id == "1"):
        return id

    f = open('Patients.txt', 'r')
    flag = False

    # check patient id
    for check in f:
        if (check.startswith(id)):
            flag = True
            break

    #Invalid patient id
    if (flag == False):
        print('Invalid patient id, pls re-submit.\n')
        id = checking_id()
    return id
    f.close()
```

Figure 115: code of checking_id() function

This function is used to check whether the patient id which key in is a patient id that has been registered. The first paragraph is asking the user to key in the patient id. If the user keys in "1", system will return the id to the function and end this function.

In the second paragraph, the **"Patients.txt"** text file will be opened in **reading mode** and the third paragraph system will read the text file line by line and stop reading when the key-in patient id is found. If there is no key-in patient id in the text file, the system will run the next paragraph code which **displays "Invalid patient id"** to user and ask user **key in patient id again.**

In the end of function, the "Patients.txt" text file will be closed and **return the patient id** that key in by user to the function.

checking_age(id)

```
'''DETERMINE PATIENT AGE'''
def checking_age(id):
    f = open('Patients.txt','r')
    #get patient's age
    for check in f:
        if (check.startswith(id)):
            ID,ic,nm,password,gender,VC,add,pc,age,email,no_ph,date_dose1,date_dose2,vaccine_code = check.split(',')

    return age
    f.close()
```

Figure 116: code of checking_age(id)function

This function is used to **get the patient's age** from "Patients.txt" text file for use in the next function. First, the **"Patients.txt"** text will be opened in **reading mode** and the system will read the data line by line in the text file. When the system reads the line starting with the patient id which is keyed in by user, it will split all the data in the line into 14 variables and **return the age of patient** to the function. In the end, the "Patients.txt" text file will be closed after use.

approve (type, age)

```
'''VACCINE APPROVEMENT ACCORDING AGE'''
def approve(type, age):
    age = int(age)
    #display AF or DM vaccine error
    if (type == 'AF'or type == 'DM'):
        if (age < 12) :
            print('\nThis patient is not suitable for having this dosage')
            flag = False
        else:
            flag = True

    #display BV or EC vaccine error
    elif (type == 'BV' or type == 'EC'):
        if (age <18) :
            print('\nThis patient is not suitable for having this dosage.')
            flag = False
        else:
            flag = True

    # display CZ vaccine error
    elif (type == 'CZ'):
        if (age <12 or age >45):
            print('\nThis patient is not suitable for having this dosage.')
            flag = False
        else:
            flag = True

     # error display + back to start
    else:
        flag = False
        print('\nInvalid vaccine code, pls re-submit.\n')
        administration()

    return flag
```

Figure 117: code of approve (type, age) function

This function is used to make sure patients' age **reaches the age requirement** for the dosage that they choose. For the first line code is **converting** the age that we get from the last function **into integer form** because all the data that we get from the text file currently is in the string form. In this form, the data cannot be used to compare with the integer.

For the next come into condition and comparison part. System will compare the patients' age with the age requirement according to the vaccine code. The age requirement for **"AF"** and **"DM"** vaccine is more than **12 years old**, **"BV"** and **"EC"** vaccine is more than **18 years old** and the **"CZ"** is **between 12 years old and 45 years old**. If the patients' age **reaches the requirement**, there is a **flag** variable will be set in **True** as a condition in the vaccine administration function. Oppositely, when patient's age does **not reach requirement**, system will display "This patient is not suitable for having this dosage" and set the **flag become False**.

When user keys in a vaccine code besides “AF”, “DM”, “BV”, “EC” and “CZ”, the flag will be set in False and display “Invalid vaccine code” and back to administration function. In the end of function, the **flag will be returned** to the function.

check_dose1(vac_code, id)

```
'''MAKE SURE DOSE1 SAME WITH DOSE CHOSEN BY PATIENT'''
def check_dose1(vac_code,id):
    vac_code = vac_code+'\n'
    f = open('Patients.txt', 'r')

    # get vaccine code chosen by patient
    for check in f:
        if (check.startswith(id)):
            ID,ic,nm,password,gender,VC,add,pc,age,email,no_ph,date_dose1,date_dose2,vaccine_code = check.split(',')

    # vaccine code key in same with vaccine code chosen by patient
    if (vac_code == vaccine_code) :
        check_dose1_approve = True

    # vaccine code key in different with vaccine code chosen by patient
    else:
        print(f'The vaccine code key in is different with the vaccine code chosen by patient. Patient choose {vaccine_code} vaccine.\nPls re-submit\n')
        check_dose1_approve = False

    return check_dose1_approve
    f.close()
```

Figure 118: code of check_dose1(vac_code, id) function

This function is used to check whether the **vaccine code** key in by user is **same** with the vaccine code that patient registers or not. The function starts with opening the **"Patients.txt"** text file with **reading mode**.

For the next, the system will read all the data line by line in the text file. When the system reads the line starting with the patient id which is keyed in by user, it will split all the data in the line into 14 variables.

In third paragraph, the **vaccine code** that get from the text file **will be compared** with the vaccine code which is keyed in by user. If they are **same**, a new variable named **"check_dose1_approve"** will be set as **True**. If **not**, system will display error to tell user these two vaccine codes are different and show the vaccine code that registered by patient. After that, the **"check_dose1_approve"** will be saved as **False**.

In the last part, the text file will be closed and **return "check_dose1_approve"** variable to the function.

check_2nd_dose_date(id)

```
'''CONFIRM PATIENT REACH SECOND DOSE DATE '''
def check_2nd_dose_date(id):
    f = open('vaccination.txt', 'r')

    # set date of on day
    import datetime
    vac_date = datetime.datetime.today()
    #for testing second dose date arrive
    vac_date = vac_date + datetime.timedelta(days= 35)

    #find second dose date from vaccination.txt
    for check in f:
        if (id in check):
            id,dose,code,date,date1,date2 = check.split(',')
            date =datetime.datetime.strptime(date,'%Y-%m-%d')

            #date reach
            if (vac_date > date):
                flag = True
                break

            #date haven't reach
            else:
                print(f"\nThe second dose date of patient haven't reach, patient cannot have second dose today.\nSecond dose date is {date}")
                flag = False
                break

    return flag
    f.close()
```

Figure 119: code of check_2nd_dose_date(id) function

This function is used to determine whether the patient who wants to have their second dose **reaches the second dose date or not**. The function starts with opening the “vaccination.txt” text file with reading mode. After that, a built-in function, datetime will be imported into this function.

Datetime is a function that is used for setting a date or calculating a date before or after an existing date. For example, the two lines of code after import are used to **set an assumption current date** that 35 days after current date. The assumption current date is set after 35 days because when testing the program for run the second dose function, 35 days can fulfill the time interval requirement of all vaccine.

In the third paragraph, the system will read all the data line by line in the text file. When the system reads the line containing the patient id which is keyed in by user, it will split all the data in the line into 6 variables. The **second dose date** get from text file will be **converted into date** from by using one of the features in datetime function because next step we need to do a comparison between assumption current date and the second dose date. Therefore, these two dates must be in the same data type for doing the comparison.

If the comparison **condition is reached**, the system will set the **flag in True** and **stop the looping**. If **not**, the system will notify the user that this patient hasn't reached the second dose date and show the second dose date. After that, the **flag** will be set as **False**.

Before the function ends, the system will close the text file and **return back the flag** to the function.

check_dose2(dose2_code, id)

```
'''MAKE SURE DOSE1 SAME WITH DOSE2'''
def check_dose2(dose2_code,id):
    f = open('vaccination.txt','r')

    # find first dose code from vaccination.txt
    for check in f:
        if (id in check):
            id,dose,code,date,date1,date2 = check.split(',')

            #dose1 = dose2
            if (dose2_code == code):
                check_dose2_approve = True
                break

            # dose1 = EC
            elif(code == 'EC'):
                print('\nThis patient is having EC vaccine, no need have second dose.')
                check_dose2_approve = None
                break

            #dose1 not= dose2
            else:
                check_dose2_approve = False
                print(f"\nPatient's dose 2 type vaccine is different with dose 1. Patient's dose1 vaccine code is {code}\nPls re-submit\n")
                break

    return check_dose2_approve
    f.close()
```

Figure 120: code of check_dose2(dose2_code, id) function

This function is used to ensure the **vaccine code** of the patient who wants to have the **second dose** is **same** with the vaccine code of first dose. For this function, we also need to get the data from the text file. Therefore, it is still starting with opening “vaccination.txt” in reading mode.

The next step is also similar to the previous function, the system will read all the data line by line in the text file. When the system reads the line containing the patient id which is keyed in by user, it will split all the data in the line into 6 variables. However, the data that need to use in this function is the vaccine code.

The **vaccine code that gets from the text file** will be **compared** with the vaccine code which keyed in by user for a checking. If they are the **same code**, system will set a new variable named **“check_dose2_approve” in True** and **end the looping**. But if the vaccine code is **“EC”**, system will **return a message** to user about this patient get “EC” vaccine, no need having the second dose. After that, the “check_dose2_approve” will be saved as None and the looping will be stopped. For the last situation is these two vaccine codes are **different**, system will show these two vaccine codes is different and what is the vaccine code of first dose of the patient to user and the **“check_dose2_approve”** will be set in **False** continue with end the looping.

In the end, the text file will be closed and the **“check_dose2_approve” is returned** to the function.

dose1_step (id, code)

```
'''DOSE1 STEP'''
def dose1_step(id,code):
    f = open('vaccination.txt','a+')

    #set date of on day
    import datetime
    vac_date = datetime.date.today()

    #write patient data into text file + calculate date of next dose
    #AF vaccine set
    if (code == 'AF'):
        next_dose = vac_date + datetime.timedelta(days = 14)
        f.write(f'{id},dose1,{code},{next_dose},{vac_date},NONE\n')
        print('\nSubmit Successfully')
        print(f'Second dose date: {next_dose}')

    #BV or CZ vaccine set
    elif (code == 'BV' or code == 'CZ'):
        next_dose = vac_date + datetime.timedelta(days=21)
        f.write(f'{id},dose1,{code},{next_dose},{vac_date},NONE\n')
        print('\nSubmit Successfully')
        print(f'Second dose date: {next_dose}')

    #DM vaccine set
    elif (code == 'DM'):
        next_dose = vac_date + datetime.timedelta(days=28)
        f.write(f'{id},dose1,{code},{next_dose},{vac_date},NONE\n')
        print('\nSubmit Successfully')
        print(f'Second dose date: {next_dose}')

    #EC vaccine set
    elif (code == 'EC'):
        f.write(f'{id},dose1,{code},NONE,{vac_date},NONE\n')
        print('\nSubmit Successfully')
        print('patient get EC vaccine no need has second dose.')

    f.close()
```

Figure 121: code of dose1_step (id,code) function

This function is used to **write all the vaccine detail** of a patient who having **first dose** into the "vaccination.txt" text file. First of all, the "vaccination.txt" text file will be opened in append mode. In order to get the current date and calculate the second dose date in the following paragraph, we need to import the datetime function and **save the current date** in "vac_date" variable.

In the following paragraph, system will **calculate the second dose date** through the timedelta feature in the datetime function according to the vaccine code. For "AF" vaccine is 14 days after current date, "BV" and "CZ" vaccines are 21 days and "DM" vaccine is 28 days. The "EC" vaccine is need having one dose only. Thus, there is no second dose date for it.

After calculating the second dose date, **all the vaccine detail** including patient id, number dose that have done, vaccine code, second dose date, date having first dose and date having second dose will be **write into the text file**. Before closing the text file, the second dose date is showed on the screen to user and the “EC” vaccine doesn’t have second dose date. So, it will notify patient no need has second dose.

dose2_step (id, code)

```
'''DOSE2_STEP'''
def dose2_step(id,code):
    #set current date
    import datetime
    vac_date = datetime.date.today()
    vac_date = vac_date + datetime.timedelta(days=35)
    vac_date = vac_date.strftime('%Y-%m-%d')

   #take 1st dose date from vaccination.txt
    f = open('vaccination.txt','r')
    for check in f:
        if (id in check):
            id,dose,code,date,date1,date2 = check.split(',')
    f.close()

    # count the line of patient id
    f = open('vaccination.txt', 'r')
    count = 0
    for line in f:
        count += 1
        if id in line:
            break
    f.close()

    #change the dose status of patient in list
    f = open('vaccination.txt','r')
    lines = f.readlines()
    lines[count-1] = id + ',dose2,'+ code+',NONE,'+date1+','+vac_date+'\n'
    f.close()

    #write back the list that change into text file
    f = open('vaccination.txt', 'w')
    for rewrite in lines:
        f.write(rewrite)
    f.close()

    print('\nSubmit Successfully')
```

Figure 122: code of dose2_step (id, code) function

This function is used to **change the vaccine detail** of patient who having the **second dose** in the “vaccination.txt” text file. At the start of function, system will import the datetime function and **set the assumption current date** which same with the assumption current date in the “check_2nd_dose_date(id)” function. In the next line, the date is converted into string form.

In the next paragraph, the “vaccination.txt” text file will be opened in reading mode and the system will read all the data line by line in the text file. When the system reads the line containing the patient id which is keyed in by user, it will split all the data in the line into 6

variables to **get the date having first dose** for the purpose to rewrite the patient's vaccine detail in the following paragraph.

In order to determine the patient's vaccine detail in which line of the text file, a new variable named "count" will be assigned in zero and the system will start to read the text file line by line again. But after reading each line, the "count" variable will be plus one on its current integer and replace the current integer. When the system reads the line containing the patient id which is keyed in by user, it will stop the looping and the integer in "count" will be the **number of lines of patient's vaccine detail**.

For the last part is changing the patient's existing vaccine detail with a new vaccine detail and writing back them to the text file. First, **all the data** in text file will be **saved in list** form into the "lines" variable by using "readlines" feature and **replace the existing vaccine detail** with the new vaccine detail which contain the assumption current date as the date having second dose according to the number of lines that found in the previous paragraph. After changing the vaccine detail is continue with rewrite all the data back to the text file. Howbeit, the **text file** needs to be closed first and **open again with the writing mode** before rewriting the data. Due to opening the text file in writing mode, all the current data in text file will be removed and replaced with the data that are saved in "lines" variable one by one. The text file will be closed and the system will display "Submit successfully" after all the data in "lines" have been written into the "vaccination.txt" text file.

administration ()

```
#administration(main)
def administration():
    #user input
    patient_id = checking_id()
    if(patient_id == "1"):
        return

    while(True):
        print("\n*Enter '1' to exit")
        dosage_done = input('Pls enter the dosage that patient get today (dose1 / dose2) : ')
        if(dosage_done == "1"):
            return

        if((dosage_done == "dose1") or (dosage_done == "dose2")):
            break
        else:
            print("\n*Error : Invalid dosage number*")

    print("\n*Enter '1' to exit")

    vaccine_details()

    vac_code = input('\nPls enter the vaccine code of patient : ').upper()
    if(vac_code == "1"):
        return


    #read patient's age from registration.txt
    age = checking_age(patient_id)
    approve_status = approve(vac_code, age)
```

Figure 123: code of administration () function

This is the main function of the vaccine administration feature. In the beginning, the **"checking_id"** function will be **run** and **save the return variable** in "patient_id" variable. Continue with the **prompt user** for getting the **dosage** that get on the current day in a looping section. When the user input "dose1" or "dose2" the looping will be stopped. Other than that, the system will display invalid dosage number to user and ask the input again from the user.

The **"vaccine_detail ()"** function in the following line is used to **show all vaccine type details** for user and the following code will ask the input from user for the vaccine code that they register. During all the key in section, user can also key in "1" to exit this feature and back to the admin menu page.

For the last two lines, the **"checking_age(patient_id)"** will be run to **get the patient's age** and save it in the "age" variable. The "age" variable will be used in the **"approve (vac_code, age)"** to **determine** whether the **patient suitable for having the dosage** that they choose or not and the **result** that is returned from the function will be saved in the **"approve_status" variable**.

```
#determine dosage_done
if (approve_status == True):

    #INPUT DOSE2
    if (dosage_done == 'dose2'):
        flag = True
        f = open('vaccination.txt','r')

        #find patient column from vaccination.txt
        for check in f:
            if (check.startswith(patient_id)):
                if ('dose1' in check):

                    #dose1 = EC
                    if(vac_code == 'EC'):
                        print('\nEC vaccine, no need have second dose.')
                        flag = False
                        break
```

Figure 124: code of administration () function cont.

If the **"approve_status" is True**, the following **code will be executed**. If **not**, it will straight away **back to admin menu page**. When user keys in dose 2 for the dosage number, system will set the "flag" as true and open the "vaccination.txt" text file in reading mode.

After opening, system will read all the data in the text file line by line. When reading until the line which starts with patient id and "dose1", system will confirm this patient has done the first dose and collate to the following condition. If user key in "EC" for the vaccine code, user will be notified "EC vaccine no need have second dose" and system will change the "flag" into False.

```
                    #DOSE2 FLOW
                    else:
                        flag = False
                        check_dose2_approve = check_dose2(vac_code, patient_id)

                        #confirm dose1 = dose2
                        if (check_dose2_approve == True):
                            date_approve = check_2nd_dose_date(patient_id)

                            #confirm 2nd dose date reach
                            if (date_approve == True):
                                dose2_step(patient_id,vac_code)
                                break

                        #confirm dose1 not= dose2 + back to start
                        elif (check_dose2_approve == False):
                            administration()
```

Figure 125: code of administration () function cont.

Other than that, system will move into another set of code. First, the flag will also change into False same as the previous condition and the **"check_dose2(vac_code, patient_id)"** function will be run. This function is used to **make sure patient has the same type of vaccine** for the **first dose and second dose** and the result will be saved in the "check_dose2_approve" variable.

If the result is True, system will move to the next step is **running the "check_2nd_dose_date(patient_id)"** function. If False, that's mean user key in a different vaccine code with first dose and system will ask user back to start, key in all the details again.

The purpose of "check_2nd_dose_date(patient_id)" function is **confirming the second dose date of the patient reach** and also save the variable that has been returned into "date_approve" variable. When the "date_approve" is in True, that's mean all of the patient's details are correct and suitable for having the second dose. Then, system will **run** the **"dose2_step(patient_id, vac_code)"** function and renew the patient's vaccine details.

```
            #patient done second dose
            else:
                print('\nThis patient has done second dose!')
                flag = False
                break

#Switch to DOSE1 FLOW
if flag == True:
    print('\nThis patient no having the first dose yet.\nSystem will set patient in dose 1 status.')
    check_dose1_approve = check_dose1(vac_code, patient_id)

    # confirm dose1 = dose chosen by patient
    if (check_dose1_approve == True):
        dose1_step(patient_id,vac_code)

    # dose1 key in different with dose chosen by patient
    else:
        administration()
f.close()
```

Figure 126: code of administration () function cont.

If the system reads the line which starts with patient id, but there is no "dose1" exist in the line. That's mean this patient has done with his second dose. Under this situation, system will tell user about the patient has done the second dose.

In order to make the program more intelligent, there is a feature to switch the patient who does not have the first dose but key in "dose2" in the dosage number to dose 1 status. When all the above conditions haven't been achieved, it means that this patient does not have

his first dose yet. Before writing the patient's vaccine detail into "vaccination.txt" text file, system will **run the "check_dose1(vac_code, patient_id)"** function to ensure the vaccine code keyed in by user is same with the vaccine code registered by patient and the **result** returned back from the function will be **saved in the "check_dose1_approve" variable**.

If the "check_dose1_approve" is **True**, system will **run dose1_step**() which with the feature of writing all the patient's vaccine details into "vaccination.txt". Oppositely, if **False** is saved into "check_dose1_approve", system will **back to the start** and ask user for all the patient's vaccine details again.

Due to the presence of status switch feature in the program, the set of code when user **key in "dose1"** in dosage number is **quite same** with the set of code of the **"dose2" condition** above. The difference between these two sets of code is the notice to tell user about changing patient into "dose1 status" will replace with "dose 2 status" and the position will be changed to before running the "check_dose2(vac_code, patient_id)" function.

When this function is finishing run, the "admin_menu()" function will switch the page back to the admin menu page.

```
    #INPUT DOSE1
    elif (dosage_done == 'dose1'):
        flag = True
        f = open('vaccination.txt', 'r')

        # find patient column from vaccination.txt
        for check in f:
            if (check.startswith(patient_id)):
                if ('dose1' in check):

                    # dose1 = EC
                    if (vac_code == 'EC'):
                        print('\nThis patient is having EC vaccine, no need have second dose.')
                        flag = False
                        break

                    # Switch to DOSE2 FLOW
                    print('\nThis patient has done first dose.\nSystem will set patient in dose2 status')
                    flag = False
                    check_dose_approve = check_dose2(vac_code, patient_id)

                    # confirm dose1 = dose2
                    if check_dose_approve == True:
                        date_approve = check_2nd_dose_date(patient_id)

                        # confirm 2nd dose date reach
                        if (date_approve == True):
                            dose2_step(patient_id, vac_code)
                            break

                    # confirm dose1 not= dose2 + back to start
                    elif (check_dose_approve == False):
                        administration()

        #DOSE1 FLOW
        if flag == True:
            check_dose1_approve = check_dose1(vac_code, patient_id)

            # confirm dose1 key in = dose chosen by patient
            if (check_dose1_approve == True):
                dose1_step(patient_id, vac_code)

            #dose1 key in different with dose chosen by patient
            else:
                administration()
        f.close()

#back to menu
admin_menu()
```

Figure 127: code of administration () function cont.

```python
#############################################################################################################################
#DONE BY GAN MING HUI
#Let admin add new vaccine

def add_vaccine():
    flag = False
    while(flag == False):

        print("\n--------------------Add New Vaccines Code--------------------")

        #Ask new vaccine details
        #vaccine code
        print("\n*Enter 'exit' to exit")
        vc_code = input("Please enter the new vaccine code : ").upper()
        if(vc_code == "EXIT"):
            return


        #Validation for VC Code - Make sure new vaccine no exist in the text file
        f = open("vaccine.txt", "r")
        lists = []

        for line in f:
            line_split = line.split(",")
            lists.append(line_split)

        for i in range(len(lists)):
            if (vc_code) in lists[i][0]:
                print("*vaccine code exists, Please try again")
                add_vaccine()
                return #to stop asking again the dosage required after executing the add_vaccine
        f.close()

        #validation for dosage required
        dosage_required = validation_dosage()
        if(dosage_required == "exit"):
            return

        #validation for interval
        interval = validation_interval()
        if(interval == "exit"):
            return

        #validation for age_group
        age_group = validation_age()
        min_num = (age_group[:2])
        max_num = (age_group[3:])
        if(age_group == "exit"):
            return

```

Figure 128: code of administration () function cont.

