

Table of Contents

1.0 Introduction.....	4
2.0 Assumptions.....	5
3.0 System Design	6
3.1 Use Case Diagram.....	6
3.2 Class Diagram.....	7
4.0 Concepts and Justification	8
4.1 Variable.....	8
4.2 Control Structure.....	8
4.2.1 If.....	9
4.2.2 If-else	9
4.2.3 Nested if-else	10
4.2.4 Switch Case.....	10
4.3 Looping Structure	11
4.3.1 While.....	11
4.4 Exception Handling	11
4.4.1 IOException	12
4.4.2 ParseException.....	12
4.5 Overriding.....	13
4.6 Object Oriented Concepts	13
4.6.1 Inheritance.....	14
4.6.1.2 Multiple Inheritance.....	15
4.6.2 Polymorphism.....	15
4.6.2.1 Static Polymorphism.....	16
4.6.2.2 Dynamic Polymorphism	16
4.6.3 Encapsulation.....	18
5.0 Sample Output Screen.....	19
5.1 Patients/People.....	20
5.2 Personnel.....	26
6.0 Additional Features	32
6.1 String[]	32
6.2 equals().....	32

6.3 split()	32
6.4 Arrays.stream().collect(Collectors.joining()).....	32
6.5 length.....	32
Conclusion	33
Work Load Matrix	33
Appendix	34
References.....	37

1.0 Introduction

In 2019 the world underwent a virus outbreak known as COVID-19, the virus became so serious we had to close off borders. Due to this pandemic many countries rushed to create a vaccine that could protect our society from the Virus. Now the number of vaccines developed by different countries have grown and we began to lose count. My team have been tasked in developing a system for the country in order to create an easy and effective way to distribute the vaccine to the people, citizen or non-citizen, of Malaysia.

The system consisting of five entities, Personnel, People, Appointment, Vaccination Centre and the Vaccine itself. Tasked to develop and establish a connection between these objects to simulate and produce a registration system. Also assigned design and produce the system with a graphical interface whereby it is easily understood and easily navigable for any user of the system. Resulting in a smooth and fluid processing of the registration system.

With this information and the knowledge learned we are to implement Object Oriented Concepts with Java Programming into this system in order to create in fluid system and to showcase what my team and I have learned.

2.0 Assumptions

General

1. Users must register for an account.
2. There are only 3 kinds of vaccines.
3. There are only 3 vaccine Centre locations.

People

1. People can only either be a “Citizen” or a “Non-citizen”
2. People can only apply for maximum two appointments.
3. People can only receive one kind of vaccine.
4. People can register for vaccination appointment.
5. People can view vaccination status.
6. People will set the appointment date.