```
120         #Append new vaccine code into text file
121         f = open("vaccine.txt","a")
122         f.writelines([vc_code,",",dosage_required," dose required",",",interval," week interval",",",min_num,",",max_num,"\n"])
123         f.close()
124
125         while(True):
126             print("\n*Record added")
127             opt = input("\nDo you still want to add new vaccine (yes/no) : ").lower()
128
129             if(opt == "yes"):
130                 break
131
132             elif(opt == "no"):
133                 flag = True
134                 break
135
136             else:
137                 print("Invalid Input")
138
```

Figure 129: add_vaccine()

Explanation for add_vaccine():

The **main purpose** of this function is let admin to add the new vaccine that does not exist in the vaccine list. For submitting a new vaccine into the current vaccine list, admin need to fill out all the vaccine details about the vaccine including dosage required, time interval between two dosage and age group.

Line 75&76: The variable ‘flag’ is set to False for looping purpose.

Lines 78-85: Ask input for the new vaccine code

- Line 82&83: Ask the new vaccine code and convert it into capital letter
- Line 84&85: Back to admin menu page if “EXIT” is keyed in.

Lines 89-101: Validation for VC Code

- Line 89: Open “vaccine.txt” in reading mode
- Line 90: Generate a new empty list named “lists”
- Line 92: Using for loop to loop every record in the ‘Patients.txt’ text file.
- Line 93: Positioning every data of line that splitting by “,” and save in “line_split”
- Line 94: Appending the ‘line_split’ list into the ‘lists’ list. Therefore, the ‘lists’ will become a nested list.
- Line 96&97: Checking whether the vaccine code is existed in the “vaccine.txt”
- Line 98&99: vaccine code key-in is existed in the “vaccine.txt” and run the add_vaccine() function again
- Line 100: stop looping
- Line 101: Close “vaccine.txt”

Lines 104-118: Ask for the vaccine details

- Line 104-106: Ask for dosage required and enter “exit” to back to admin menu page
- Line 109-111: Ask for time interval between two dosage and enter “exit” to back to admin menu page
- Line 114-118: Ask for age group and enter “exit” to back to admin menu page

Lines 121-123: Append new vaccine code into text file

- Line 121: Open “vaccine.txt” in append mode
- Line 122: Write all the vaccine details into “vaccine.txt”
- Line 123: Close “vaccine.txt”

Lines 125-137: Ask user whether need add another new vaccine or not

- Line 125: Start a looping
- Line 126&127: Ask from user whether continue add another new vaccine or not
- Line 129&130: Stop looping after user key in “yes”
- Line 132-134: Change “flag” into True and stop looping
- Line 136&137: Tell user about the invalid input

```python
def delete_vaccine():
    flag = True
    while(flag == True):
        print("\n--------------------Delete Vaccine--------------------")

        #Ask the vaccine code that admin want to delete
        print("\n*Enter '1' to exit")
        vc_code = input("Please enter the vaccine code you want to delete : ").upper()
        if(vc_code == "1"):
            return

        #Make sure vaccine exist in the text file
        f = open("vaccine.txt", "r")
        lists = []

        for line in f:
            line_split = line.split(",")
            lists.append(line_split)

        for i in range(len(lists)):
            if (vc_code) in lists[i][0]:
                ans = "yes"
                break

            else:
                ans = "no"
        f.close()

        if(ans == "yes"):
            f = open("vaccine.txt","r")
            text = f.readlines()
            del text[i]
            f.close()

            #insert updated record to the text file
            f = open("vaccine.txt","w")

            #loop each record in the list
            for a in text:
                f.write(a)
            f.close()

            print("\n*Deleted Successfully")

        else:
            print("*vaccine code not exists, Please try again")
            delete_vaccine()
            return #to stop asking again the below statement

        #want to continue delete or not
        while(True):
                opt = input("\nDo you still want to delete vaccine (yes/no) : ").lower()

                if(opt == "yes"):
                    break

                elif(opt == "no"):
                    flag = False
                    break

                else:
                    print("Invalid Input")

```

Figure 130: delete_vaccine()

Explanation for delete_vaccine():

The **main purpose** of this function is let admin to delete the current vaccine that exist in the vaccine list.

Line 259&260: The variable 'flag' is set to True for looping purpose.

Lines 264-267: Ask input for the new vaccine code

- Line 264&265: Ask the vaccine code that want to delete and convert it into capital letter
- Line 266&267: Back to admin menu page if "1" is keyed in.

Lines 270-284: Make sure vaccine exist in the text file

- Line 270: Open "vaccine.txt" in reading mode
- Line 271: Generate a new empty list named "lists"
- Line 273: Using for loop to loop every record in the 'Patients.txt' text file.
- Line 274: Positioning every data of line that splitting by "," and save in "line_split"
- Line 275: Appending the 'line_split' list into the 'lists' list. Therefore, the 'lists' will become a nested list.
- Line 277&178: Checking whether the vaccine code is existed in the "vaccine.txt"
- Line 278: vaccine code key-in is existed in the "vaccine.txt" and set a new variable "ans" in "yes"
- Line 279: vaccine code key-in is not existed in the "vaccine.txt" and set a new variable "ans" in "no"
- Line 100: stop looping
- Line 101: Close "vaccine.txt"

Lines 286-305: Delete the vaccine details and update the text file record

- Line 286&287: If the "ans" is "yes", the "vaccine.txt" will be opened in reading mode
- Line 288: Read all the data from "vaccine.txt" and save in "text" variable in list form
- Line 289: Delete the set of data according to the vaccine code keyed in by admin
- Line 290: Close "vaccine.txt"
- Line 293: Open "vaccine.txt" in writing mode
- Line 296&297: Write back all the data in "text" into "vaccine.txt"
- Line 298&300: Close "vaccine.txt" and display "*Deleted Successfully"

- Line 302-304: If the “ans” not “yes”, notify admin about the vaccine code is not exist and run the delete_vaccine() function again.
- Line 305: Stop the delete_vaccine() function

Lines 308-319: Ask user whether need delete another vaccine or not

- Line 308: Start a looping
- Line 309: Ask from user whether continue delete another vaccine or not
- Line 311&312: Stop looping after user key in “yes”
- Line 314-316: Change “flag” into True and stop looping
- Line 318&319: Tell user about the invalid input

5.0 Screenshots of Sample Input / Output & Explanation

5.1 - GAN MING HUI

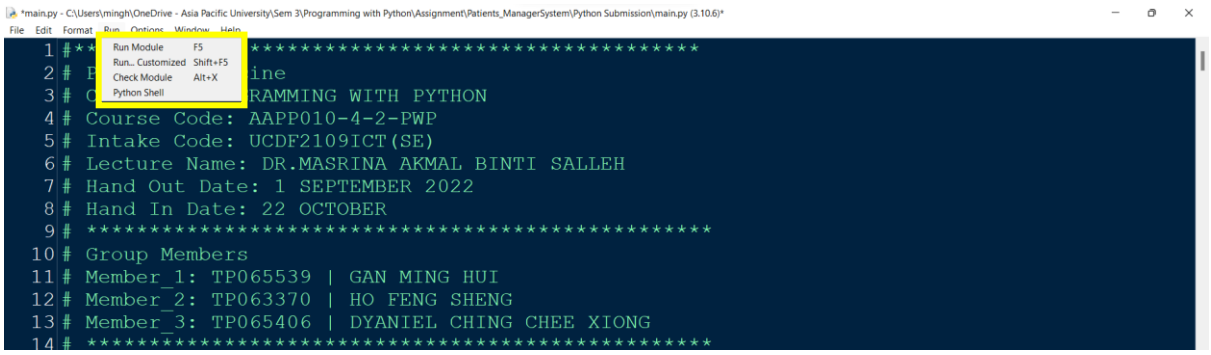


Figure 131: Running Program

To run the vaccine system program, users need to open the 'main.py' file first. Once it is open, users can run it by clicking on 'Run' and then 'Run module' as shown in figure 47 or by pressing Ctrl + F5 to run the program.

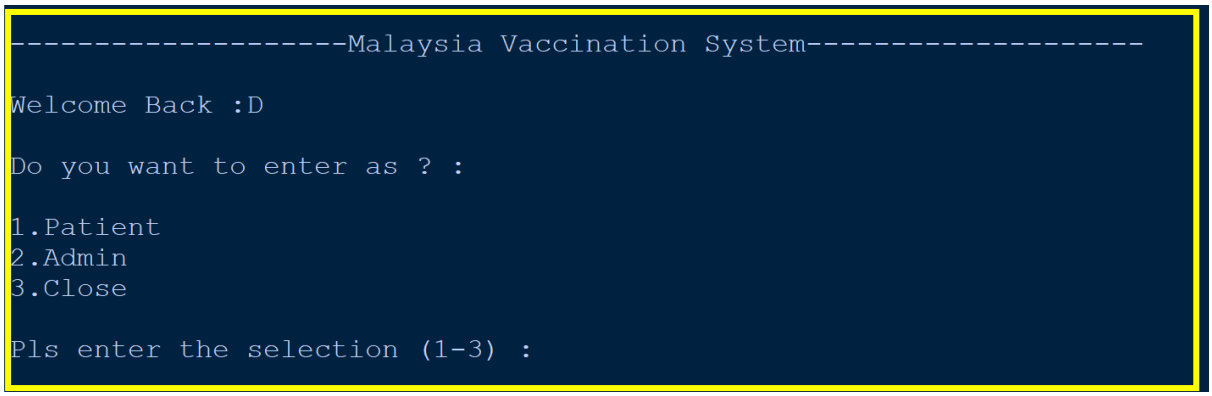


Figure 132: Main Menu for Malaysia Vaccination System

When users run the program, the main menu with the title 'Malaysia Vaccination System' will appear as shown in figure 48. The program will also ask the users in what identity they want to access the system, either 'Patient' or 'Admin'. The user can also enter '3' to close the program.

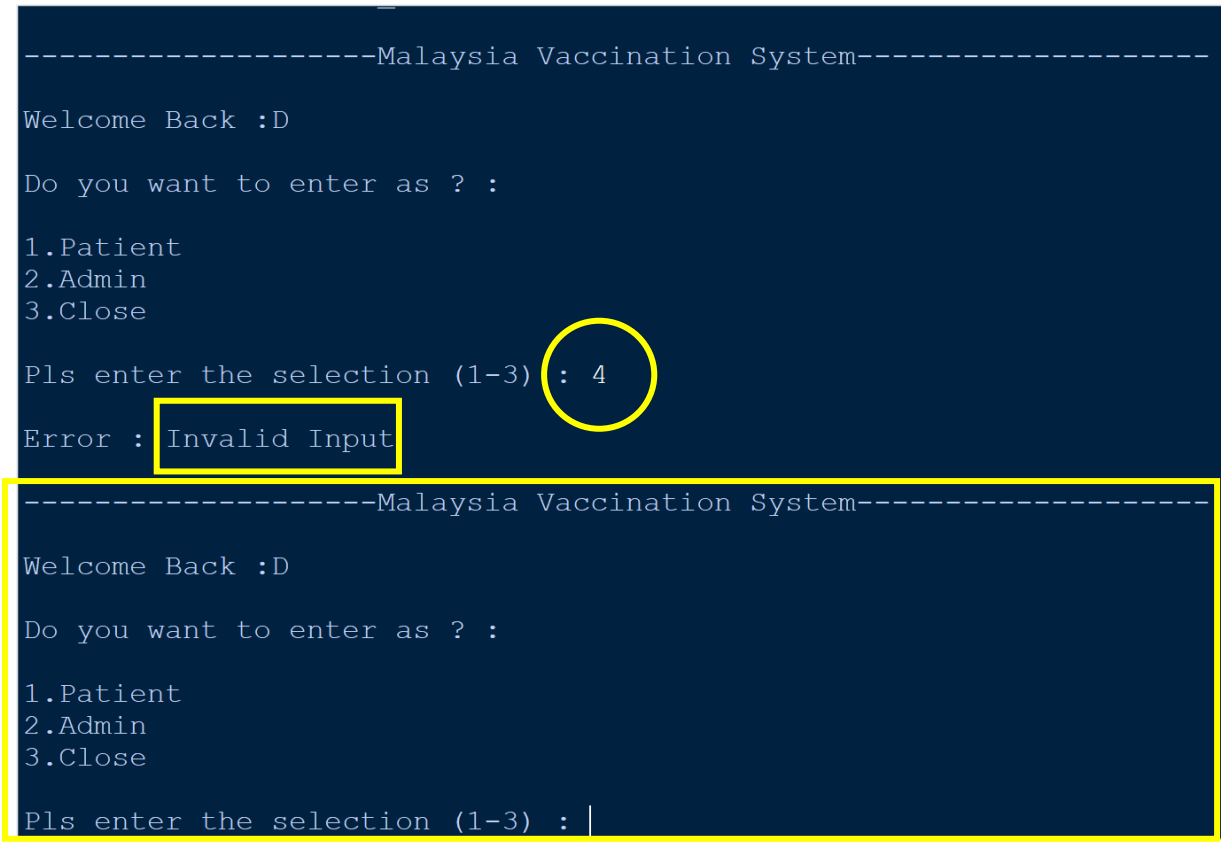


Figure 133: Invalid Input in Main Menu for Malaysia Vaccination System

If the users enter a selection other than 1-3 in the main menu of the 'Malaysia Vaccination System', as in figure 49 above, the 'Invalid Input' will be printed. Also, the users will be asked again what identity they want to access the system.

```
--------------------Malaysia Vaccination System--------------------

Welcome Back :D

Do you want to enter as ? :

1.Patient
2.Admin
3.Close

Pls enter the selection (1-3) : 1

--------------------Patient Account--------------------

1.Create New Account
2.Login into Existing Account
3.Back

Pls enter the selection :
```

Figure 134: Patient Account [Input == '1']

If the user selects '1', they will be taken to the 'Patient Account' page as shown in figure 50. On this side, the users have three options to choose from, '1.Create New Account', '2.Login into Existing Account' and '3.Back'.

```
--------------------Patient Account--------------------

1.Create New Account
2.Login into Existing Account
3.Back

Pls enter the selection : 4

*Error : Invalid Input

--------------------Patient Account--------------------

1.Create New Account
2.Login into Existing Account
3.Back

Pls enter the selection :
```

Figure 135: Invalid Input in Patient Account

If the users enter a selection other than 1-3 in the menu of the 'Patient Account', as in figure 51 above, the 'Invalid Input' will be printed. Also, the users will be asked one more time do they want to sign up for a new account, log in into an existing account or back to the previous page.

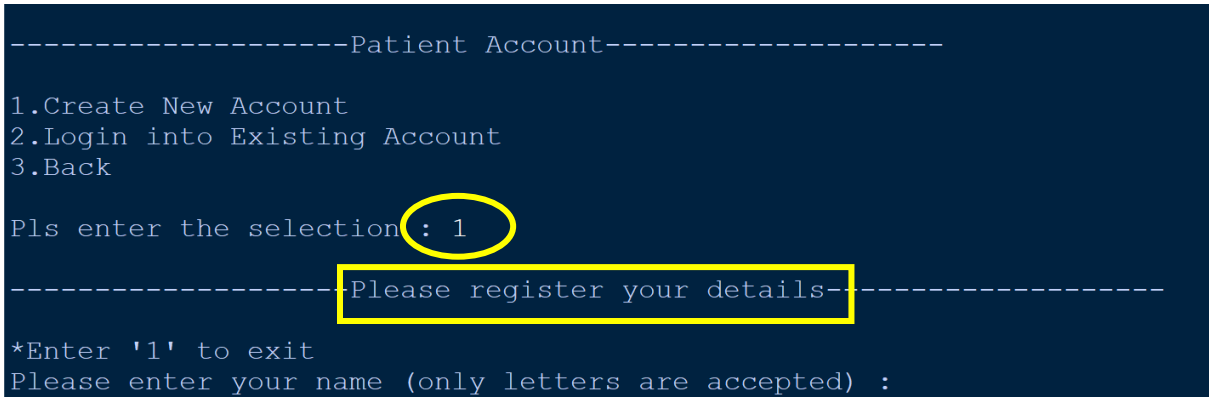


Figure 136: Create New Account [Input == '1']

If the user enters '1' it is 'Create New Account'. They will start registering their account. First, they will be asked to enter their name as shown in figure 52.

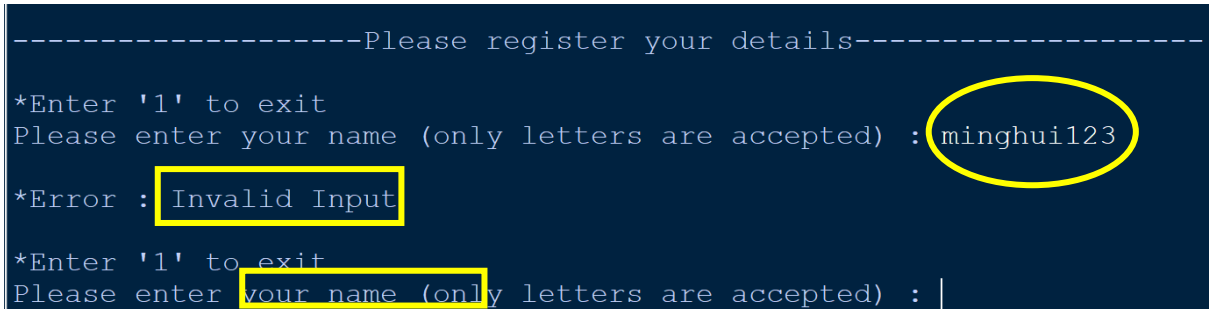


Figure 137: Invalid Input in Entering the Name [Input == "minghui123']

For the name users can only enter letters, no numbers or symbols are allowed. If the user enters characters other than letters, 'Invalid Input' will be printed, and they will then need to fill in their name again as shown in figure 53.

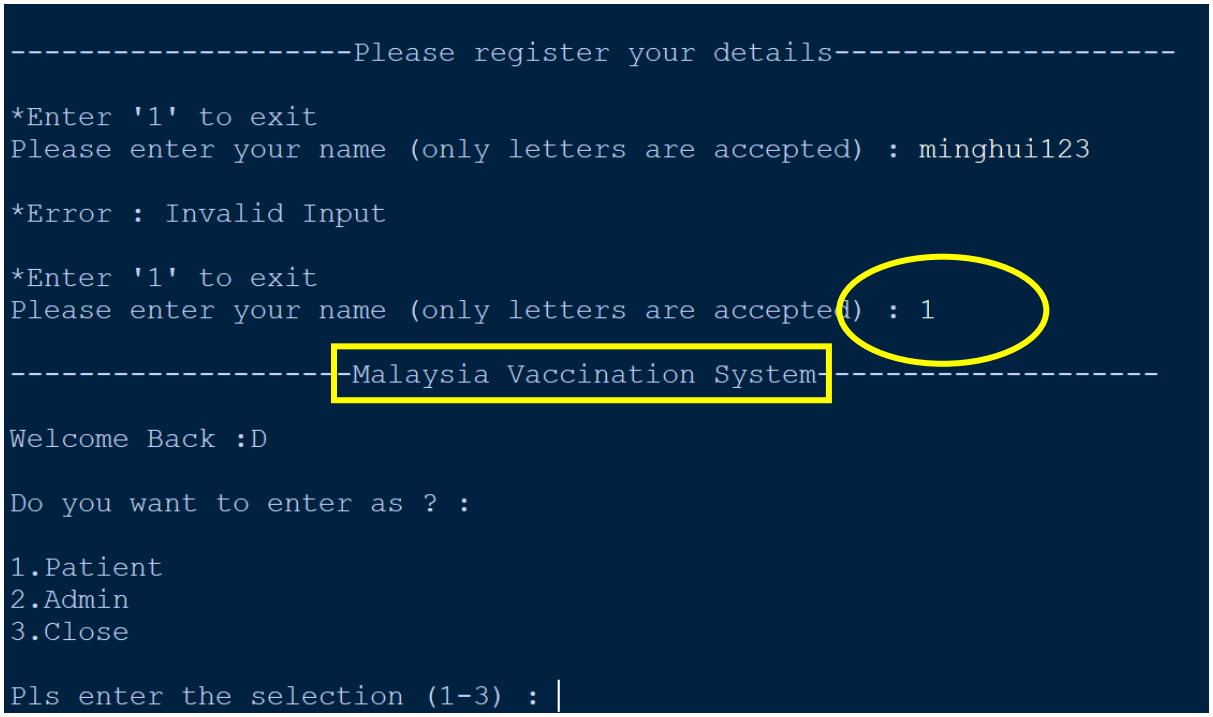


Figure 138: Exit from the Registration [Input == '1']

If the user accidentally clicks to register a new account, then they can exit the registration by entering '1' and they will be taken back to the system's main menu as shown in figure 54.

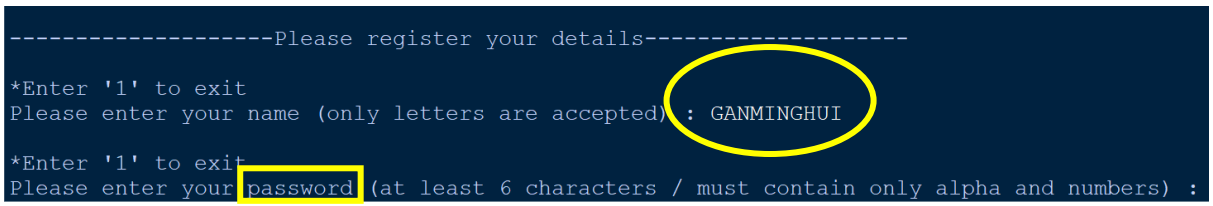


Figure 139: Correct name [Input == 'GANMINGHUI'], then move to the password

After entering the name as shown in figure 55, users have to enter their password for the new account. For passwords, the password must be at least six characters and must have both letters and numbers.

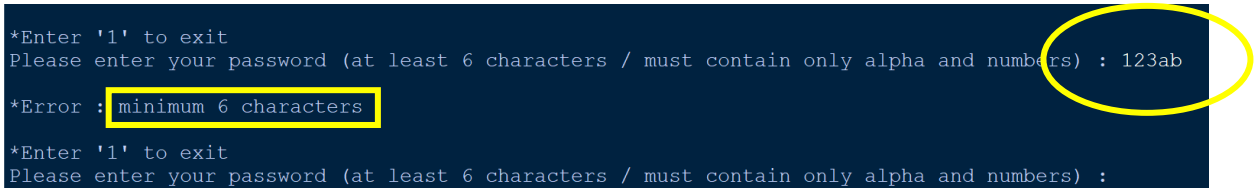


Figure 140: Invalid Password - No at least six characters [Input == '123ab']

As shown in figure 56, although the users have met the condition that they must have only letters and numbers, they have not met the condition that they must have at least six characters.

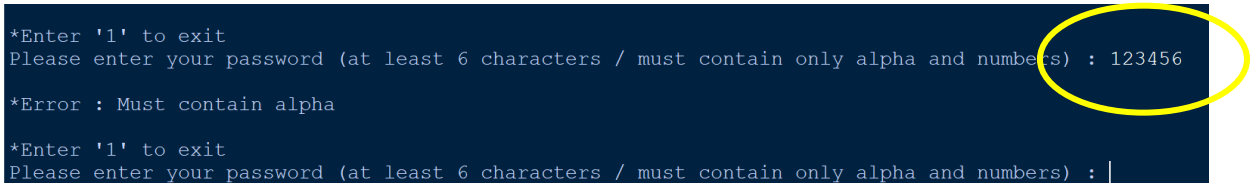


Figure 141: Invalid Password - No alpha [Input = '123456']

As shown in figure 57, this time the users have only met the condition that they must have at least six characters, but not yet meet the condition that they must have letters and numbers because the users have only numbers.

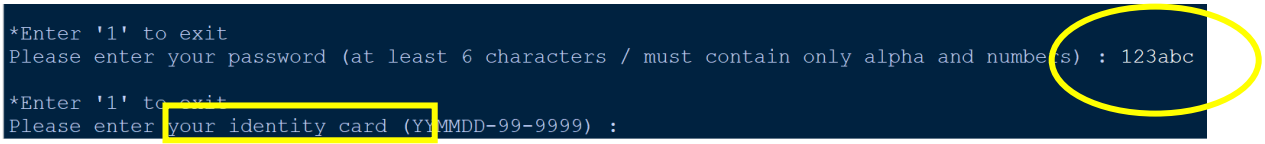


Figure 142: Correct password [Input == '123abc'], then move to the IC

After entering the password as shown in figure 58, the users need to enter their IC. The first two digits of the IC represent the patient's year of birth and can be any number from 0-9. The third and fourth digits of the IC represent the patient's month of birth. The fifth and sixth digits of the IC represent the patient's date of birth. After entering the six numbers, the user must enter - first and then enter the seventh and eighth numbers. The seventh and eighth numbers can be one of the numbers 0-9. After entering the eighth number, the user must first enter - and then enter the remaining four numbers. In addition, the IC entered must only contain numbers and only 14 characters.

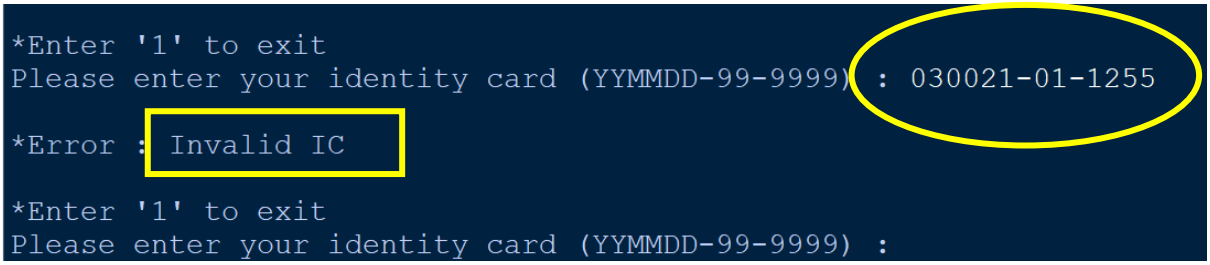


Figure 143: Invalid IC – Wrong Month [Input == '030021-01-1255']

As shown in figure 59, the IC entered by the user is wrong because the month cannot be '00'. The month can only be from January to December.

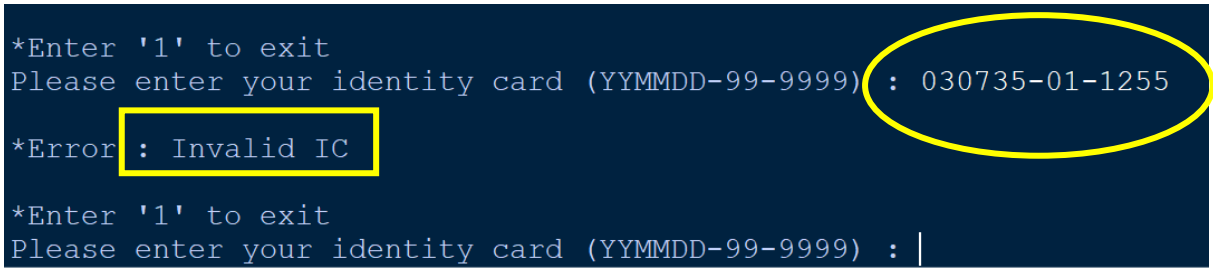


Figure 144: Invalid IC – Wrong Date [Input == '030735-01-1255']

As shown in figure 60, the IC entered by the user is wrong because the date cannot be '35'. The date can only be from the first to the 30th or 31st.

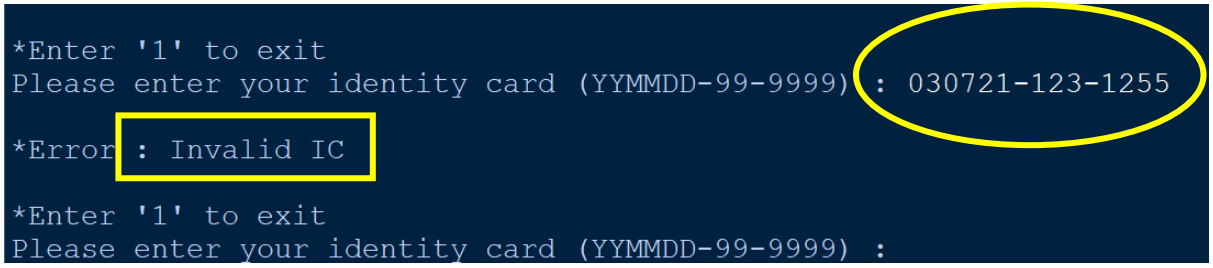


Figure 145: Invalid IC - Wrong Place of Birth [Input == '030721-123-1255']

As shown in figure 61, the IC entered by the user is wrong because the place of birth cannot exceed two numbers.

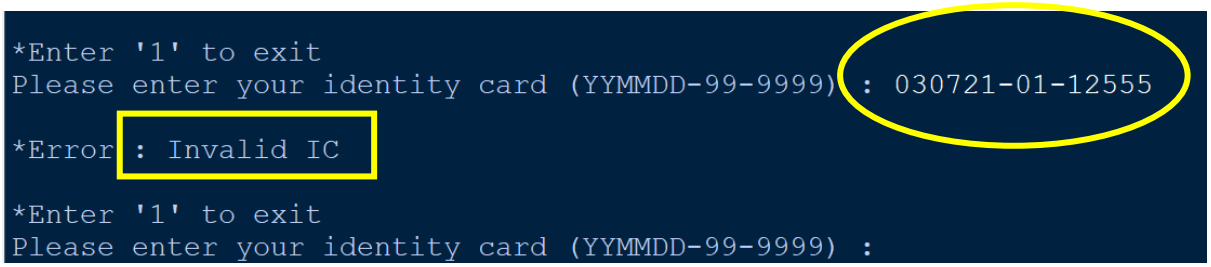


Figure 146: Invalid IC - Last Four Numbers are Wrong [Input == '030721-01-12555']

As shown in figure 62, the IC entered by the user is wrong because the last part of the IC can only be four numbers.

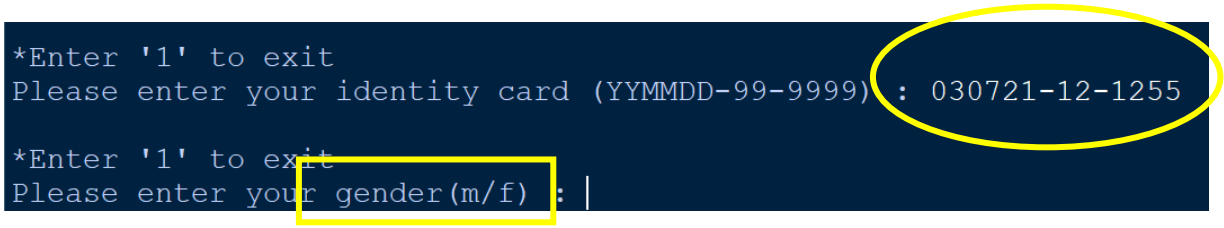


Figure 147: Correct IC [Input == '030721-12-1255'], then move to the gender

After entering the IC as shown in figure 63, the users need to enter their gender as either m(male) or f(female).

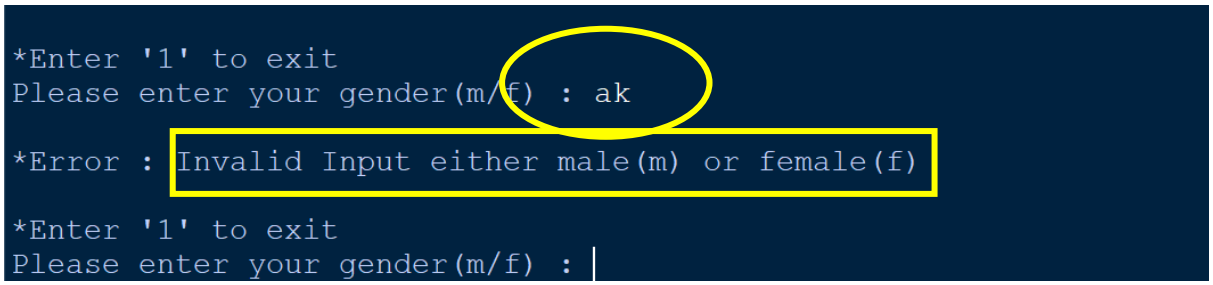


Figure 148: Invalid Gender [Input == 'as']

As shown in figure 64, the gender entered by the user is wrong because the gender can be either 'm' or 'f'.

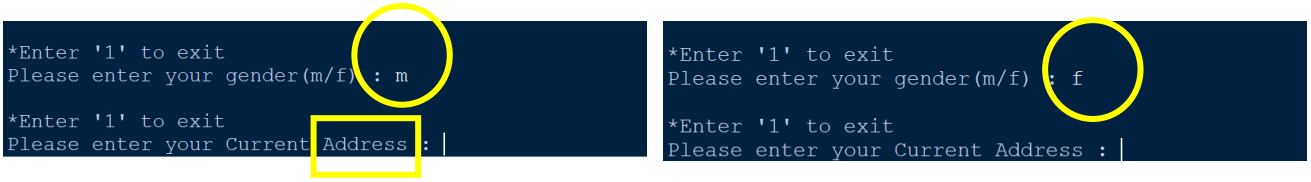


Figure 149: Correct gender [Input == 'm' or 'f'], then move to the address

After entering their gender as shown in figure 65, the users need to enter their current address.

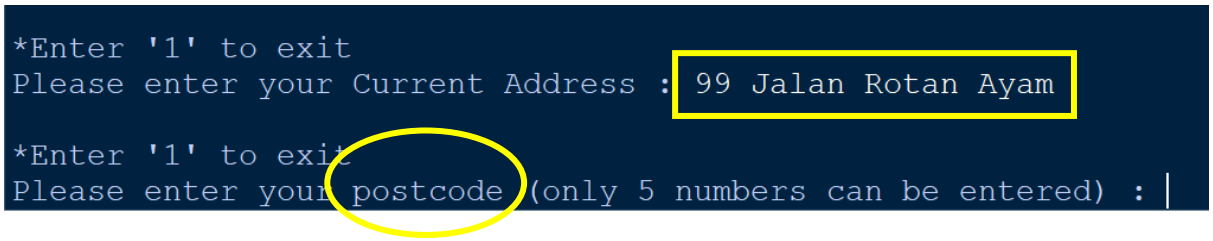


Figure 150: Address [Input= '99 Jalan Rotan Ayam'], then move to the postcode

After entering their address as shown in figure 66, the users need to enter their postcode. For postcode, only five numbers can be entered. Any less than five or more than five will not be accepted.

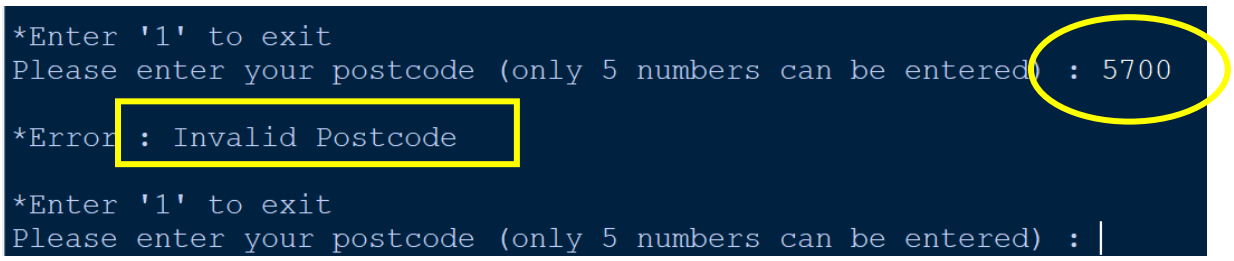


Figure 151: Invalid Input - Only Four Numbers [Input == '5700']

As shown in figure 67, the postcode entered by the user is wrong because the user only entered four numbers and the postcode can only be five numbers.

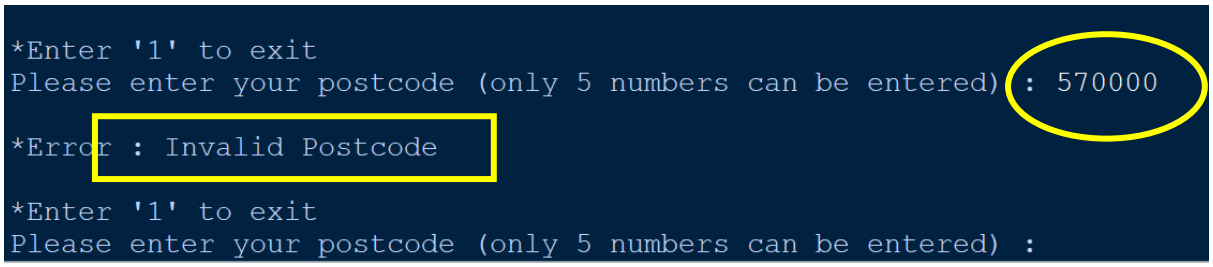


Figure 152: Invalid Input - Cannot More than 5 Numbers [Input == '570000']

As shown in figure 68, the postcode entered by the user is wrong because the user has entered more than five numbers and the postcode must only be five numbers.

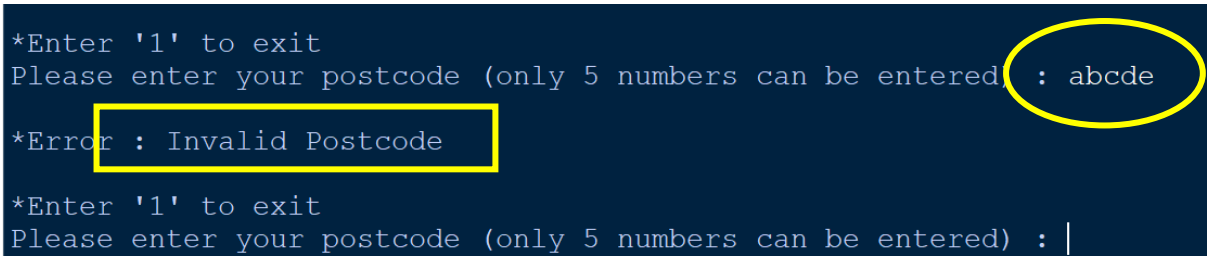


Figure 153: Invalid Input - Postcode must be numbers [Input == 'abcde']

As shown in figure 69, the postcode entered by the user is wrong because the user entered letters and the postcode can only be five number.

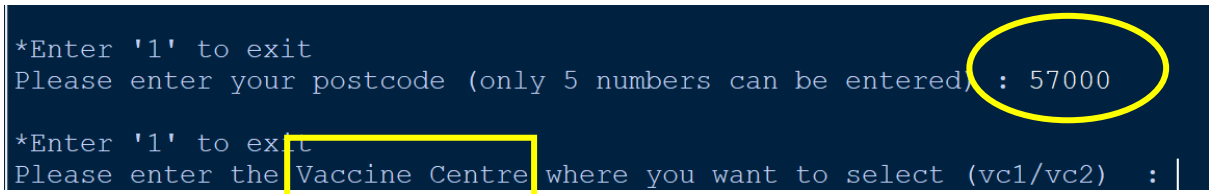


Figure 154: Correct Input - [Input == '57000'], then move to the VC

After entering the postcode as shown in figure 70, the users need to enter which vaccine centre (VC) they want to receive the vaccine at. The user can only enter VC1 or VC2, the user can enter the VC in upper or lower case because eventually it will be automatically converted to upper case.

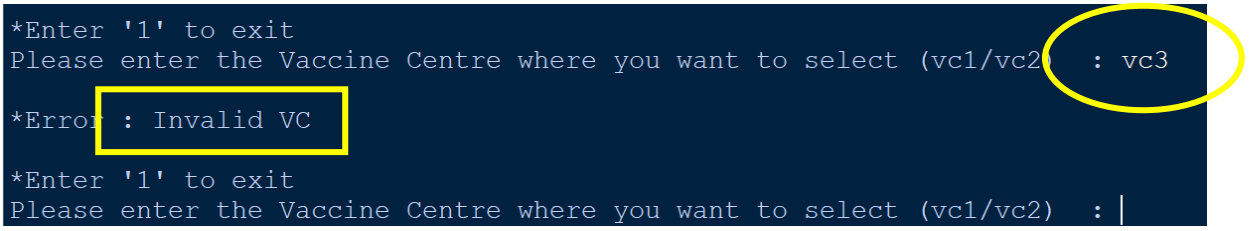


Figure 155: Invalid Input - [Input == 'vc3']

As shown in figure 71, the VC entered by the user is invalid because the only vaccine centers that can be selected are VC1 or VC2.

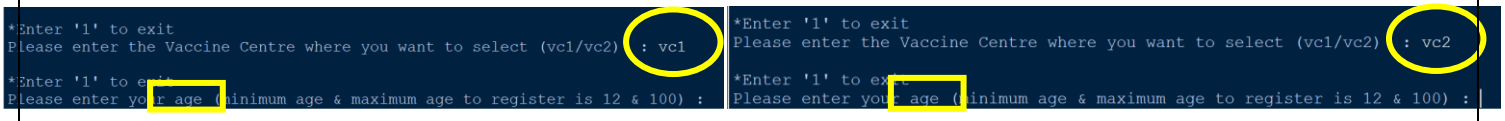


Figure 156: Correct Input, then move to the age

[Input == 'vc1' or 'vc2'] [Output == 'Age']

After users select the vaccine centre as shown in figure 72, the users need to enter their age. For age, the vaccine system only allows users between the ages of 12 and 100 to register.

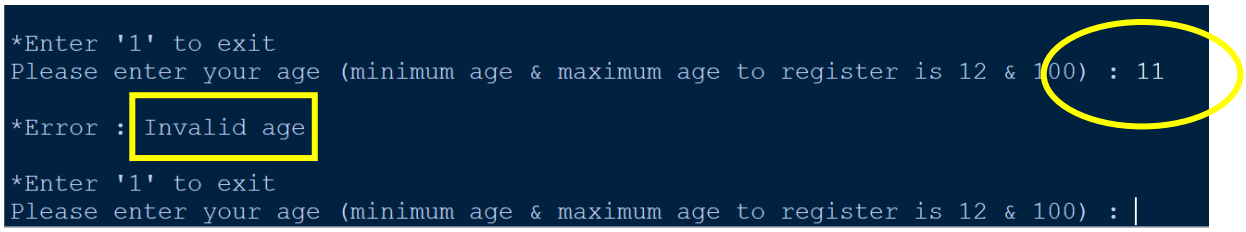


Figure 157: Invalid Input - [Input == '11']

As shown in figure 73, the age entered by the user is invalid because the minimum age that can be allowed to register is 12.

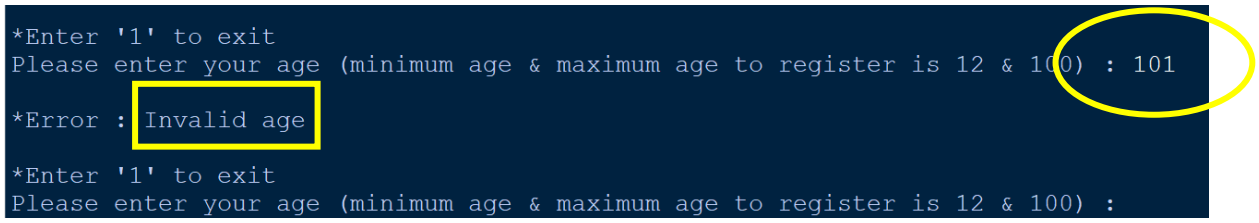


Figure 158: Invalid Input - [Input == '101')

As shown in figure 74, the age entered by the user is invalid because the maximum age that can be allowed to register is 100

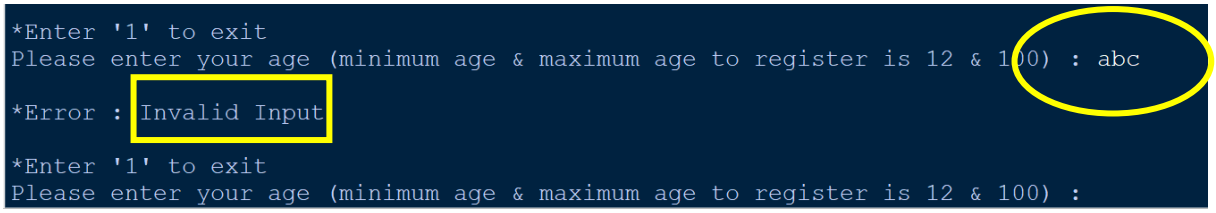


Figure 159: Invalid Input - [Input == 'abc']

As shown in figure 75, the age entered by the user is invalid because the age must be numbers.

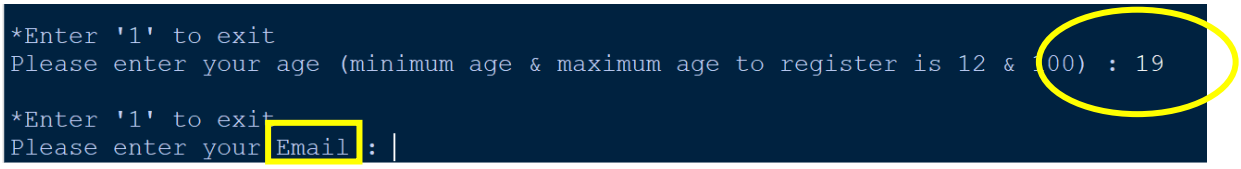


Figure 160: Correct Input - [Input == '19'], then move to the email

After users enter their age as shown in figure 76, the users need to enter their email. For email, the user must have a username before the @, and then a mail server after the @ and a domain after the mail server.

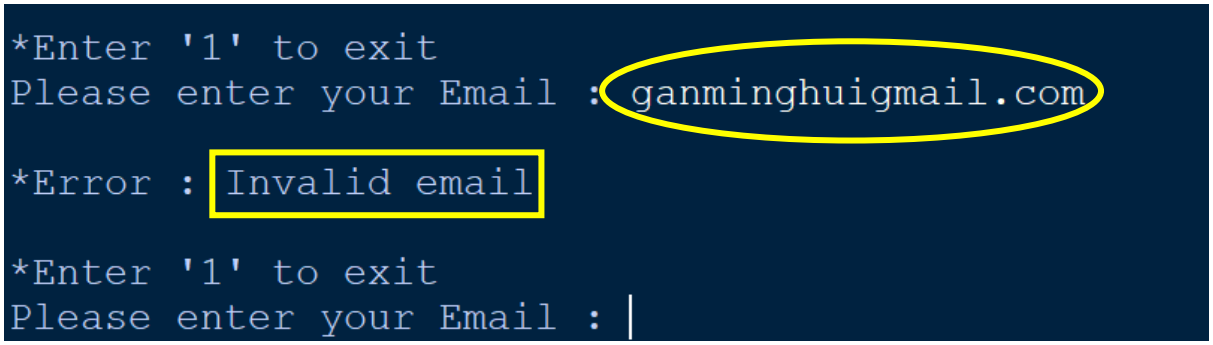


Figure 161: Invalid Input - Missing '@' [Input == 'ganminghuigmail.com']

As shown in figure 77, the email entered by the user is invalid because the email must have '@' between the username and the mail server.

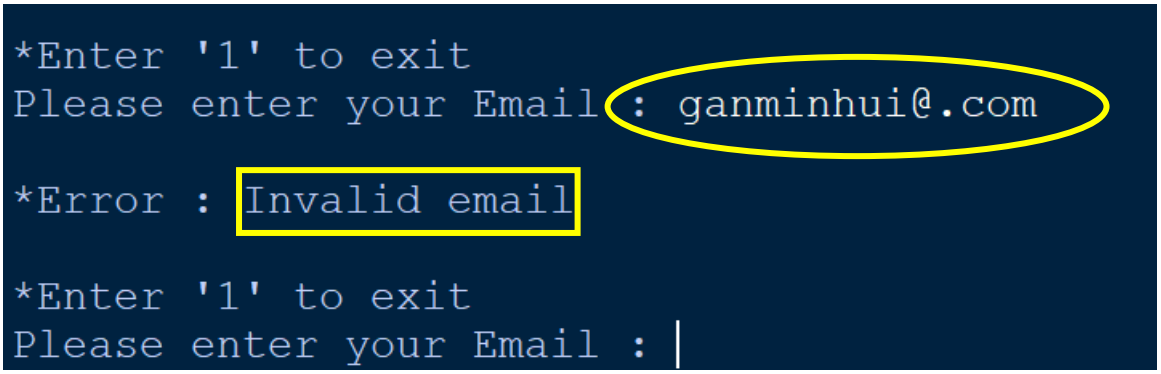


Figure 162: Invalid Input - Missing mail server [Input == 'ganminhui@.com']

As shown in figure 78, the email entered by the user is invalid because the email must have mail server between the '@' symbol and the domain.

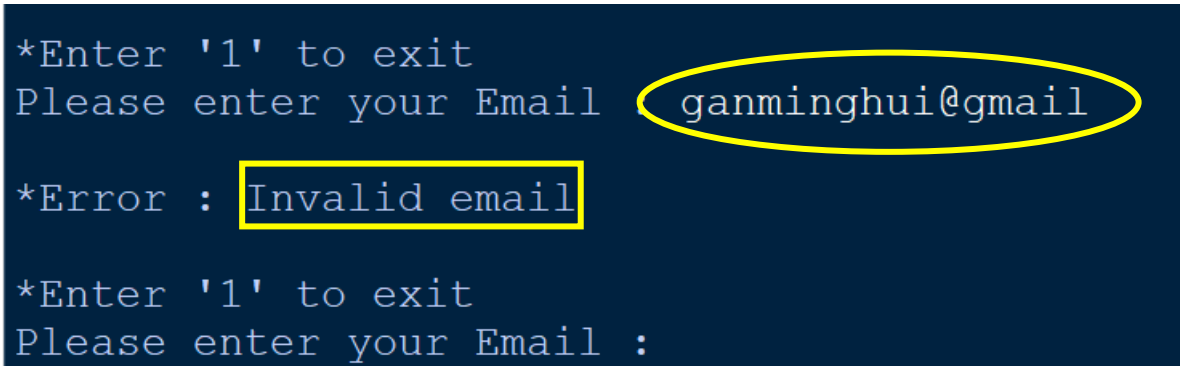


Figure 163: Invalid Input - Missing domain [Input == 'ganminghui@gmail']

As shown in figure 79, the email entered by the user is invalid because the email must have domain in the end of the email.

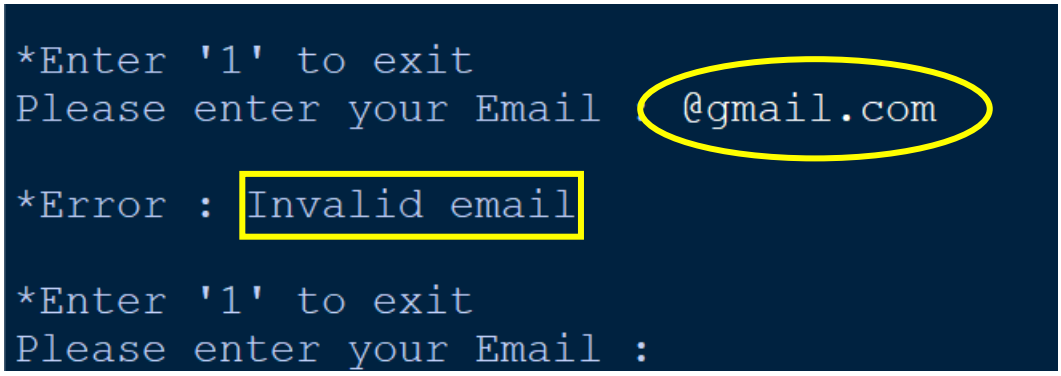


Figure 164: Invalid Input - Missing Username [Input == '@gmail.com']

As shown in figure 80, the email entered by the user is invalid because the email must have username before the '@' symbol.

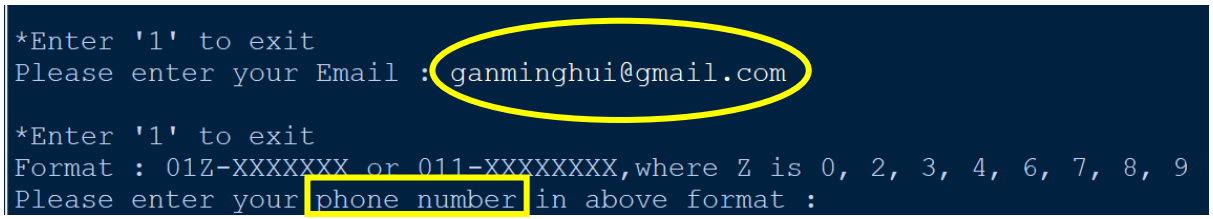


Figure 165: Correct email [Input == 'ganminghui@gmail.com'], then move to the phone number

After entering the email as shown in figure 81, users have to enter their phone number. For the phone number, the first two digits of the phone number pattern must be 01. The third digit can be 0-9 except for 5. This is because 015 is not a valid phone number. After entering the three digits, the user must enter '-' before the remaining digits can be entered. If the third digit of the phone number is 1 then there must be 8 more digits to enter but if the third digit of the phone number is other than 1 and 5 then there must be 7 more digits to enter.

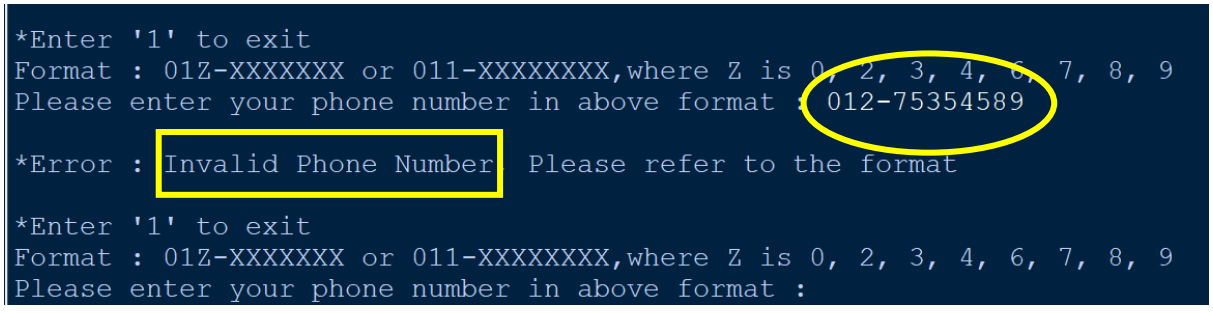


Figure 166: Invalid Input - Wrong Format [Input == '012-75354589']

As shown in figure 82, the phone number entered by the user is invalid because the phone number starts with 012 and must be followed by seven numbers.

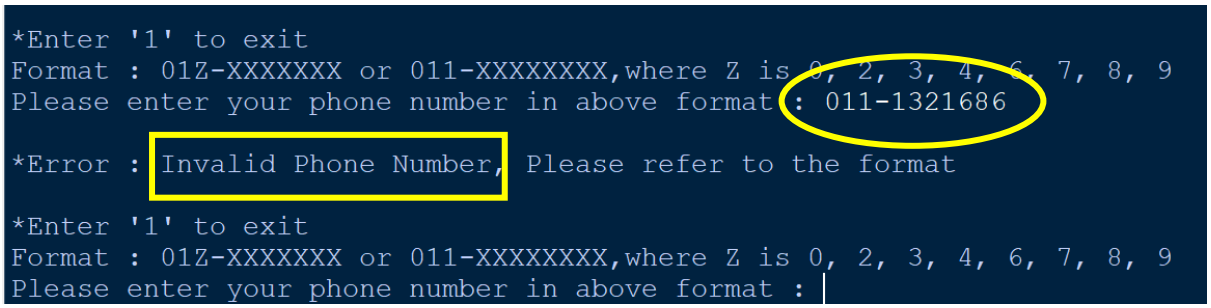


Figure 167: Invalid Input - Wrong Format [Input == '011-1321686']

As shown in figure 83, the phone number entered by the user is invalid because the phone number starts with '011' and must be followed by eight numbers.

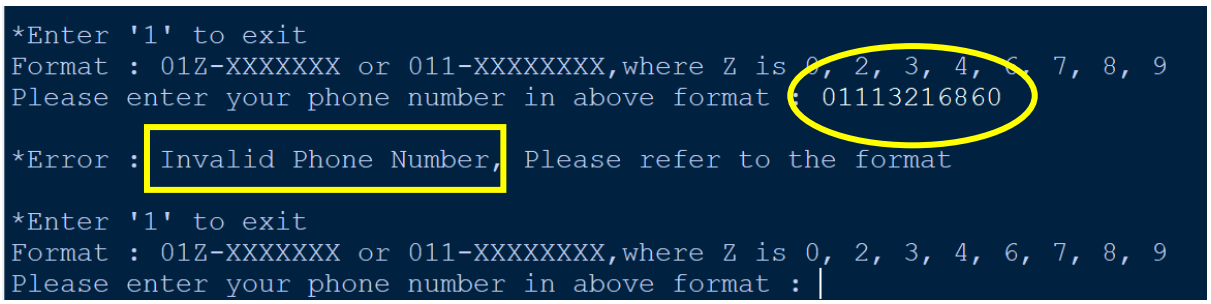


Figure 168: Invalid Input - Wrong Format [Input == '01113216860']

As shown in figure 84, the phone number entered by the user is invalid because the user did not enter '-' after entering three numbers.

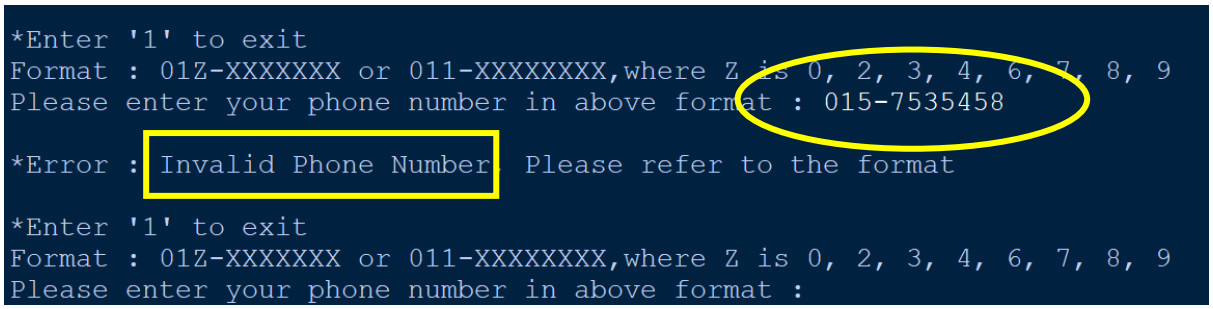


Figure 169: Invalid Input - 015 is not allowed [Input == '015-7535458']

As shown in figure 85, the phone number entered by the user is invalid because '015' phone number is not allowed.

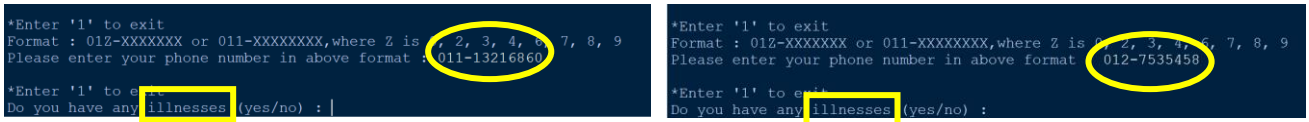


Figure 170: Correct Phone Number [Input == '011-13216860' or '012-7535458'], then move to the illness

After entering the phone number as shown in figure 86, the users need to enter do they have an illness. The users can only enter 'yes' or 'no'.

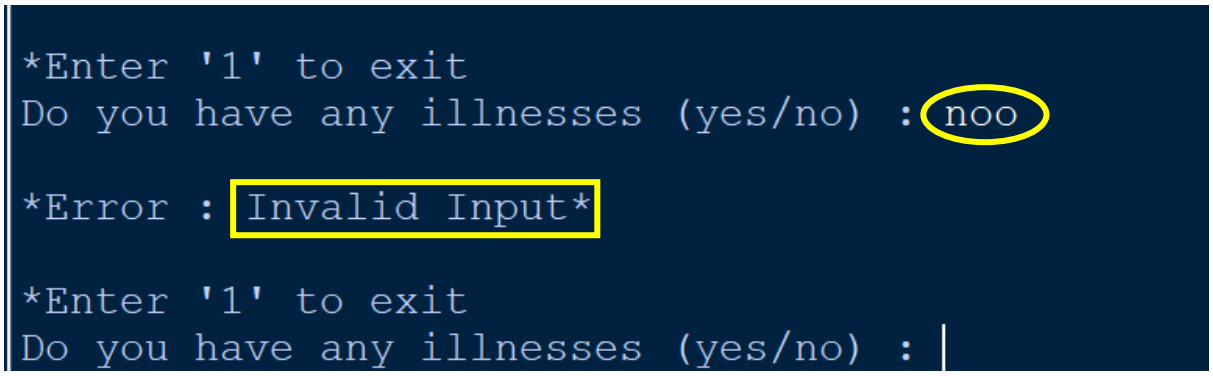


Figure 171: Invalid Input - (yes/no) [Input == 'noo']

As shown in figure 87, the input entered by the user is invalid because the users can only enter 'yes' or 'no'.

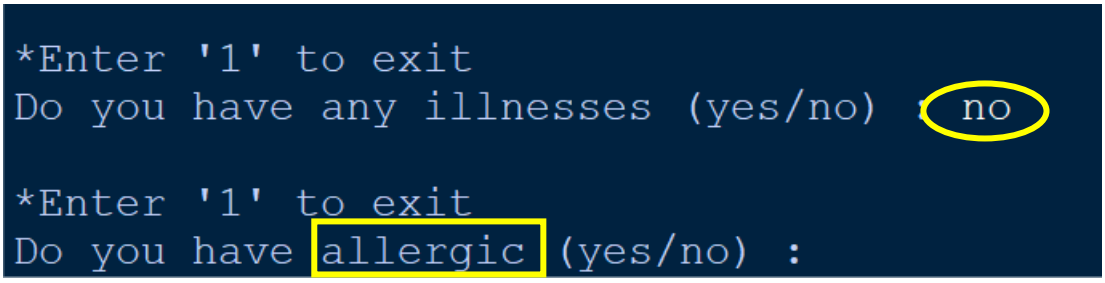


Figure 172: Correct Input [Input == 'no'], then move to the allergic

After entering whether they have an illness or not as shown in figure 88, the users need to enter do they have allergic. The users can only enter 'yes' or 'no'.

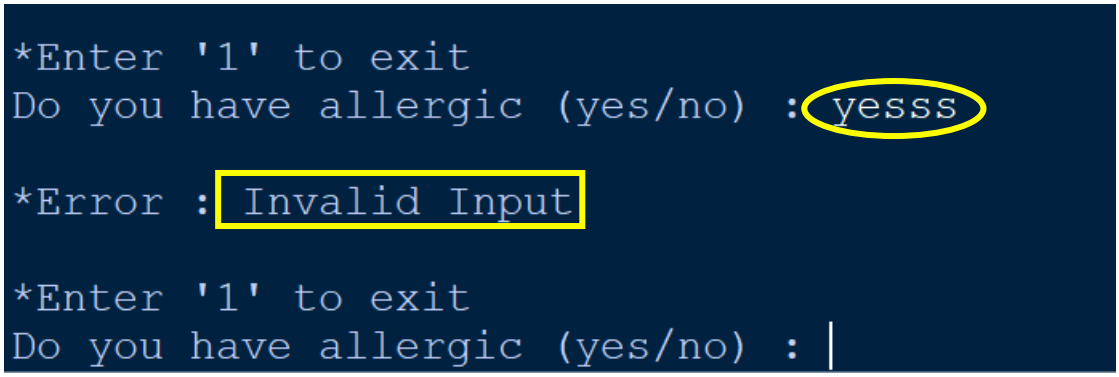


Figure 173: Invalid Input – (yes/no) [Input == 'yesss']

As shown in figure 89, the input entered by the user is invalid because the users can only enter 'yes' or 'no'.

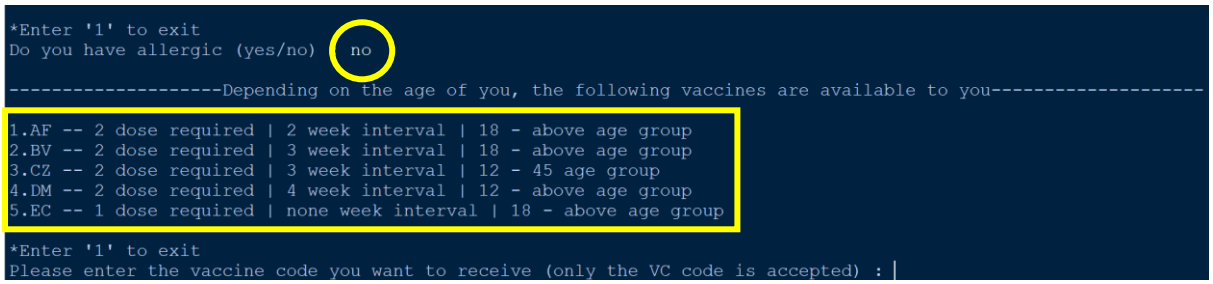


Figure 174: Correct Input [Input == 'no'], then move to the vaccine selection

After entering whether they have allergic or not as shown in figure 90, the users need to enter the selection of the vaccine. The users can only choose from the vaccines that are available to them.

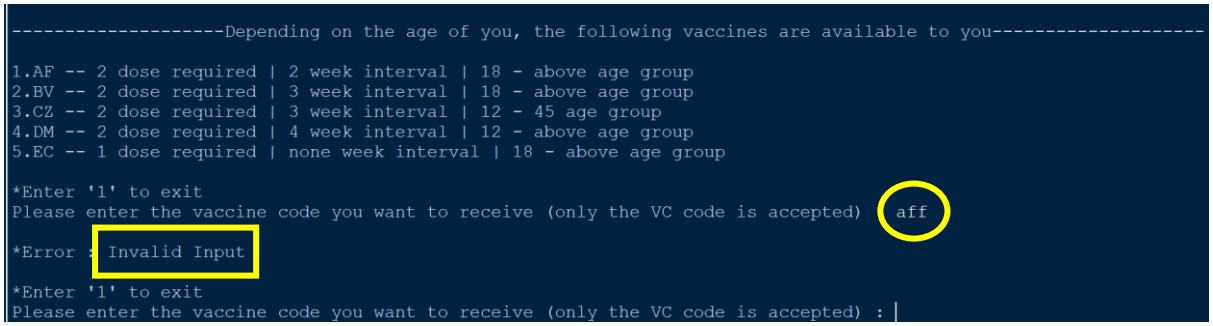


Figure 175: Invalid Input - Not included in the selection [Input == 'aff']

As shown in figure 91, the vaccine entered by the user is invalid because there is no 'aff' vaccine in the options.

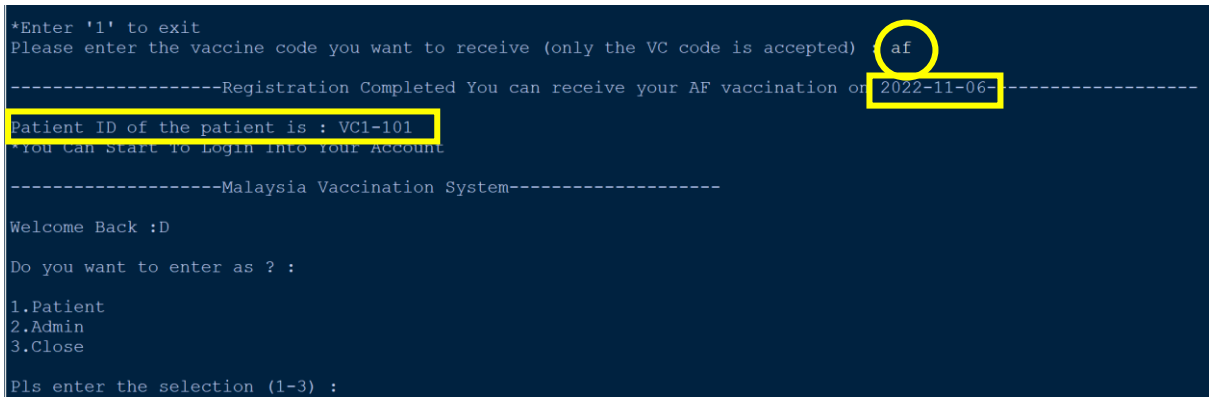


Figure 176: Correct Input [Input == 'af'], then move to the Main Menu

Once the user selects the vaccine, they will be advised a time to receive the vaccine, but they can still receive it at a time that is convenient for them. They will then also receive their patient ID. Finally, they will also be returned to the main menu, and they will be able to start logging into their account.

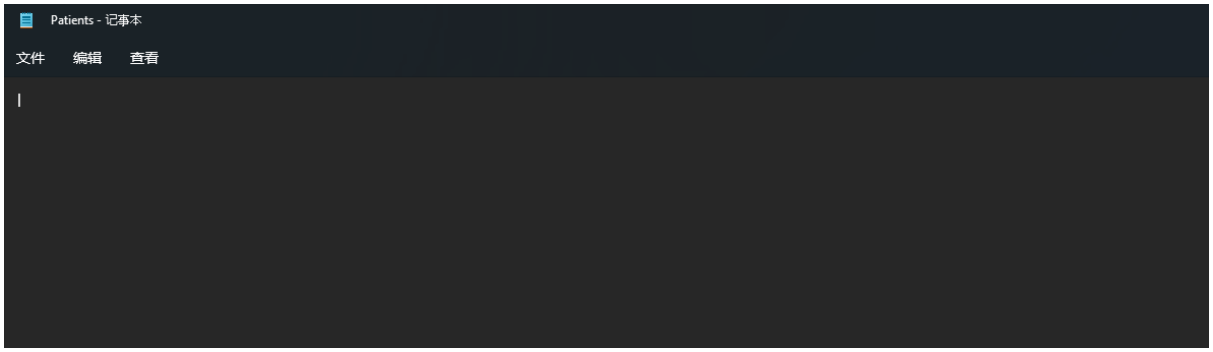


Figure 177: Before Registration

According to figure 93, the text file will not have any patient information until the user has registered.

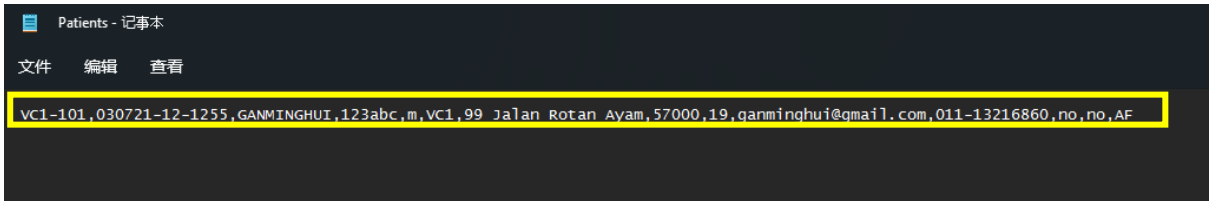


Figure 178: After Registration

According to figure 94, once the user has registered, their information will be stored in 'Patients.txt'.

```
--------------------Malaysia Vaccination System--------------------

Welcome Back :D

Do you want to enter as ? :

1.Patient
2.Admin
3.Close

Pls enter the selection (1-3) : 1

--------------------Patient Account--------------------

1.Create New Account
2.Login into Existing Account
3.Back

Pls enter the selection  2

-------------------Please login to your personal account-------------------

*Enter '1' to exit
Please enter your IC :
```

Figure 179: Login Account

If users want to log in to their account, they first need to enter the vaccine system as a 'Patient' in the main menu of the system, which is option '1'. Then they have to select '2' on the 'Patient account' page, which is to login account.

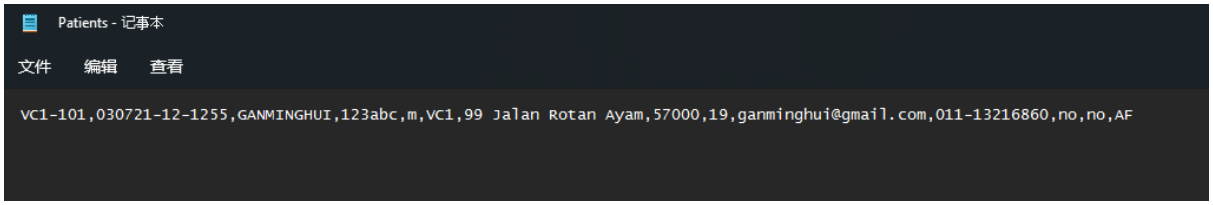


Figure 180: IC & Password

IC = 030721-12-1255

Password = 123abc

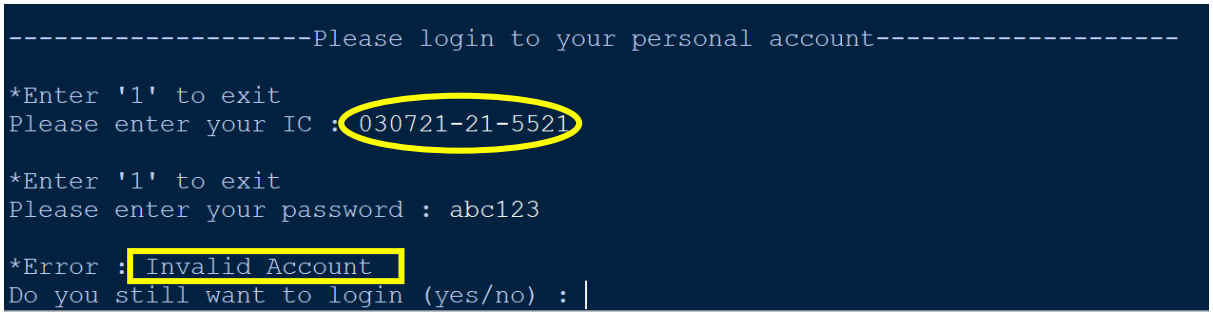


Figure 181: Invalid Account - Wrong IC [Input == '030721-21-5521']

As shown in figure 97, the user cannot log in to the account because the IC has been entered incorrectly.

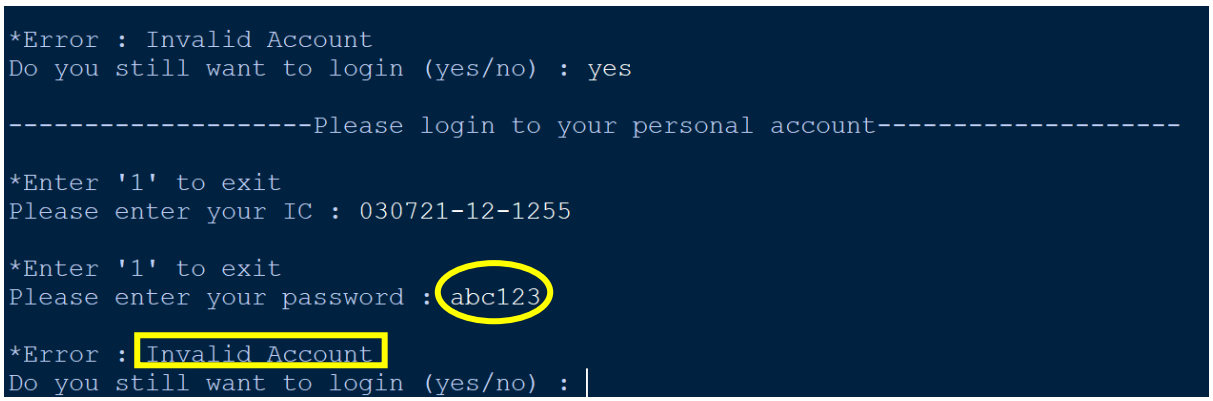


Figure 182: Invalid Account - Wrong Password [Input == 'abc123']

As shown in figure 98, the user cannot log in to the account because the password has been entered incorrectly.

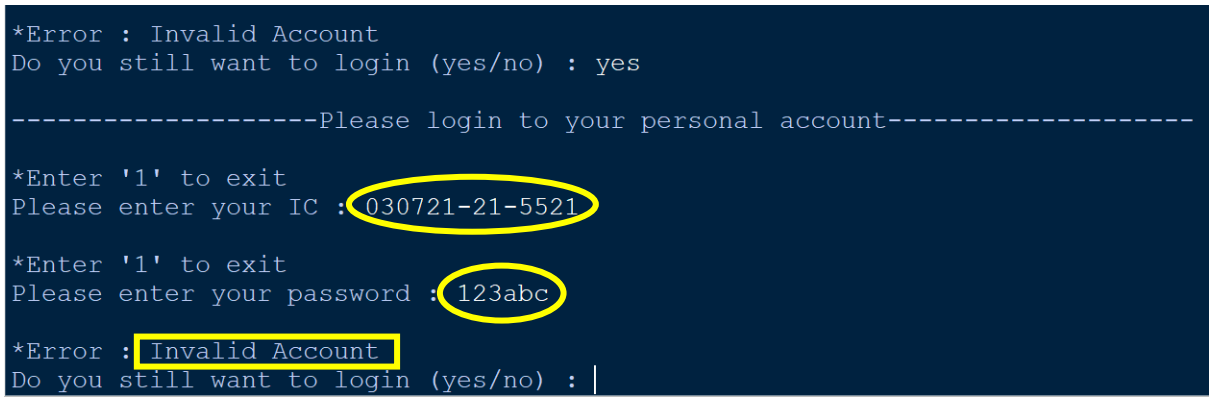


Figure 183: Invalid Account - Wrong IC & Password

[Input == '030721-21-5521' & '123abc']

As shown in figure 99, the user cannot log in to the account because the IC and password has been entered incorrectly.

```
--------------------Please login to your personal account--------------------

*Enter '1' to exit
Please enter your IC : 030721-21-5521

*Enter '1' to exit
Please enter your password : 123abc

*Error : Invalid Account
Do you still want to login (yes/no) : no

---------------------Malaysia Vaccination System---------------------

Welcome Back :D

Do you want to enter as ? :

1.Patient
2.Admin
3.Close

Pls enter the selection (1-3) :
```

Figure 184: Exit from login Account [Input == 'no']

As shown in figure 100, if the user does not want to continue logging into their account, they can answer 'no' and they will be taken back to the system's main menu.

```
*Error : Invalid Account
Do you still want to login (yes/no) : yes

--------------------Please login to your personal account--------------------

*Enter '1' to exit
Please enter your IC :
```

Figure 185: Continue to Login Account [Input == 'yes']

As shown in figure 101, if the user wants to continue logging into their account, they can answer 'yes' and they will be able to continue to login into their account.

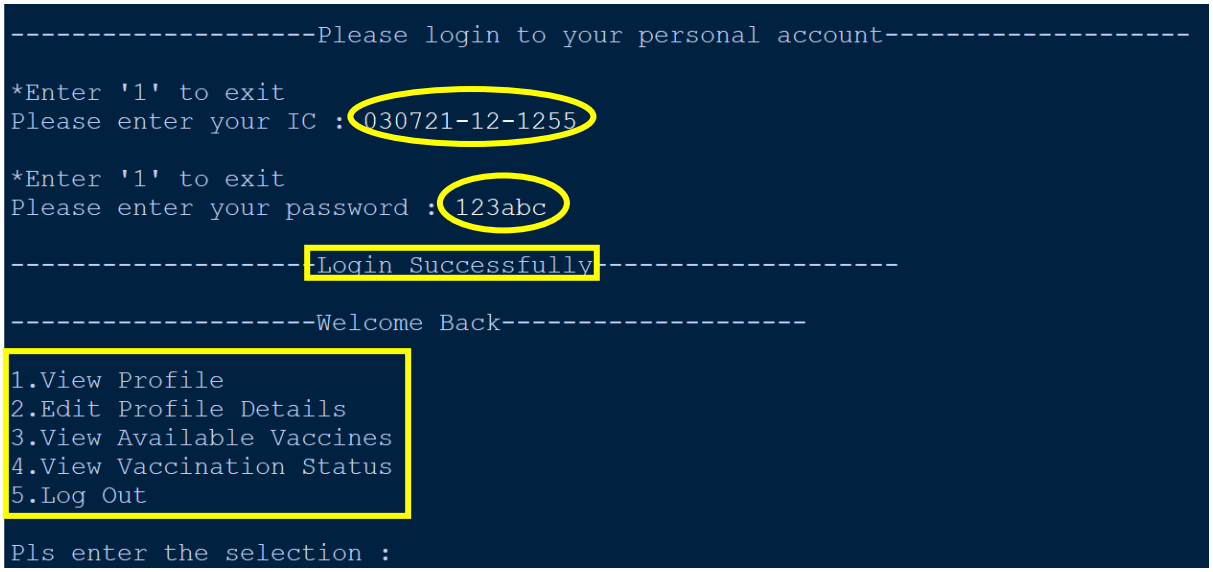


Figure 186: Successfully Login into Account [Input = '030721-01-1255' & '123abc']

As shown in figure 102, when the user successfully logs in, they will see a user menu. This menu contains five options that the user can perform. '1. They can view their profile', '2. they can modify profile information', '3. they can view what vaccines are currently available at the vaccine center', '4. they can also view their personal vaccine status and finally' and 5. they can log out from their account.

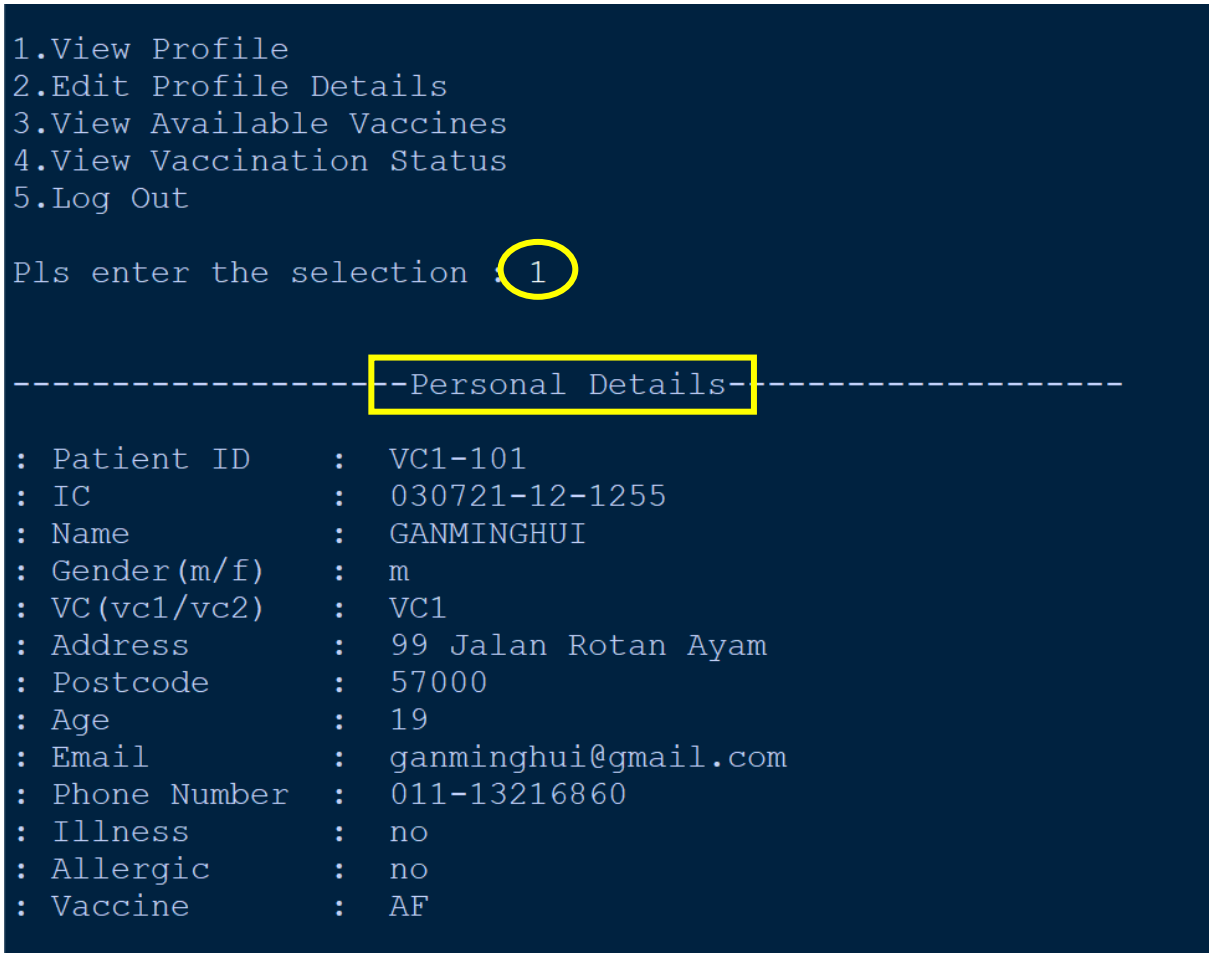


Figure 187: View Profile [Input == '1']

As shown in figure 103, users can view their profile by entering '1'. All the personal information they viewed is what they filled out when they registered

```
--------------------Login Successfully--------------------

--------------------Welcome Back--------------------

1.View Profile
2.Edit Profile Details
3.View Available Vaccines
4.View Vaccination Status
5.Log Out

Pls enter the selection : 2


--------------------Edit Profile--------------------

1.Password
2.Address & Postcode
3.Email
4.Phone Number
5.Vaccine
6.Back

Please Enter the selection :
```

Figure 188: Edit Profile Details [Input == '2']

As shown in figure 104, users can also modify their personal information by entering '2'. They can change their account password, address & postcode, email, phone number and the vaccine they want to receive.

```
--------------------Edit Profile--------------------

1.Password
2.Address & Postcode
3.Email
4.Phone Number
5.Vaccine
6.Back

Please Enter the selection : 1

--------------------Edit Password--------------------

Please enter old password before edit your password :
```

Figure 189: Edit Password [Input == '1']

As shown in figure 105, after users enter '1' in order to change their password, they will be asked about their old password before they have set a new one.

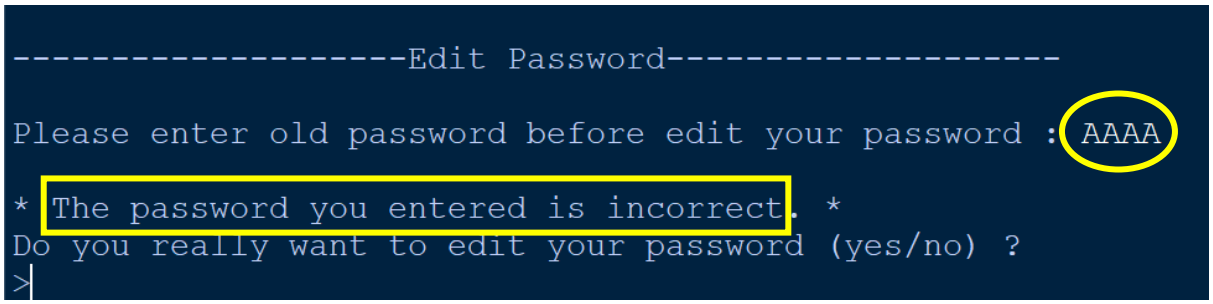


Figure 190: Unable to Change Password [Input == 'AAAA']

As shown in figure 106, if the user enters an incorrect original password, they will not be able to change it and will be asked if they want to continue changing their password.

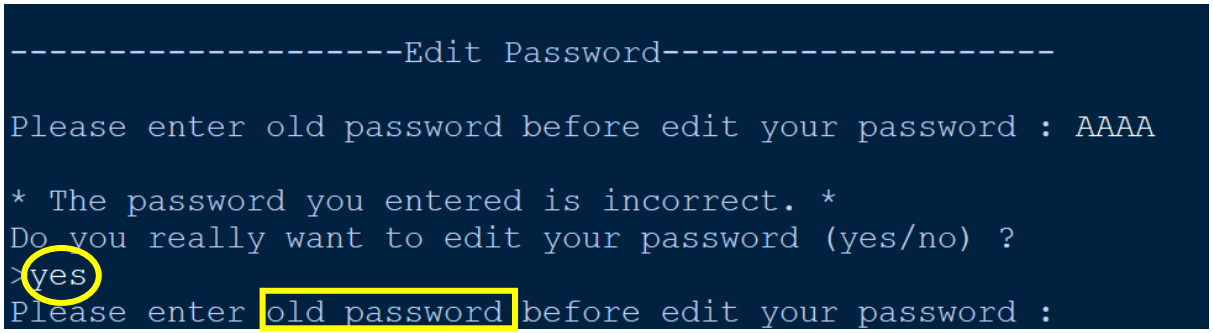


Figure 191: Edit the Password Again [Input == 'yes']

As shown in figure 107, if the user wants to continue changing their password they must enter 'yes' and they will be asked to enter the original password again.

```
* The password you entered is incorrect. *
Do you really want to edit your password (yes/no) ?
>no

*Nothing has changed*

--------------------Edit Profile--------------------

1.Password
2.Address & Postcode
3.Email
4.Phone Number
5.Vaccine
6.Back

Please Enter the selection :
```

Figure 192: Don't Want to Continue to Edit Password [Input == 'no']

As shown in figure 108, if the user does not want to continue changing their password, they must enter 'no'. As a result, nothing will be changed, and they will be taken back to the 'Edit Profile' page.

```
--------------------Edit Password--------------------

Please enter old password before edit your password : 123abc
Please enter new password : abc123
Do you really want to edit (yes/no) ?
> yes

--------------------Edited Successfully--------------------

* Password has been edited from 123abc to abc123 *
```

Figure 193: Password Edited Successfully

[Input == '123abc' AND 'abc123' AND 'yes']]

As shown in figure 109, if the user enters the correct original password, they will be able to set a new one. Once the new password has been entered, the user will be asked if they really want to change their password. If the user enters 'yes' as shown, they will successfully change their password and the new password will be updated with 'Patients.txt' as shown in figure 110 and 111.

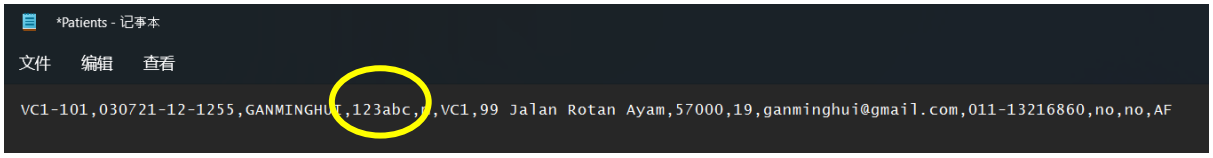


Figure 194: Password before Edit

Figure 195: Password after Edit

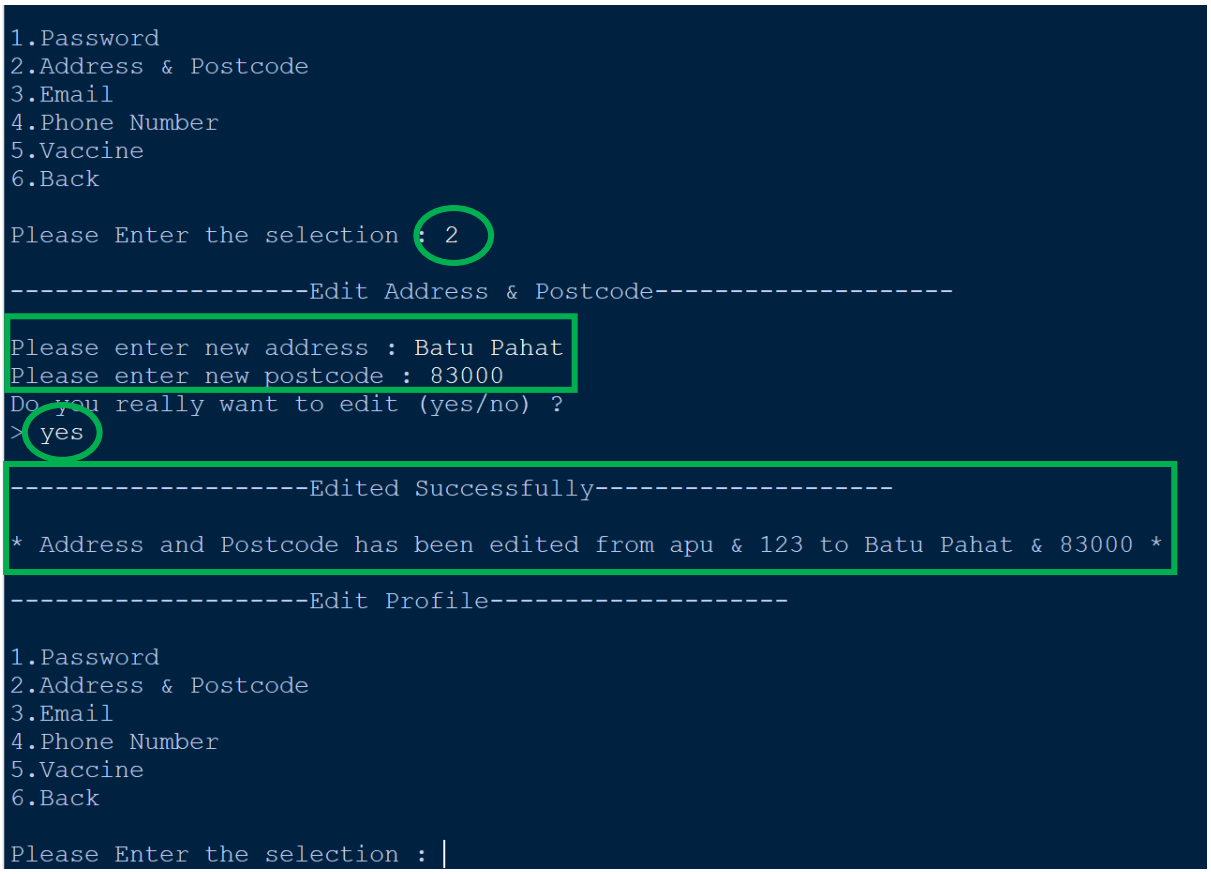


Figure 196: Address & Postcode Edited Successfully

[Input == 'Batu Pahat' AND '83000' AND 'yes']

As shown in figure 112, if the user wants to change their address and postcode, they just need to enter '2'. After they enter '2', the user will be asked for the new address and postcode, and after both are entered, the system will confirm with the user if they really want to change it (yes/no). If the user enters 'yes', they will successfully change the address and postcode.

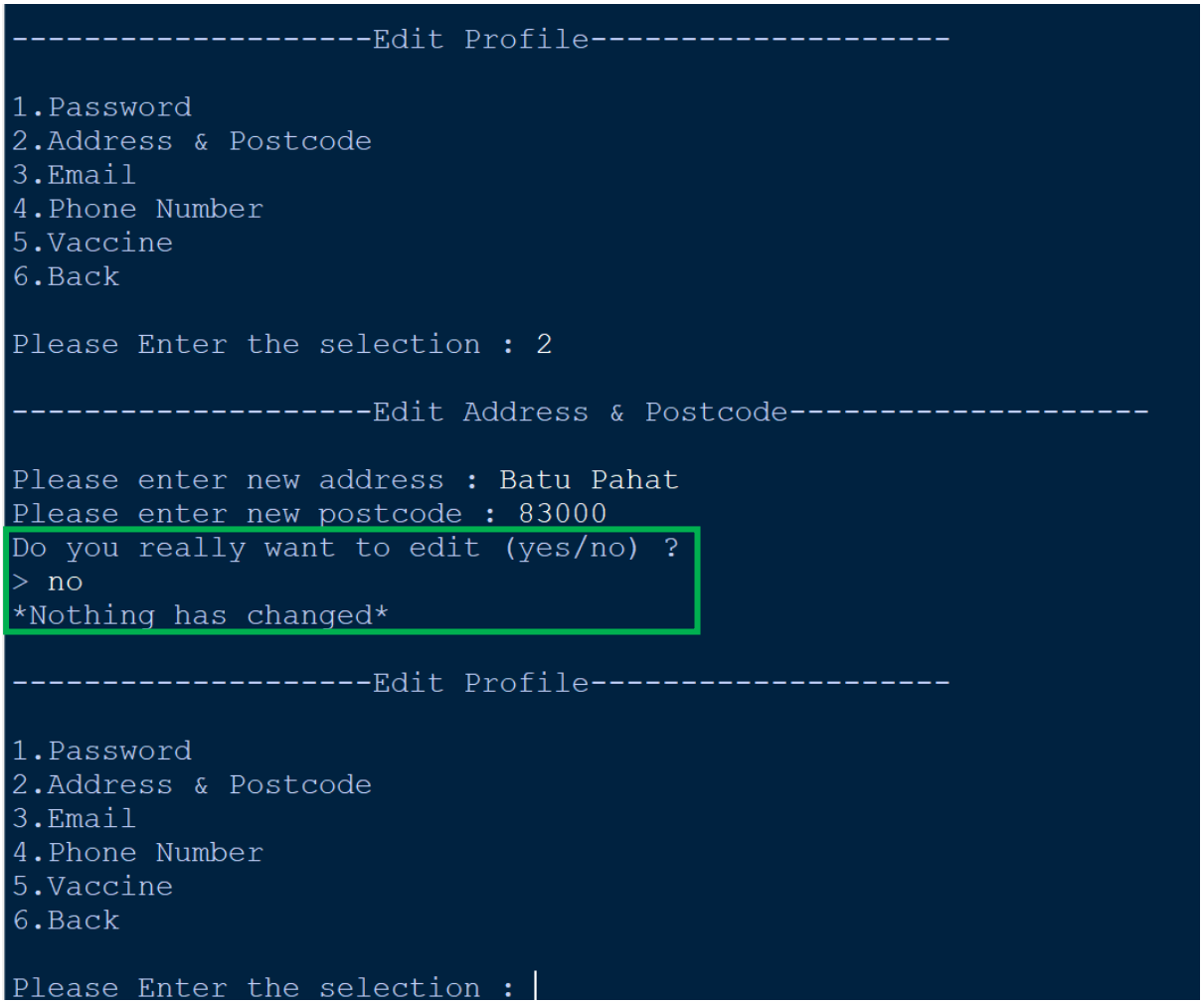


Figure 197: No Changes on Address & Postcode

[Input == 'Batu Pahat' AND '83000' AND 'no']

As shown in figure 113, if the user finishes entering the new address and postcode, but suddenly regrets not wanting to change it, they can answer 'no' when the system asks them to confirm if they want to change it.

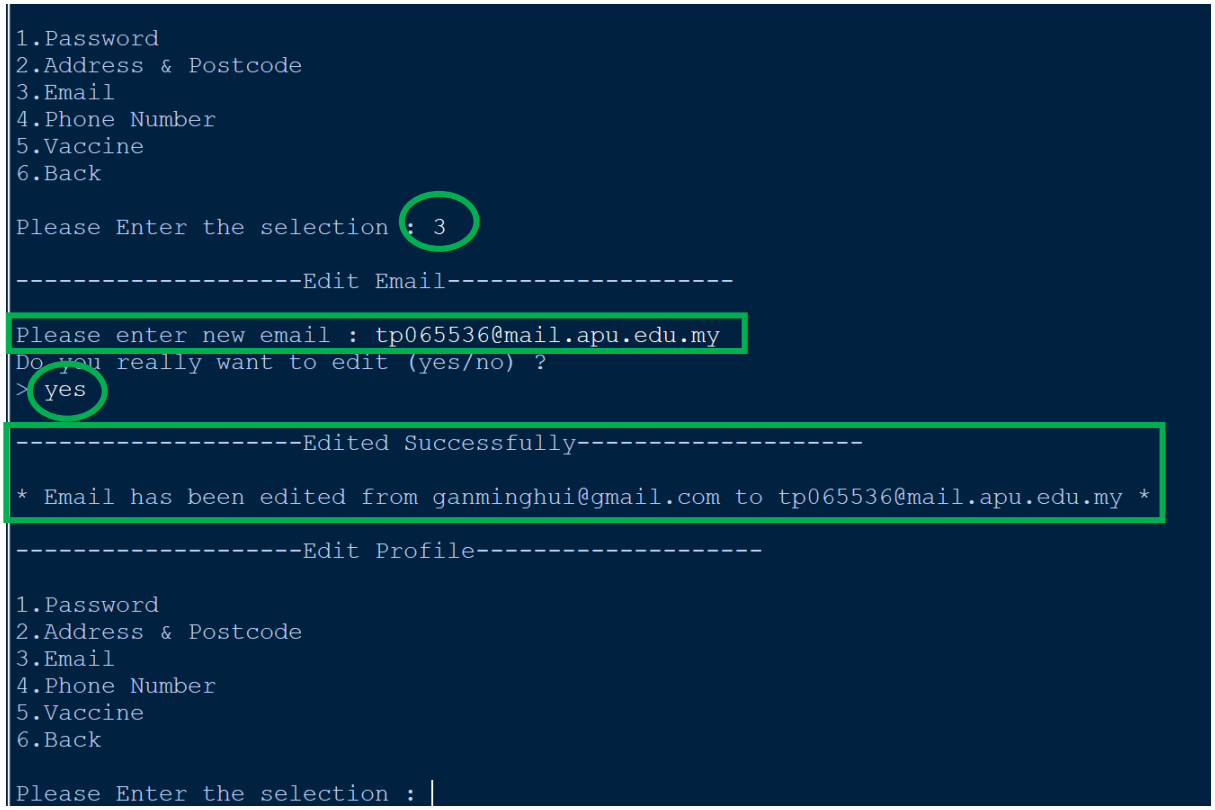


Figure 198: Email Edited Successfully

[Input == '3' AND 'tp065536@mail.apu.edu.my' AND 'yes']

As shown in figure 114, if the users want to change their email, they just need to enter '3'. After they enter '3', the users will be asked for the new email, and after new email is entered, the system will confirm with the users if they really want to change it (yes/no). If the users enter 'yes', they will successfully change the email.

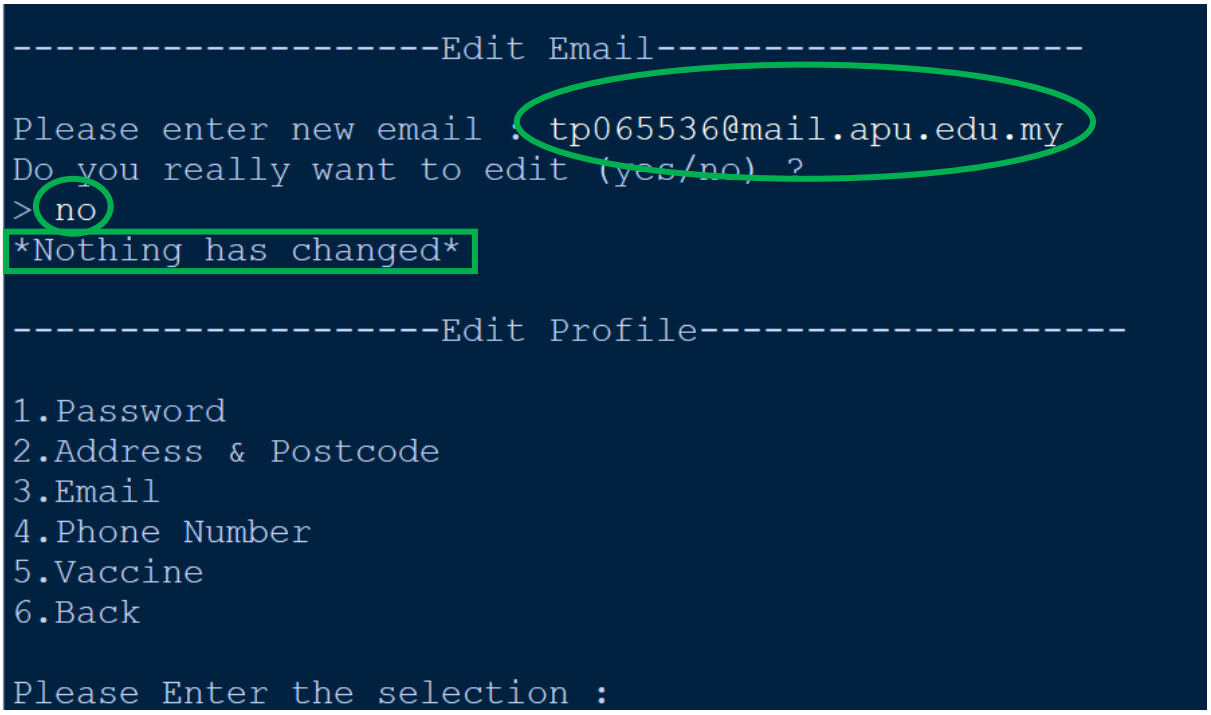


Figure 199: No Changes on Email

[Input == 'tp065536@mail.apu.edu.my' AND 'no']

As shown in figure 115, if the user finishes entering the new email, but suddenly regrets not wanting to change it, they can answer 'no' when the system asks them to confirm if they want to change it.

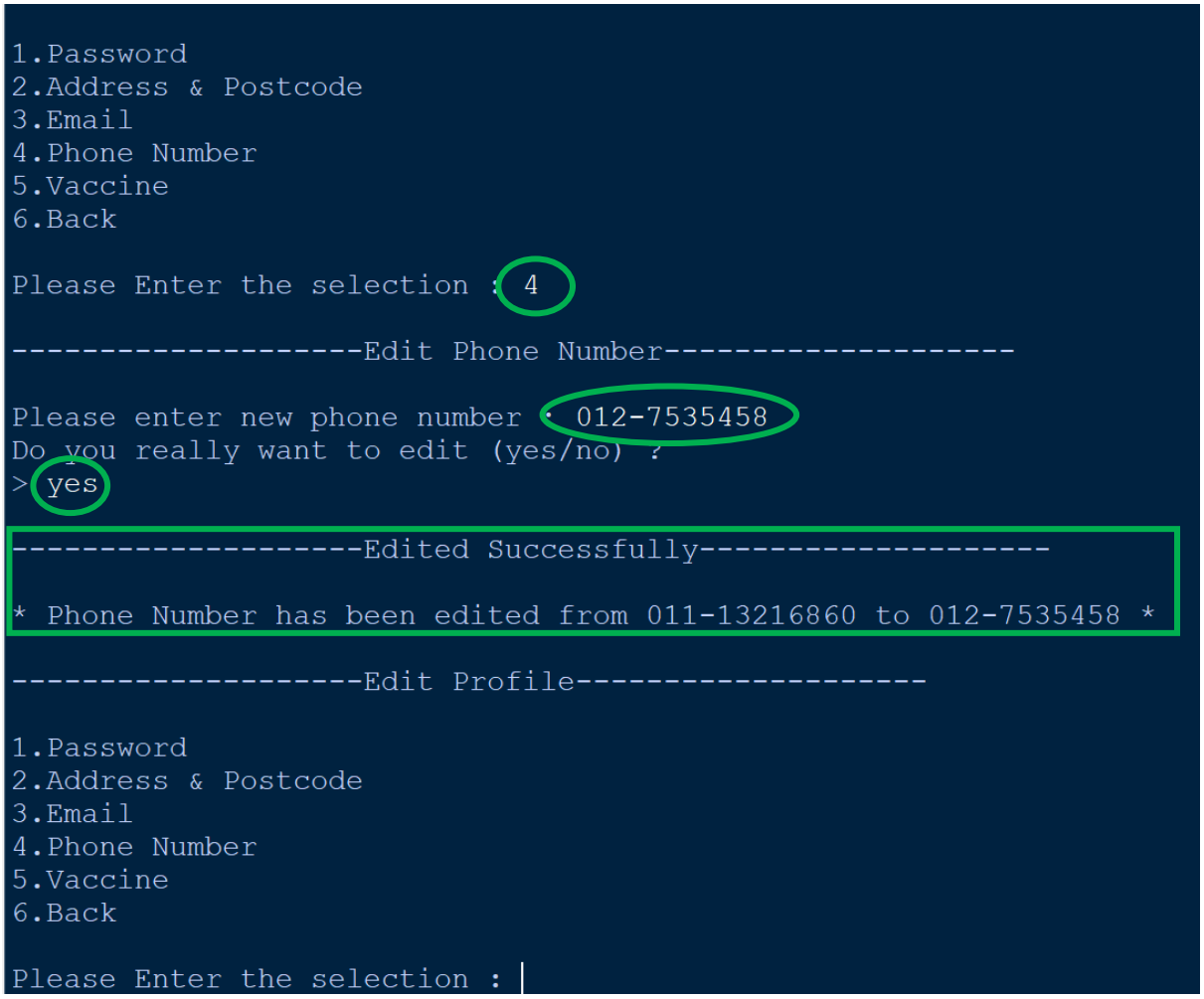


Figure 200: Phone Number Edited Successfully

[Input == '4' AND '012-7535458' AND 'yes']

As shown in figure 116, if the user wants to change their phone number, they just need to enter '4'. After they enter '4', the user will be asked for the new phone number, and after new phone number is entered, the system will confirm with the user if they really want to change it (yes/no). If the user enters 'yes', they will successfully change the phone number.

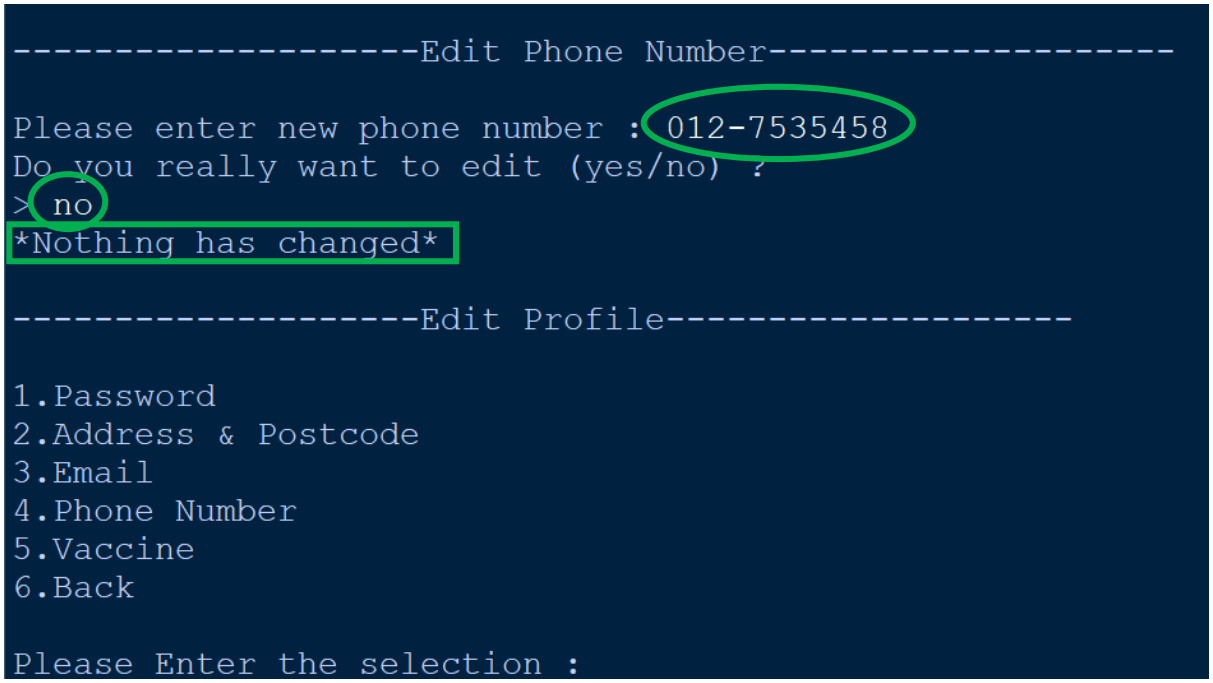


Figure 201: No Changes on Phone Number

[Input == '012-7535458' AND 'no']

As shown in figure 117, if the user finishes entering the new phone number, but suddenly regrets not wanting to change it, they can answer 'no' when the system asks them to confirm if they want to change it.

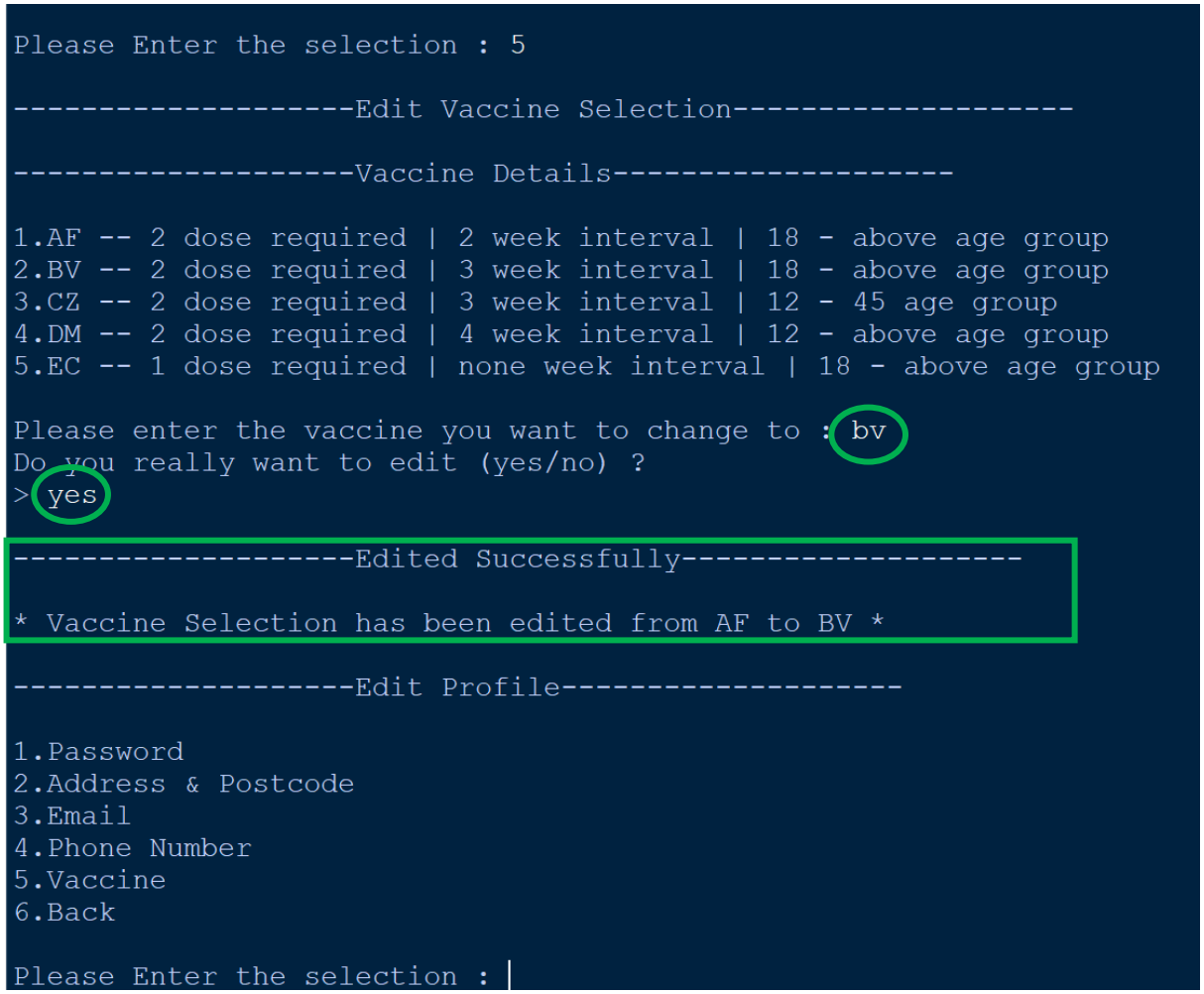


Figure 202: Vaccine Selection Edited Successfully

[Input == '5' AND 'bv' AND 'yes']

As shown in figure 118, if the user wants to change their vaccine selection, they just need to enter '5'. After they enter '5', there will be a vaccine menu where the user can select which vaccine to switch to and after new vaccine is entered, the system will confirm with the user if they really want to change it (yes/no). If the user enters 'yes', they will successfully change to receive the new vaccine.

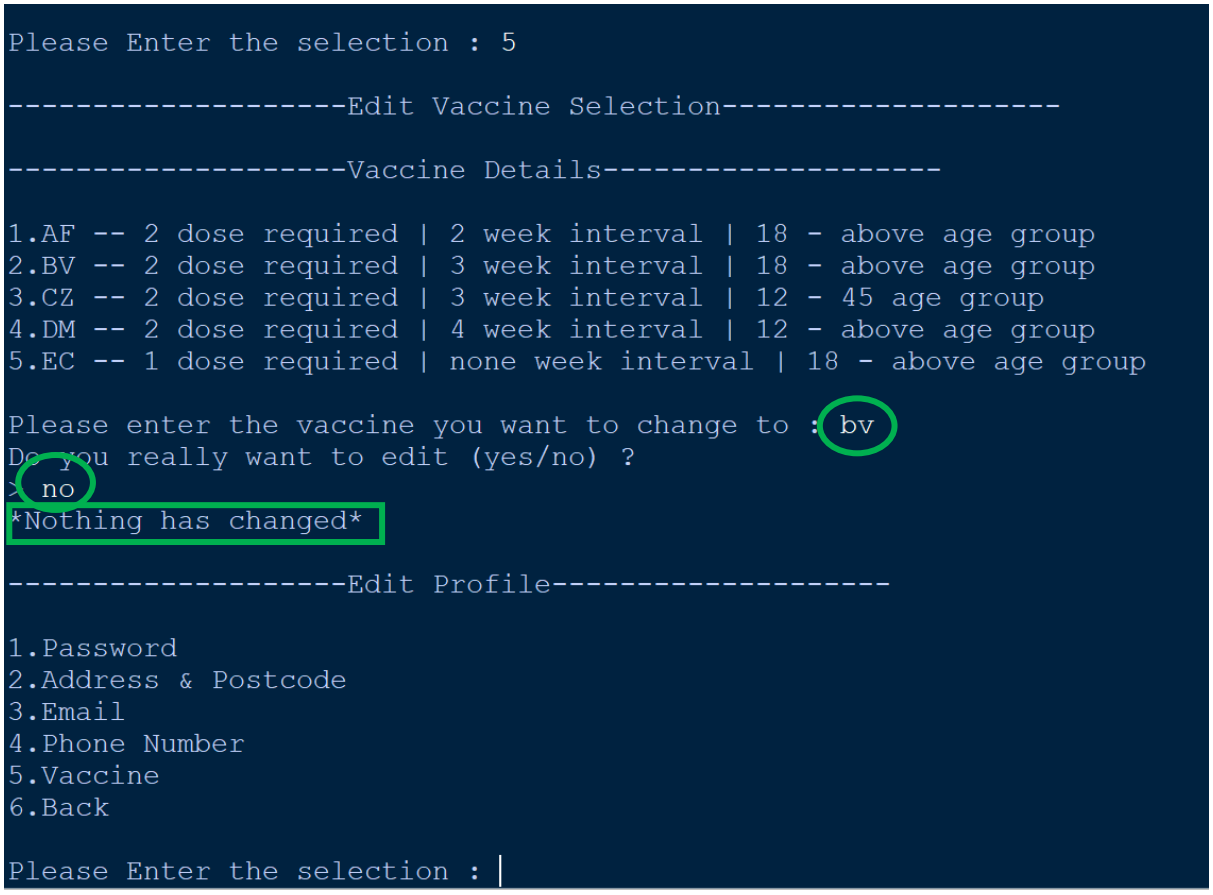


Figure 203: No Changes on Vaccine Selection

[Input == 'bv' AND 'no']

As shown in figure 119, if the user has already entered the vaccine they want to change to, but suddenly regrets not wanting to change it, they can answer 'no' when the system asks them to confirm if they want to change it.

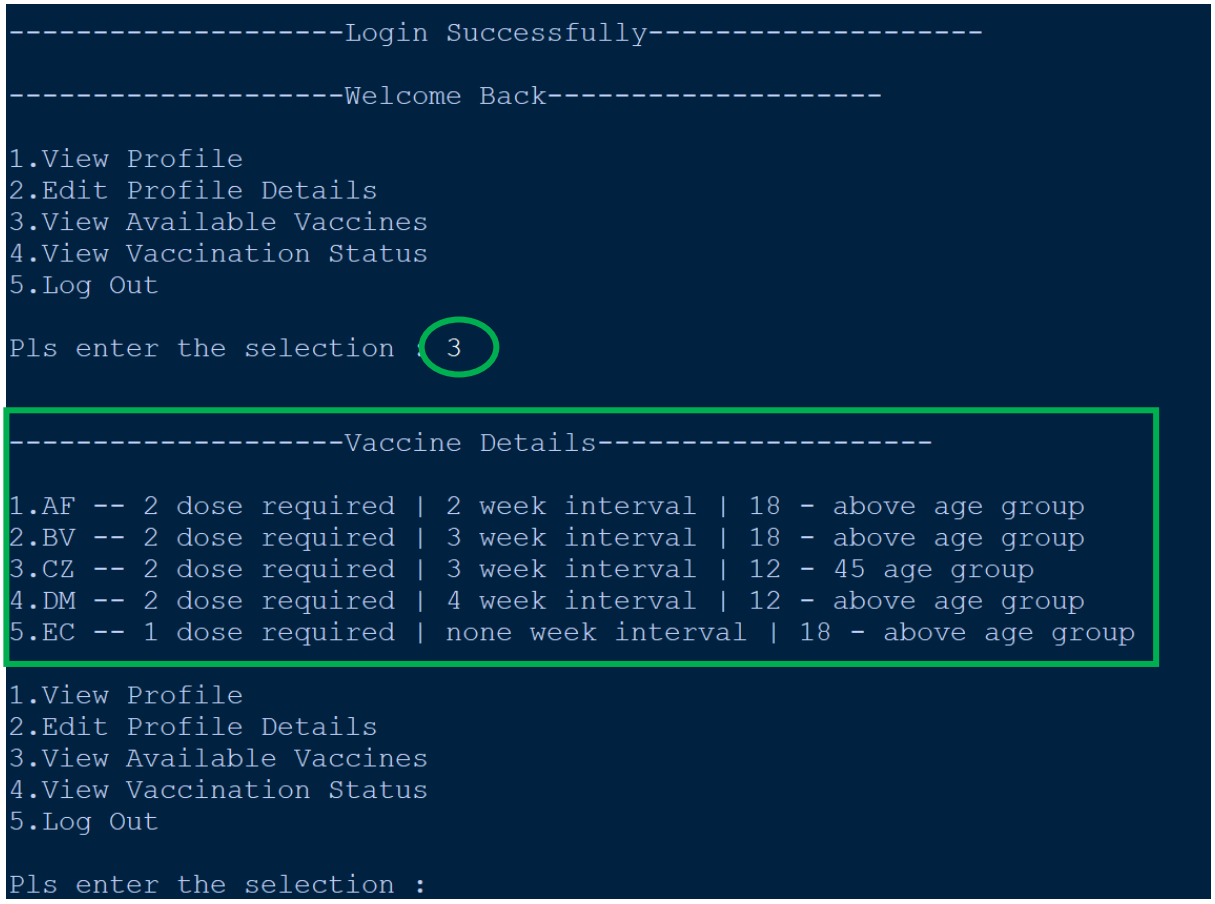


Figure 204: View Available Vaccines [Input == '3']

As shown in figure 120, if the user wants to view all the available vaccines, they just need to enter '3'. After they enter '3', all vaccines provided by the Vaccine Center and its details will be presented.

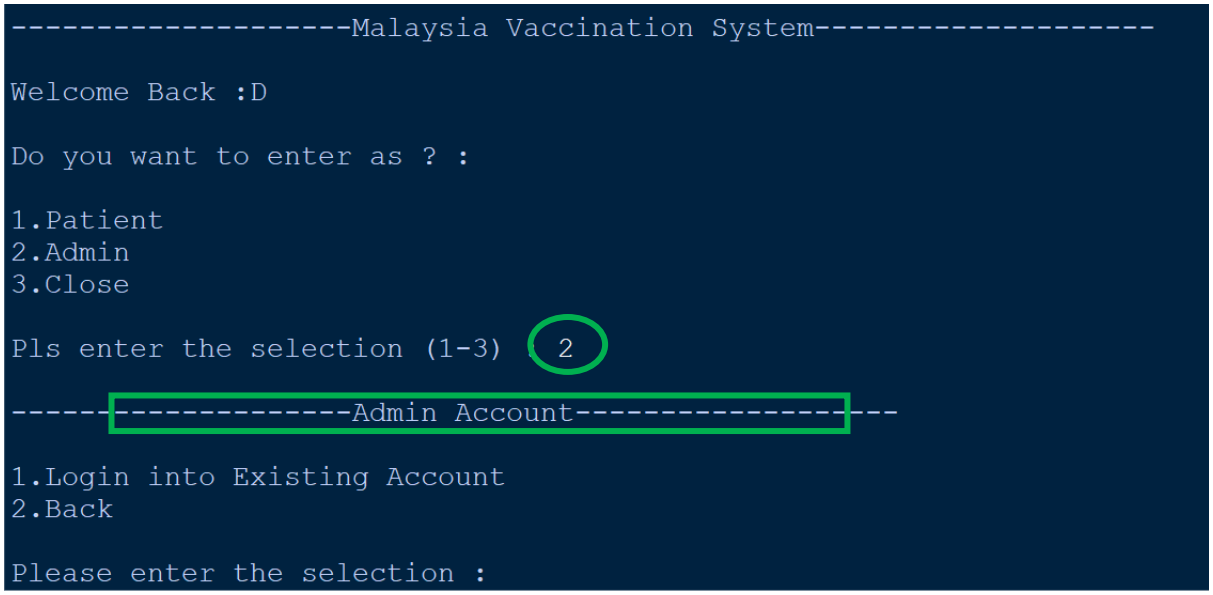


Figure 205: Admin Account [Input == '2']

As shown in figure 121 if the users want to log in as admin, they need to enter '2' in the main menu of the system.

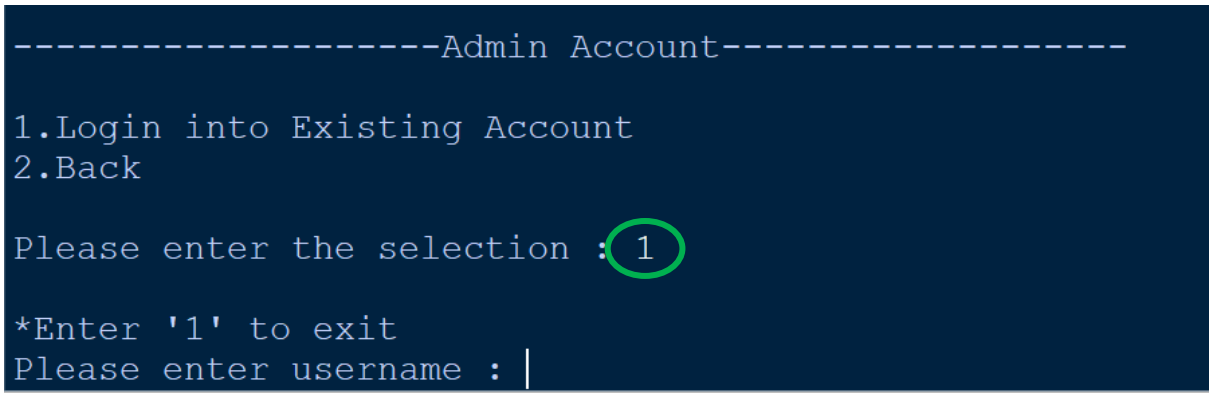


Figure 206: Login into Admin Account [Input == '1']

As shown in figure 122 when the users come to the 'Admin Account' page, the users need to type '1' to login into the admin account.

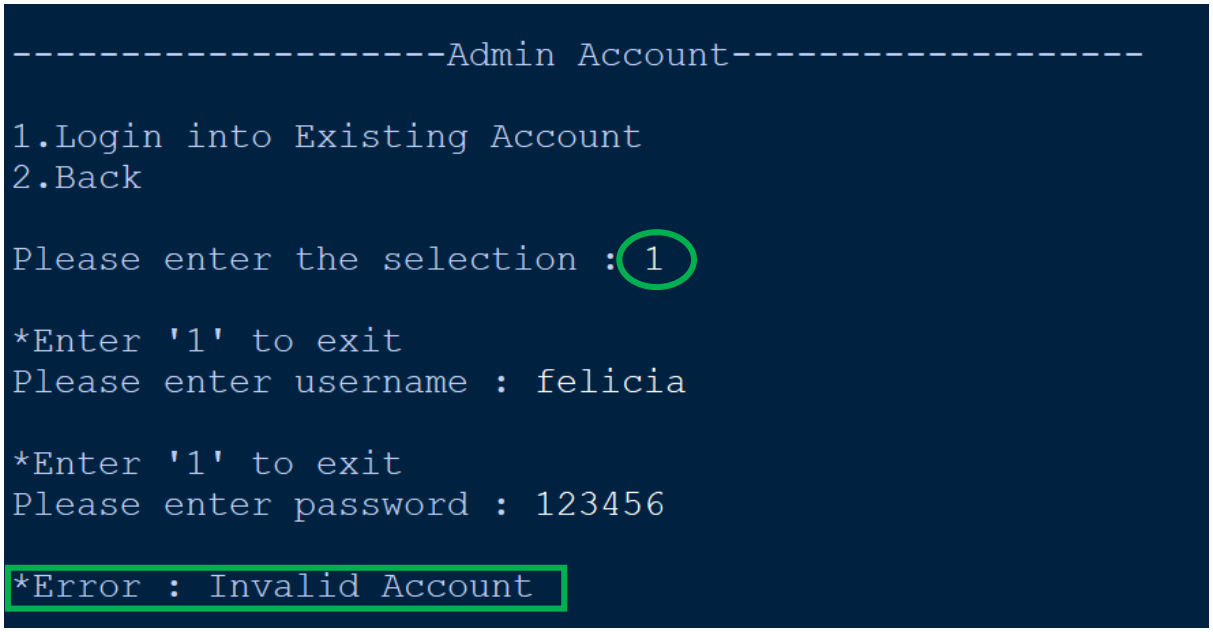


Figure 207: Invalid Account for Admin

[Input == '1' AND 'felicia' AND '123456']

As shown in figure 123 when logging in to the admin account, if the users enter an account username that is not any of the admins in the Vaccine Center, they will not be allowed to log in.

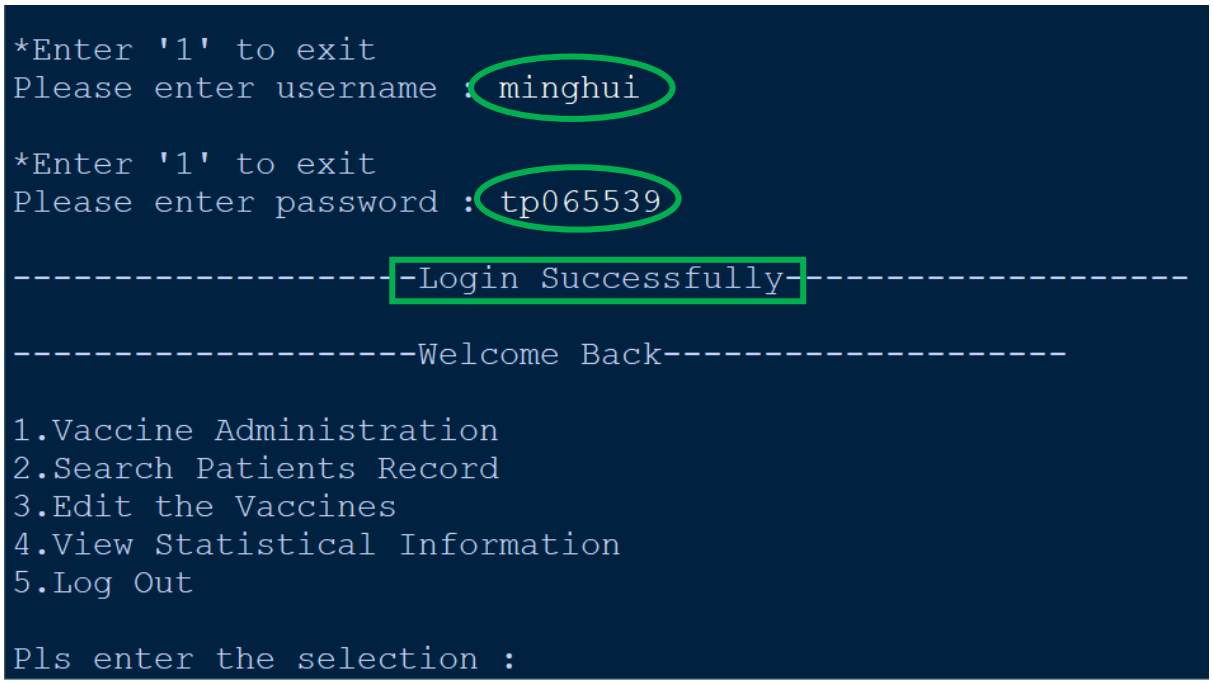


Figure 208: Admin Account Login Successfully

[Input == 'minghui] AND 'tp065539']

As shown in figure 124 once users have successfully logged into the admin account, they will see a menu with five options. 1. Vaccine Administration, 2. Search Patients Record, 3. Edit the Vaccines, 4. View Statistical Information and 5. Log Out.

```
--------------------Login Successfully--------------------

--------------------Welcome Back--------------------

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection  3

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
6.MI -- 2 dose required | 2 week interval | 18 - 50 age group
7.UO -- 2 dose required | 2 week interval | 18 - 60 age group

--------------------Edit Vaccination--------------------

1.Add New Vaccines
2.Delete Existing Vaccines
3.Back

Please Enter the selection :
```

Figure 209: Edit Vaccines [Input == '3]

As shown in figure 125 users which is the admins can make changes to the vaccine list by entering '3'. After they enter '3', they will see all the vaccines offered by the vaccine center. They can choose whether they want to add a new vaccine type or delete a vaccine on the 'Edit Vaccination' page. When they delete it, the vaccine center will no longer offer this vaccine option to patients. Also, when they add new vaccines or delete old ones, the updated vaccines are immediately reflected in the 'vaccine.txt'.

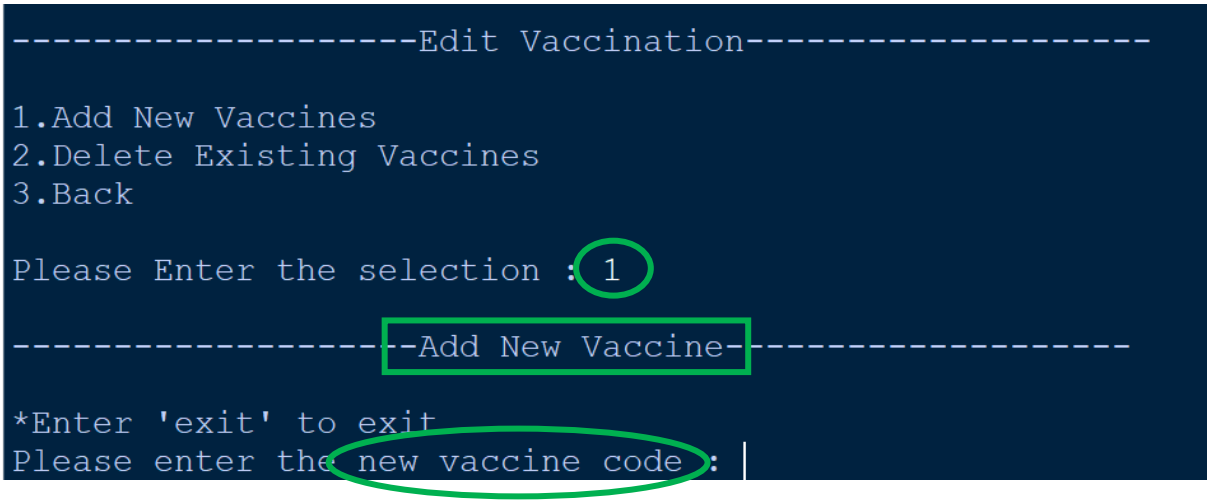


Figure 210: Add New Vaccine [Input = '1']

As shown in figure 126, users can add new vaccines by entering '1' on the 'Edit Vaccination' page. After they enter '1', they will be taken to the 'Add New Vaccines Code' page. First, they will be asked to enter the code of the new vaccine.

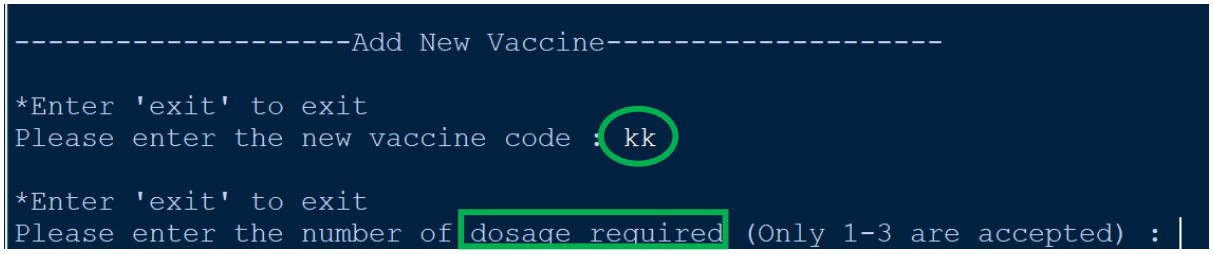


Figure 211: Vaccine Code [Input = 'kk']

As shown in figure 127 after they have entered the code for the vaccine, they will be asked to enter the dose required for that vaccine.

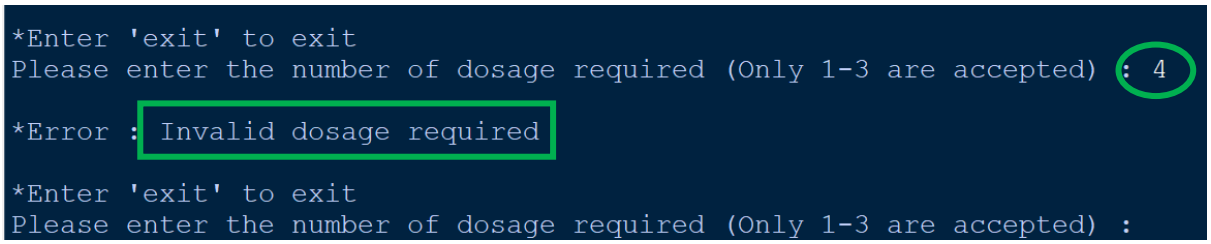


Figure 212: Invalid Dosage [Input = '4']

The dose required for the vaccine can only be 1-3. As shown in figure 128, if the users enter an input other than 1-3, it will be an invalid input.

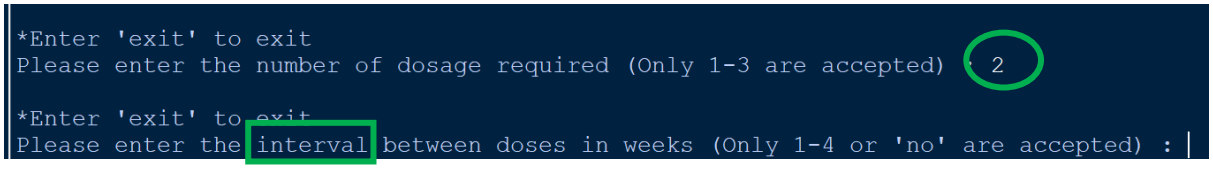


Figure 213: Valid Dosage, then move to the Interval [Input == '2']

As shown in the figure 129, if the users enter a valid required dose, they will then be asked to enter the interval for each dose and the interval between each dose can only be 1-4 weeks. For vaccines that require only one dose, the users can enter 'no' to indicate that there is no interval for that vaccine.

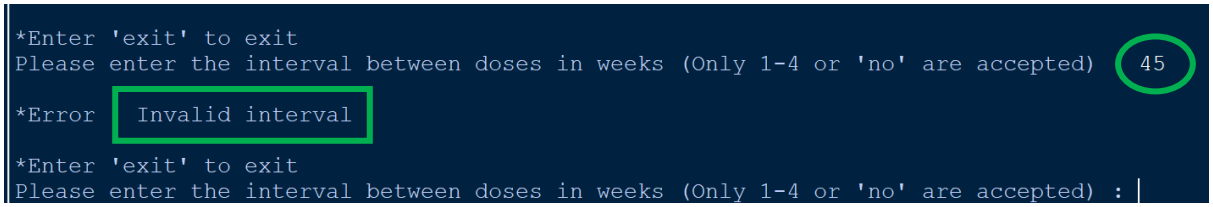
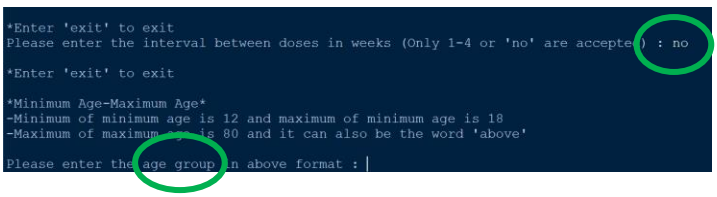


Figure 214: Invalid Interval [Input == '45']

As shown in figure 130, if the users enter an input other than '1-4' or 'no', it will be an invalid input.



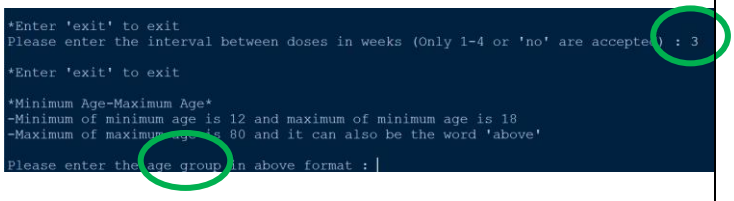


Figure 215: Valid Interval, then move to the Age Group

[Input = 'no' OR '3']

As shown in the figure 131, if the users enter a valid interval, they will then be asked to enter the age group for that vaccine and the format for entering the age group of the vaccine must be 'minimum age-maximum age'. In addition, the minimum age must be between 12-18 years old and for the maximum for the maximum age will be 80. For vaccines that do not have an exact upper age limit, the users can enter 'above' to indicate that there is no exact maximum age limit.

```
*Minimum Age-Maximum Age*
-Minimum of minimum age is 12 and maximum of minimum age is 18
-Maximum of maximum age is 80 and it can also be the word 'above'

Please enter the age group in above format : 11-50

*Error : Invalid Input

*Enter 'exit' to exit

*Minimum Age-Maximum Age*
-Minimum of minimum age is 12 and maximum of minimum age is 18
-Maximum of maximum age is 80 and it can also be the word 'above'

Please enter the age group in above format :
```

Figure 216: Invalid Age Group [Input == '11-50']

As shown in figure 132, '11-50' is an invalid input because the minimum age to receive the vaccine must be '12-18'.

```
*Minimum Age-Maximum Age*
-Minimum of minimum age is 12 and maximum of minimum age is 18
-Maximum of maximum age is 80 and it can also be the word 'above'

Please enter the age group in above format : 15-85

*Error : Invalid Input

*Enter 'exit' to exit

*Minimum Age-Maximum Age*
-Minimum of minimum age is 12 and maximum of minimum age is 18
-Maximum of maximum age is 80 and it can also be the word 'above'

Please enter the age group in above format :
```

Figure 217: Invalid Age Group [Input = '15-85']

As shown in figure 133, '15-85' is an invalid input because the maximum age to receive the vaccine cannot exceed '80'.

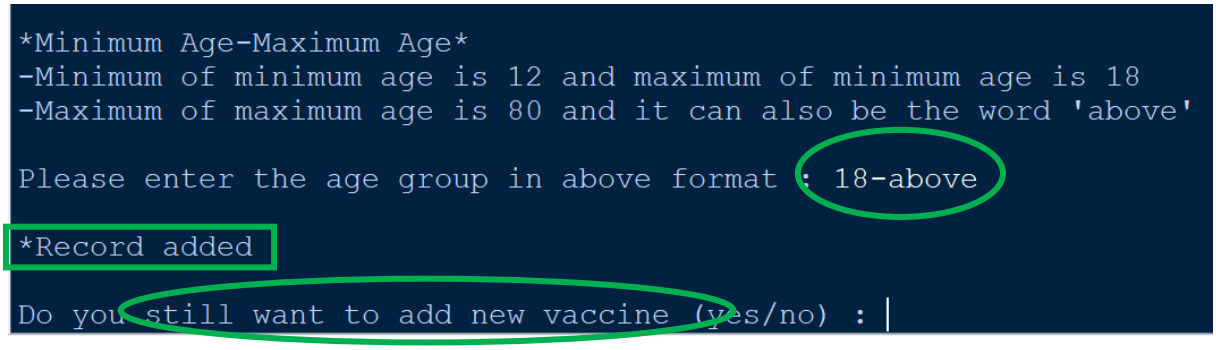


Figure 218: Valid Age Group and New Vaccine Added Successfully [Input = '18-above']

As shown in the figure 134, if the users enter a valid age group, it means that they have successfully added a new vaccine. Users will be asked if they want to continue adding new vaccines.

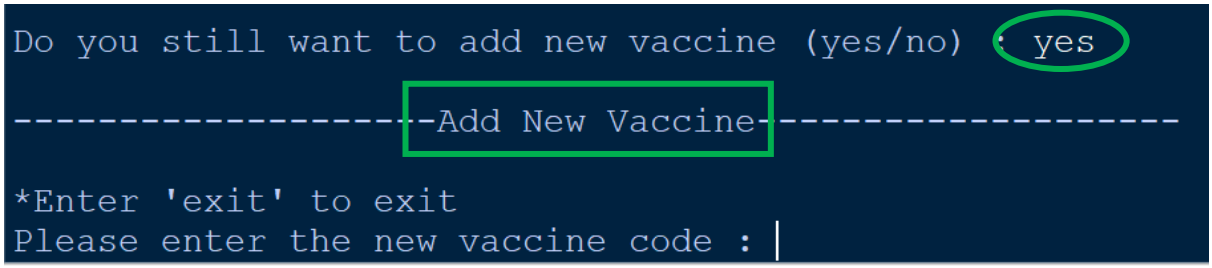


Figure 219: Continue to Add New Vaccine [Input = 'yes']

As shown in figure 135, if users choose to continue adding other new vaccine. They will be taken to the 'Add New Vaccine' page again to add other new vaccine.

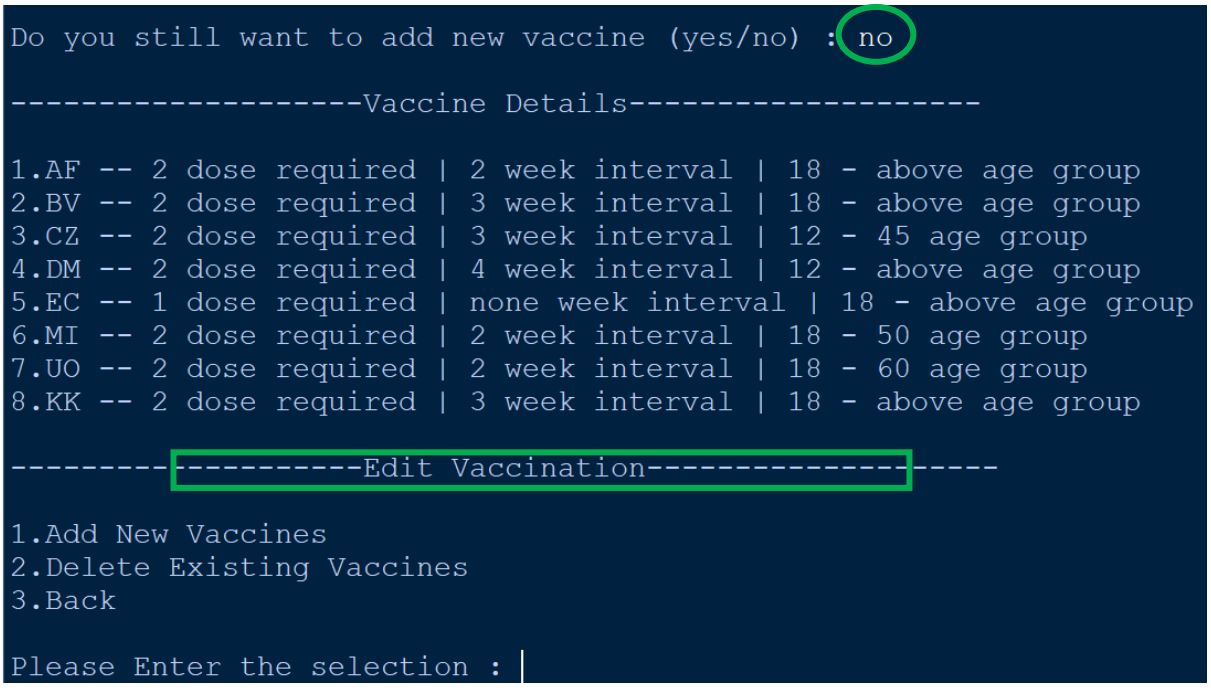


Figure 220: Don't Want to Continue Adding New Vaccine [Input == 'no']

As shown in figure 136, if the user enters 'no' to not continue adding new vaccines, then they will be taken back to the 'Edit Vaccination' menu.

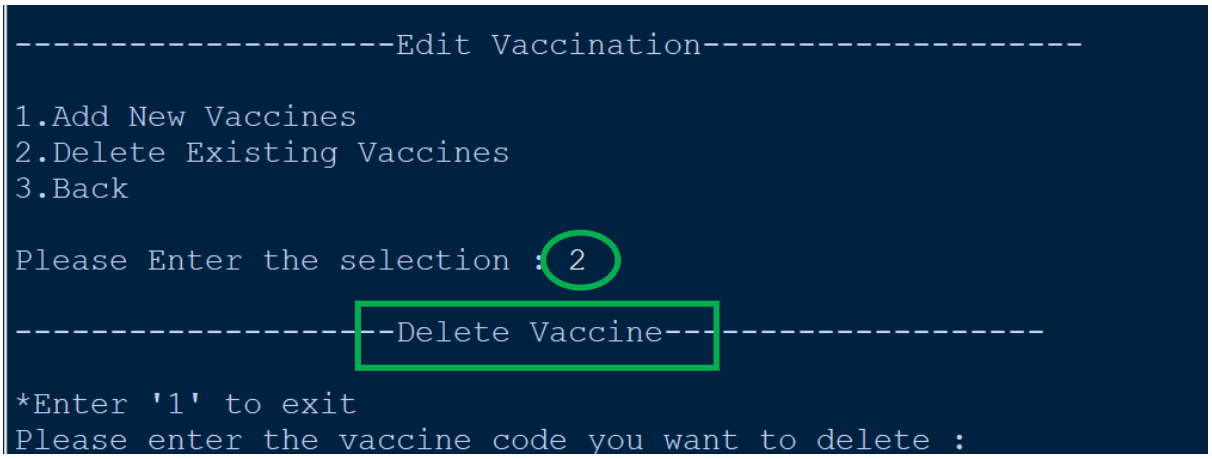


Figure 221: Delete Vaccine [Input = '2']

As shown in figure 137, users can delete existing vaccines by entering '2' on the 'Edit Vaccination' page. After they enter '2', they will be taken to the 'Delete Vaccine' page. First, they will be asked to enter the vaccine code they want to delete.

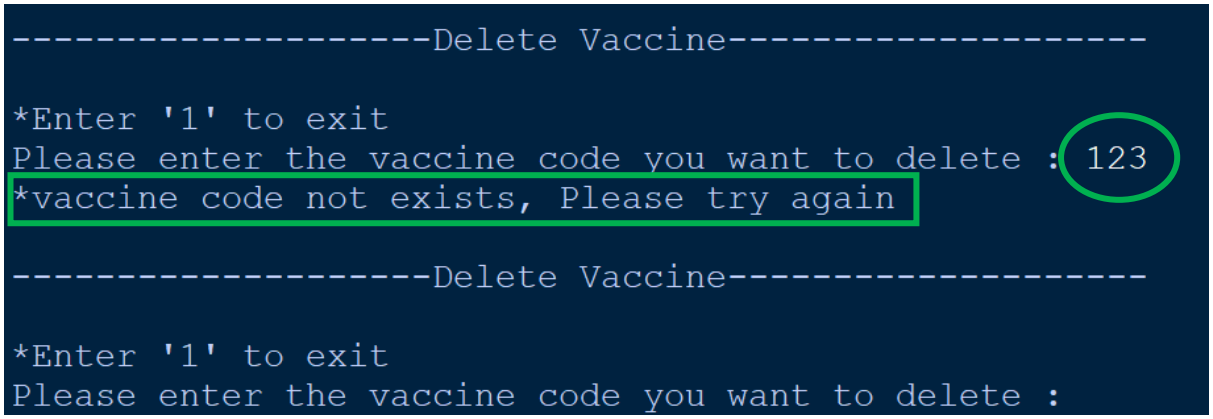


Figure 222: Unable to Delete - Vaccine Code not Exists [Input = '123']

As shown in figure 138, if the users enter a non-existent vaccine code, they will not be able to delete anything, and the system will ask the users to try again.

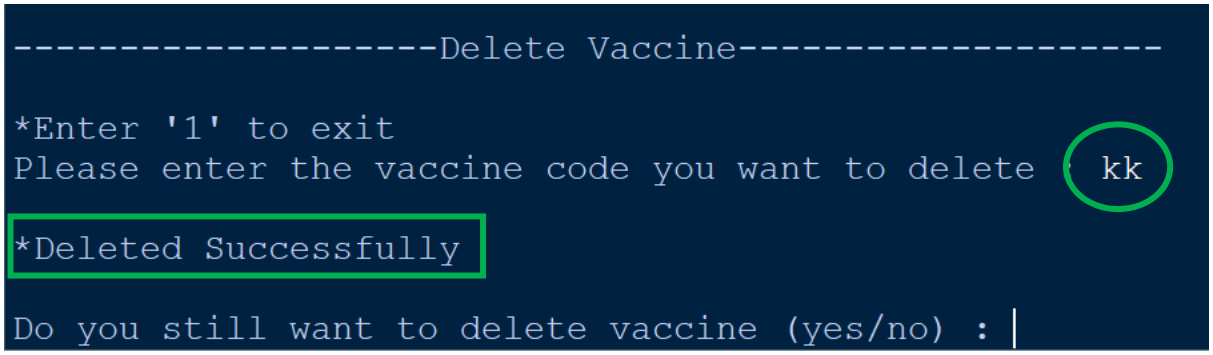


Figure 223: Vaccine Deleted Successfully [Input = 'kk']

As shown in figure 139 if the users enter a vaccine code that exists, they will successfully delete that vaccine and its details, and the users will also be asked if they want to proceed with the deletion of other vaccines.

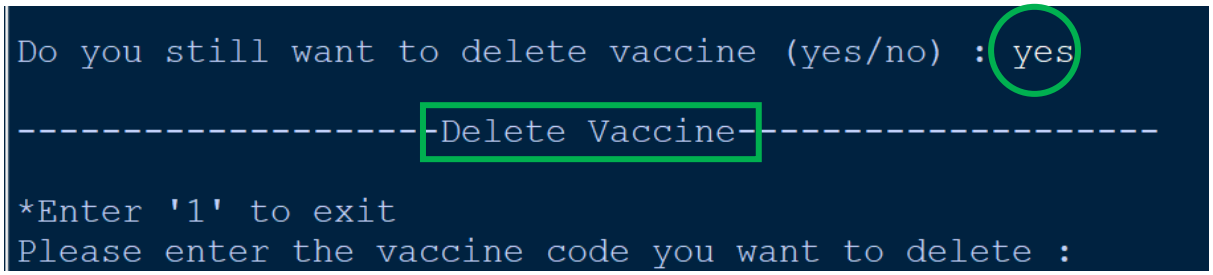


Figure 224: Continue to Delete Other Vaccine [Input = 'yes']

As shown in figure 140, if users choose to continue deleting other vaccines. They will be taken to the 'Delete Vaccine' page again to delete other existing vaccines.

```
Do you still want to delete vaccine (yes/no) : no

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group

--------------------Edit Vaccination--------------------

1.Add New Vaccines Code
2.Delete Existing Vaccines
3.Back

Please Enter the selection :
```

Figure 225: Don't Want to Continue Deleting Other Vaccine [Input = 'no']

As shown in figure 141, if the user enters 'no' to not continue deleting other vaccines, then they will be taken back to the 'Edit Vaccination' menu.

```
1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection : 4

--------------------Statistical Information--------------------

1.View Statistical Report
2.Back

Please enter the selection :
```

Figure 226: Statistical Information [Input = '4']

As shown in figure 142, users can view vaccine center statistics by entering '4'. After the users enter '4' they will be taken to the 'Statistical Information' page.

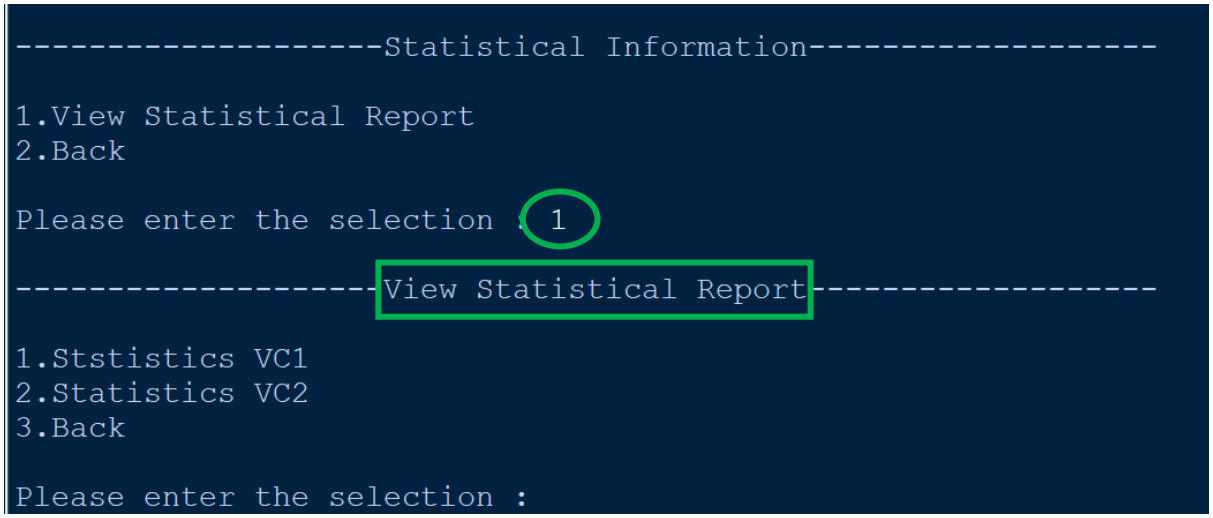


Figure 227: View Statistical Report [Input = '1']

As shown in figure 143 on the 'Statistical Information' page the users can view the statistics report by entering '1'. If the users choose to enter '1' to view the statistic report, they can choose whether they want to view the statistic report for VC1 or VC2.

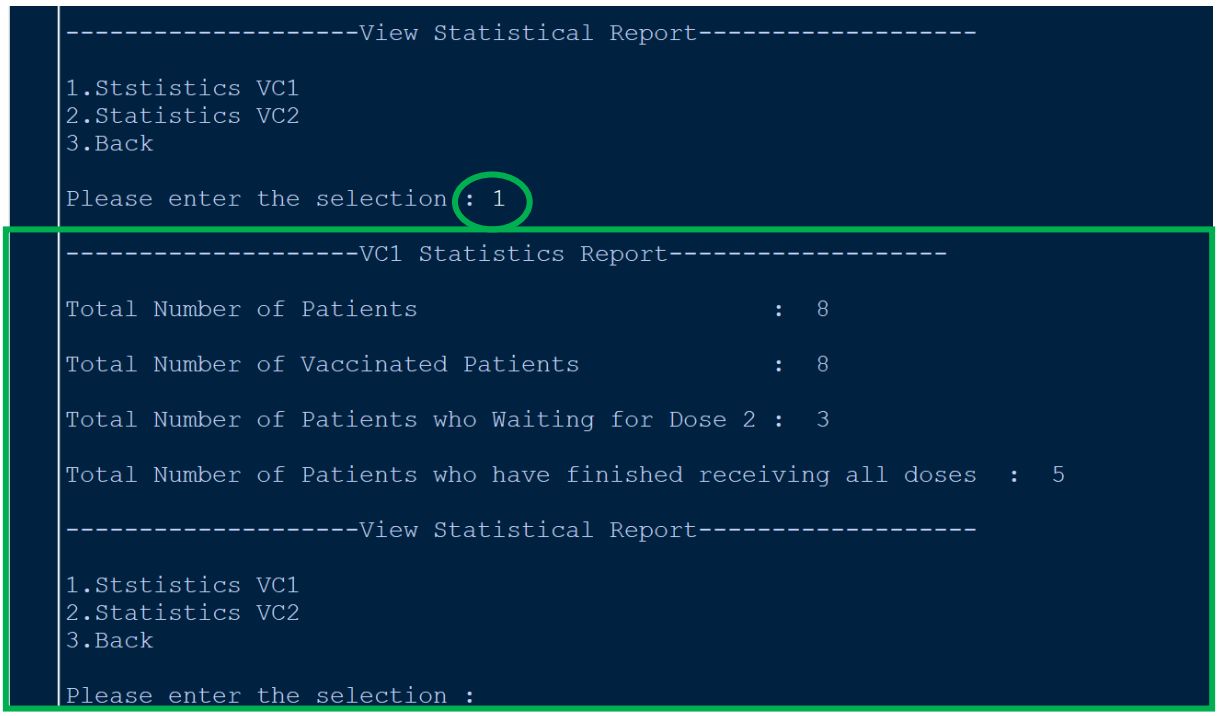


Figure 228: Statistical Report - VC1 [Input = '1']

As shown in the figure, if the users enter '1' and chooses to view the statistics report of VC1. Then the statistics report of VC1 will be presented. In this report, the user will be able to see the total number of patients, those who have already received the vaccine, those who are waiting for the second dose and those who have finished receiving all doses.

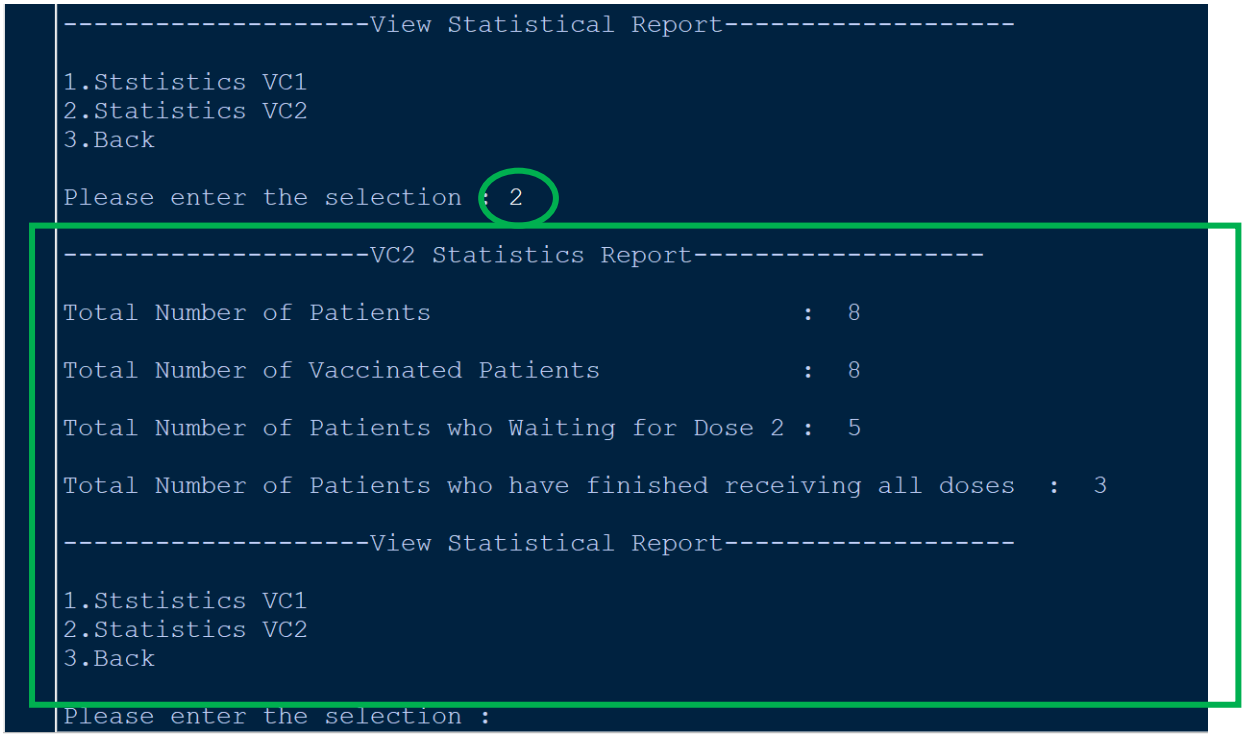


Figure 229: Statistical Report - VC2 [Input = '2']

As shown in the figure, if the users enter '2' and chooses to view the statistics report of VC2. Then the statistics report of VC2 will be presented. In this report, the user will be able to see the total number of patients, those who have already received the vaccine, those who are waiting for the second dose and those who have finished receiving all doses.

5.2 - HO FENG SHENG

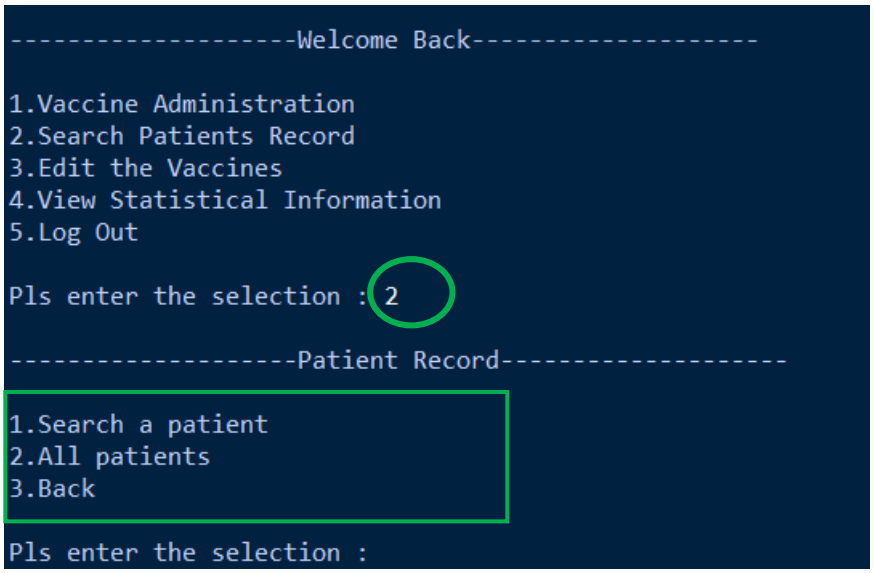


Figure : Search Patients Record [Input ==’2’]

As shown in figure admin can search patients record by entering ‘2’. After they enter ‘2’, they will see there will be 3 options for them to select which are “1. Search a patient”, “2. All patients” and “3.Back”.

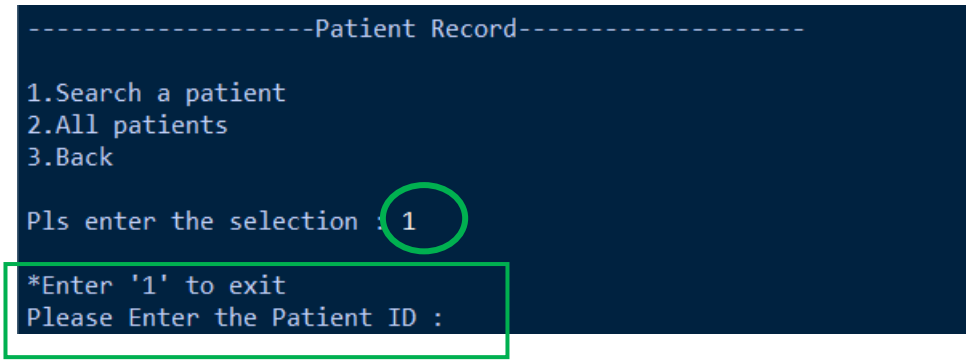


Figure: Search a patient [input == ‘1’]

If the administrator wants to search the details of a specific patient, he can enter "1", which means "Search for one patient". "Enter '1' to exit" will be displayed on the admin's screen and will also prompt the administrator to enter the patient ID he wants to search for.

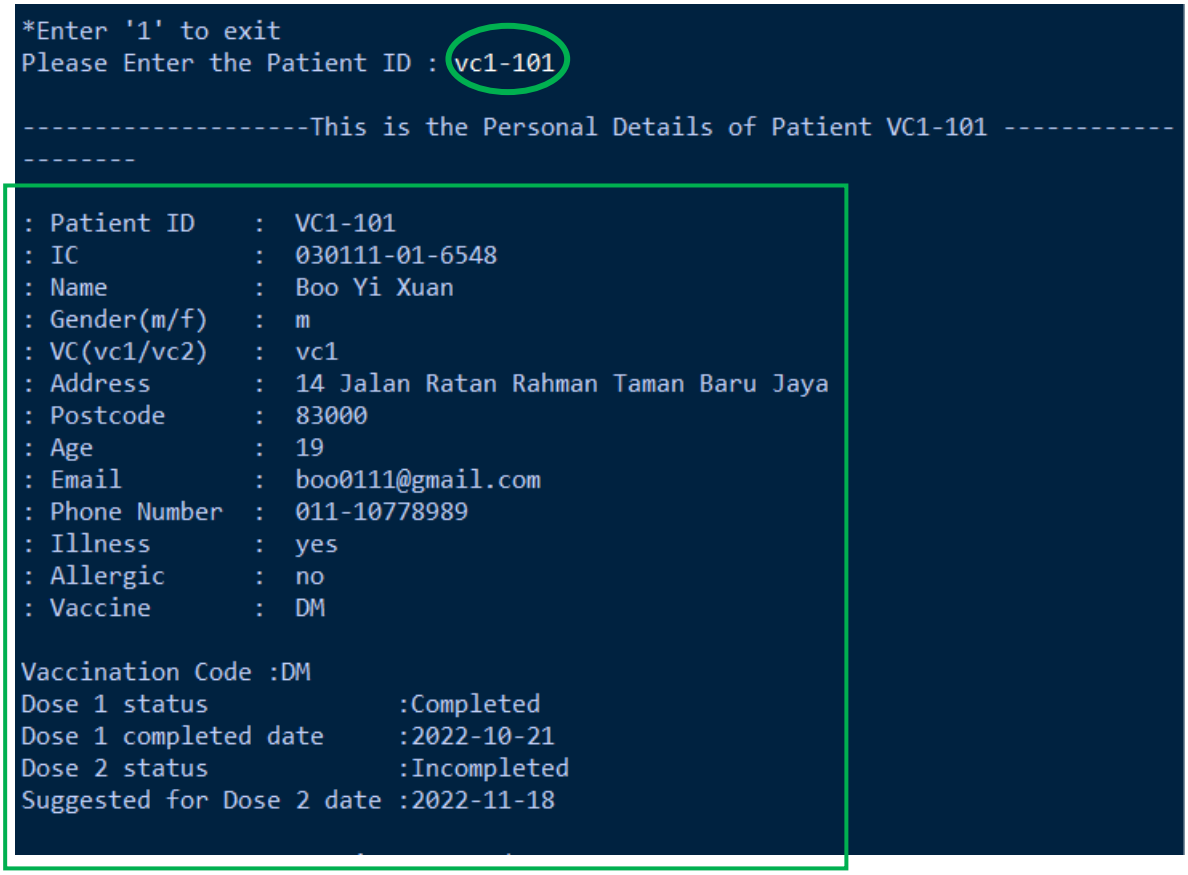


Figure: Search a patient [input = vc1-101]

As shown in the figure, after entering the patient ID that the administrator wants to view. The patient information corresponding to that patient id will be displayed.

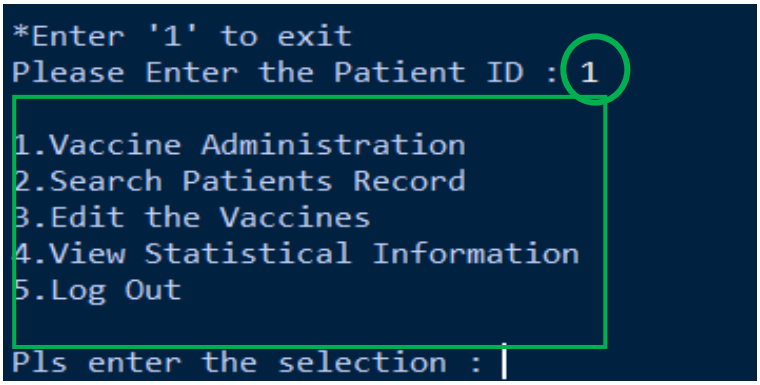


Figure: Exit [input == 1]

If the administrator enters "1" at "Please enter patient number", the system will exit this page and go to the main menu.

```
--------------------Patient Record--------------------

1.Search a patient
2.All patients
3.Back

Pls enter the selection : 2

--------------------This is the Personal Details of Patient 1 --------------------

: Patient ID    :  VC1-101
: IC            :  030111-01-6548
: Name          :  Boo Yi Xuan
: Gender(m/f)   :  m
: VC(vc1/vc2)   :  vc1
: Address       :  14 Jalan Ratan Rahman Taman Baru Jaya
: Postcode      :  83000
: Age           :  19
: Email         :  boo0111@gmail.com
: Phone Number  :  011-10778989
: Illness       :  yes
: Allergic      :  no
: Vaccine       :  DM


Vaccination Code :DM
Dose 1 status             :Completed
Dose 1 completed date     :2022-10-21
Dose 2 status             :Incompleted
Suggested for Dose 2 date :2022-11-18

--------------------This is the Personal Details of Patient 2 --------------------

: Patient ID    :  VC1-102
: IC            :  981211-10-5648
: Name          :  Bryan Ng
: Gender(m/f)   :  m
: VC(vc1/vc2)   :  vc1
: Address       :  22 Jalan Bentara 16 Perumahan Pengawai Kanan Kastam Kampung Jawa
: Postcode      :  41200
: Age           :  24
: Email         :  bryan12@gmail.com
: Phone Number  :  011-13216860
: Illness       :  no
: Allergic      :  yes
: Vaccine       :  EC
```

Figure: All patient [input ==2]

And if the administrator wants to search the details of all patients, then he can enter "2", which means "all patients". After entering "2", the details of all registered patients will be listed.

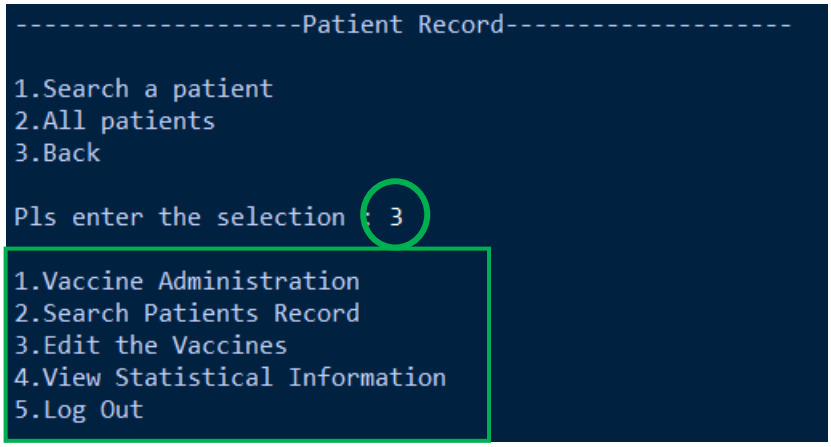


Figure: Back [input==3]

If the administrator wants to go back to the main menu, he can enter "3".

5.3 - DYANIEL CHING CHEE XIONG

```
*Enter '1' to exit
Pls enter patient id:
```

Figure 232: output of asking patient id

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) :
```

Figure 231: output for asking dosage number

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose1

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient :
```

Figure 230: output for asking vaccine code

When running the vaccine administration function, system will ask user for the **patient id**, following by the **dosage number** and the **vaccine code**. Before entering the vaccine code, there is a list about the **vaccine details of all vaccine** will be showed above the enter vaccine code column.

Dose 1 input

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose1

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : af

Submit Successfully
Second dose date: 2022-11-04

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 233: output for dose1 submit

After user key in the **correct vaccine code**, system will **submit the patient's vaccine details** into "vaccination.txt" text file and **display the second dose date** for the patient to user. Once the patient's vaccine details are submitted successfully, system will switch the page back to the admin menu page.

Dose 2 input

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose2

*Enter '1' to exit

-------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : af

Submit Successfully

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 234: output for dose2 submit

When the patient comes for having the **second dose**, most of the input and output are **same** with the situation when patient have his **first dose**. The **difference** between these two is that the user need to key in “dose2” for the dosage number and system will only show the “Submit Successfully” at the last part before back to the admin menu page.

“EC” vaccine dose 1 input

```
*Enter '1' to exit
Pls enter patient id: vc1-103

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose1

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : ec

Submit Successfully
patient get EC vaccine no need has second dose.

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 235: output for “EC” dose1 submit

Due to the patient who register “EC” vaccine only need having one dose. Therefore, the output for the patient who having **“EC” vaccine** has a little bit **different** with the previous output. After user keys in all the requirement which is asked by the system, system will display the “Submit Successfully” same with previous output and follow by a notice about **patient is having “EC” vaccine no need get the second dose.**

“EC” vaccine dose 2 input

```
*Enter '1' to exit
Pls enter patient id: vc1-103

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose2

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : ec

EC vaccine no need have second dose.

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 236: output for “EC” dose1 submit

If the patient who having **“EC” vaccine** still come for having **second dose**, the system will notify user about **“EC vaccine no need has second dose”** after user keys in all the patient’s vaccine details.

Error input

There is also the possibility that user **keys in the invalid patient's vaccine detail** during using vaccine administration function. Thus, this program has been designed with some features that are used to return the error message to the user when they key in invalid input.

```
*Enter '1' to exit
Pls enter patient id: vc1-106
Invalid patient id, pls re-submit.


*Enter '1' to exit
Pls enter patient id:
```

Figure 237: output for invalid patient id

If the user keys in the **patient id** that hasn't been registered yet, system will tell user the patient id that he keys in is an invalid patient id and ask him to **key in again.**

```
*Enter '1' to exit
Pls enter patient id: vc1-103

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose3

*Error : Invalid dosage number*

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose

*Error : Invalid dosage number*

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) :
```

Figure 238: output for invalid dosage number

For the next continue with the **dosage number** key-in. In dosage number, system will accept "dose1" and "dose2" input only. When the user keys in the dosage number other than these two, the error message **" *Error : Invalid dosage number* "** will be display out and the user need to **key in the dosage number again**.

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose1

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : bv
The vaccine code key in is different with the vaccine code chosen by patient. Patient choose AF
 vaccine.
Pls re-submit


*Enter '1' to exit
Pls enter patient id:
```

Figure 240: output for different vaccine code between dose 1 and dose register

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose2

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : bv

Patient's dose 2 type vaccine is different with dose 1. Patient's dose1 vaccine code is AF
Pls re-submit


*Enter '1' to exit
Pls enter patient id:
```

Figure 239: output for different vaccine code between dose 1 and dose2

In last part of the vaccine administration, the user accidentally key in the wrong vaccine code of the patients. The system will pop out the **error message** and tell user the **correct vaccine code**. After that, the user will be asked to **submit again**. The error message for dose 1 and dose 2 are different. The first figure above is dose 1, system will tell user the vaccine code key in is different with the vaccine code chosen by patient. The second figure is dose 2, system will display “patient's dose 2 type vaccine is different with dose 1”.

Dose status switch feature

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose2

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : af

This patient no having the first dose yet.
System will set patient in dose 1 status.

Submit Successfully
Second dose date: 2022-11-06
```

Figure 241: output for change patient dose status from dose2 to dose1

When the user accidentally **keys in dose 2** in the dosage number for the patient who wants to have his **first dose**, there is a “dose status switch” feature in the system to help user straight away **switch the patient’s dose status** and submit patient’s vaccine details in dose 1 status without key in the details again. Before switching the patient’s dose status, the system will **display a message** to tell the user about the patient hasn’t got the first dose.

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose1

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : af

This patient has done first dose.
System will set patient in dose2 status

Submit Successfully
```

Figure 242: output for change patient dose status from dose1 to dose2

If the situation changes to user **submit dose1 for the second dose patient**, the **output** by the system is also **quite same** with the previous. The **difference** with the previous is only the message will be changed to **"This patient has done first dose"** and system will **change patient into dose 2 status**.

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose2

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : af

The second dose date of patient haven't reach, patient cannot have second dose today.
Second dose date is 2022-11-06 00:00:00
```

Figure 243: output for second dose date haven't reach

The situation about patient forgets his second dose date and **comes before his second dose date** is possible occur in the real life. Hence, after the user keys in all the patient's vaccine details of a patient who haven't reach his second dose date, system will **inform user** the second dose date haven't reach and **show the second dose date** to user.

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose2

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : af

This patient has done second dose!
```

Figure 244: output for patient come again after having both dose

In this program, there is also a feature to overcome the situation when the patient who have **done his both dose** and **come again** for having vaccine. After the user keys in all the patient's vaccine details, system will find out current patient has done both dose and print **"This patient has done second dose!"** on the screen to notify the user.

Exit feature

```
*Enter '1' to exit
Pls enter patient id: 1

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 245: output for exit administration feature when key in patient id

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : 1

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 247: output for exit administration feature when key in dosage number

```
*Enter '1' to exit
Pls enter patient id: vc1-104

*Enter '1' to exit
Pls enter the dosage that patient get today (dose1 / dose2) : dose1

*Enter '1' to exit

--------------------Vaccine Details--------------------

1.AF -- 2 dose required | 2 week interval | 18 - above age group
2.BV -- 2 dose required | 3 week interval | 18 - above age group
3.CZ -- 2 dose required | 3 week interval | 12 - 45 age group
4.DM -- 2 dose required | 4 week interval | 12 - above age group
5.EC -- 1 dose required | none week interval | 18 - above age group
Pls enter the vaccine code of patient : 1

1.Vaccine Administration
2.Search Patients Record
3.Edit the Vaccines
4.View Statistical Information
5.Log Out

Pls enter the selection :
```

Figure 246: output for exit administration feature when key in vaccine code

For the last feature in the vaccine administration is the user can **exit the vaccine administration page** in any key in column by **keying in "1" in the current column** like the figure above.

6.0 Conclusion – Ho Feng Sheng

This python project is designed for the vaccination center in Malaysia as the COVID-19 pandemic is going haywire. Our group has put a lot of effort into developing and debugging this project to provide user-friendly features that can make it easier for people of all ages to use it. This python project helps to manage the COVID-19 outbreak in the country.

People can register and check their vaccination dates through this python project; however, administrators can use this application to calculate the number of people vaccinated for statistics to be submitted to the government to know how many people have been vaccinated in total. These features can greatly help Malaysia to control the outbreak more effectively.

We all hope that through our development project, we can help Malaysia to come out of the pandemic faster, and we also hope that we can work together to complete the vaccination as soon as possible so that the virus will not spread easily. Although we have completed our project, in the future, we will still add or update more features according to the trend of the pandemic so that this project can be used for a long time.

7.0 References

Mao, V. (2020, November 24). *A Very Easy Tutorial to Learn Python Regular Expression (regex)*. Retrieved from Towards Data Science: https://towardsdatascience.com/a-very-easy-tutorial-to-learn-python-regular-expression-re-c42fbbc01ef2

Pankaj. (2022, August 4). *Python Trim String - rstrip(), lstrip(), strip()*. Retrieved from Digital Ocean: https://www.digitalocean.com/community/tutorials/python-trim-string-rstrip-lstrip-strip

Programiz. (n.d.). *Python Variables, Constants and Literals*. Retrieved from Programiz: https://www.programiz.com/python-programming/variables-constants-literals

S, R. A. (2022, October 21). *Break in Python: A Step by Step Tutorial to Break Statement*. Retrieved from simplelearn: https://stackoverflow.com/questions/65005066/how-to-read-and-write-multiple-lines-of-a-text-file-in-python

w3schools. (n.d.). *Python del Keyword*. Retrieved from w3schools: https://www.w3schools.com/python/ref_keyword_del.asp#:~:text=The%20del%20keyword%20is%20used,parts%20of%20a%20list%20etc.