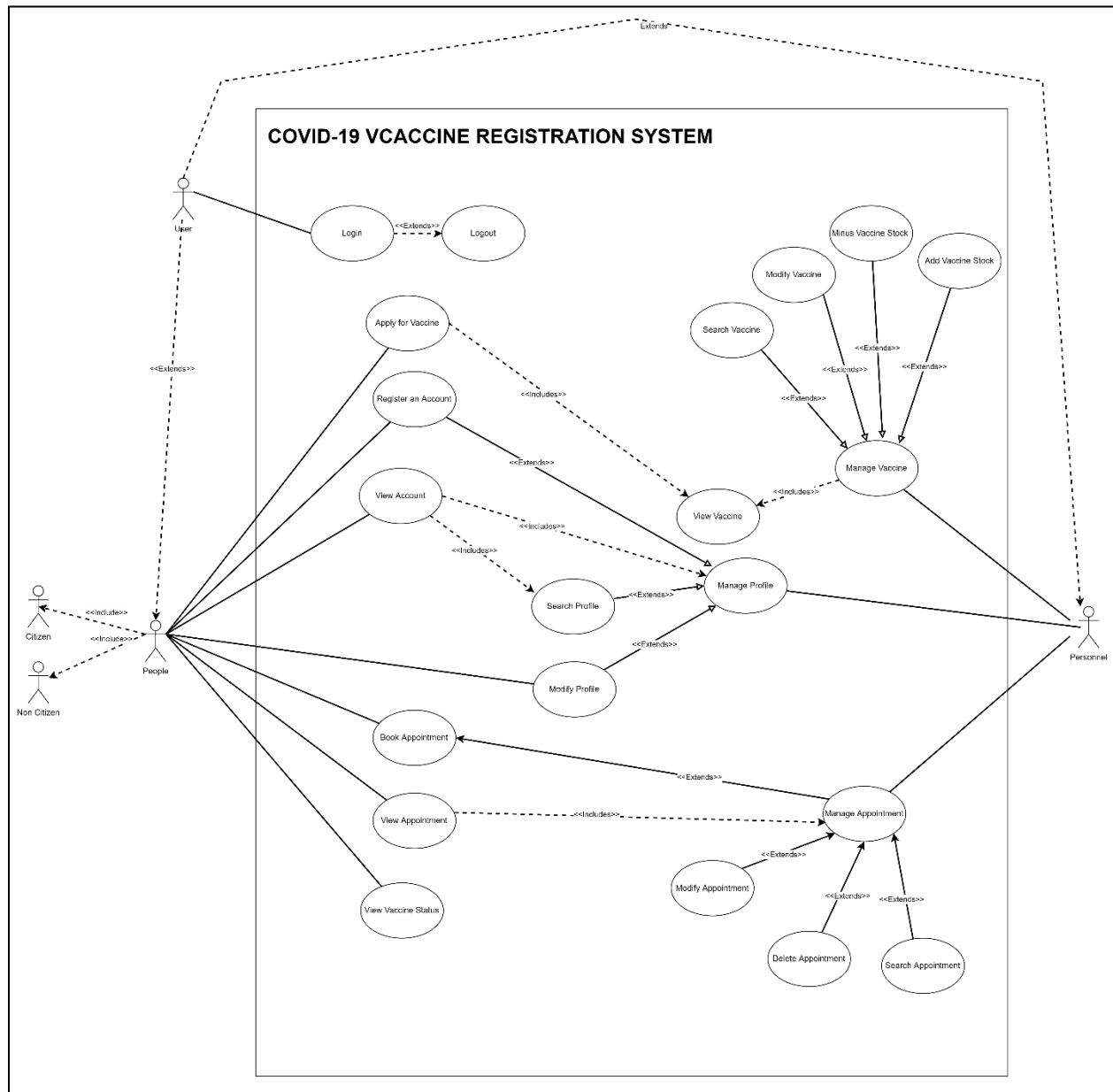
Personnel

1. Only the personnel can change the status of the vaccine of users
2. Personnel already have an account.
3. Personnel can manage people’s data.
4. Personnel can manage vaccination appointments.
5. Personnel can manage vaccines supplies.
6. Personnel can view people’s data.
7. Personnel can view vaccination appointments.
8. Personnel can view vaccine supplies.

3.0 System Design

3.1 Use Case Diagram

The Use case Diagram is a simple UML Diagram that primarily display the system requirements and what each person has the ability to access within the system. It's a summary to showcase the relationship between use cases and the actors inside the system. Each actor has different access within the system and the use case gives a clear view of what each actor is capable of doing.

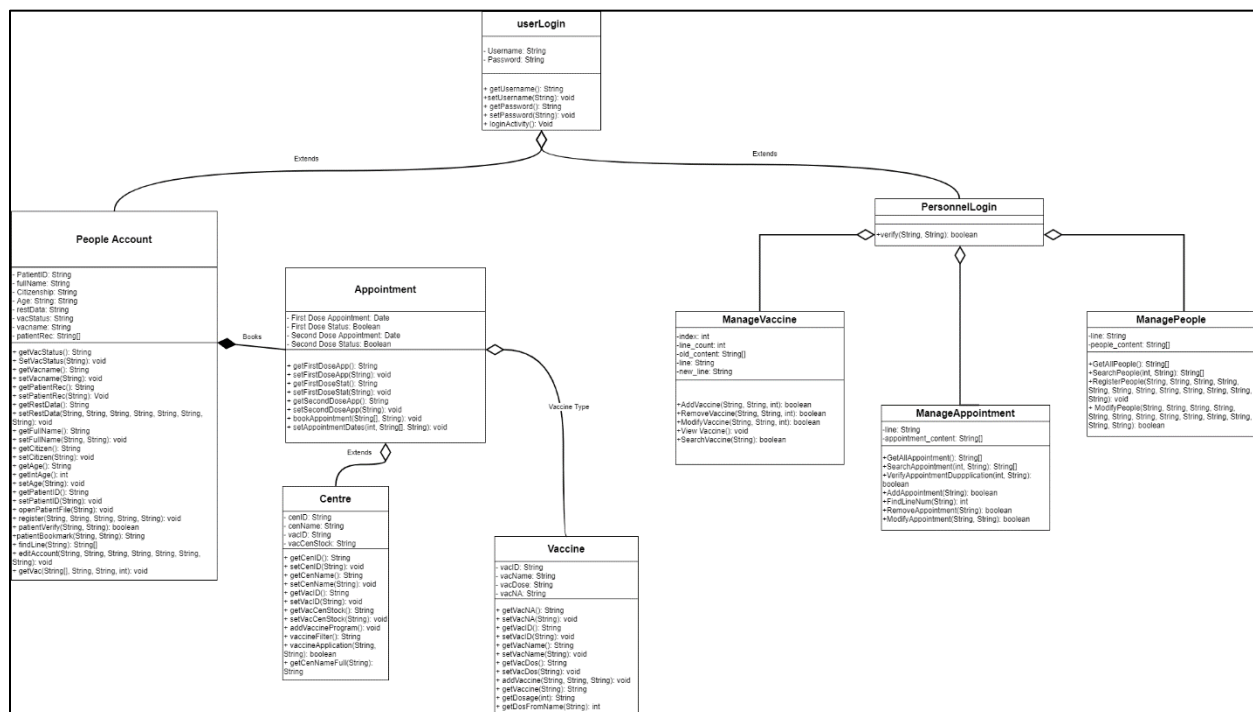


In the Diagram above there are 5 Actors and 21 Use Cases, Both the Admin and People actors are extension of the User Actor. People consists of both Citizen and Non-Citizen, meanwhile the Admin consists of Personnel. The admin has the ability to access most of the system, which consists of mostly Data Management and Manipulation whereby the People Actor are only cable of a handful of functionalities within the system that affects their own account.

The admin account could manage, by the word of manage it has the ability and includes the ability to view, add, modify and some extra functions. This allows the admin to have a stronger control over the data. Meanwhile people are only able to access some of the functions of that manage thus only connecting them to the functions they can access.

3.2 Class Diagram

The class Diagram is the other type of UML Diagram used to help the team design the system. The Main point of the Class Diagram is to showcase what each class has such as their attributes, methods, and their relationship with other objects within the system. It is a static Structure Diagram



(Bigger Figure in Appendix)

Our system consists of 9 Classes. The classes are consists of userLogin, peopleLogin, PersonnelLogin, appointment, centre, vaccine, ManageVaccine, ManageAppointment and ManagePeople. Each of the classes consisting of their own methods and how they are connected to the other classes.

4.0 Concepts and Justification

4.1 Variable

```
private String fullName, Citizenship, age, patientID, restData, vacStatus, vacname;  
private String[] patientRec;
```

A Variable in a programming language is used to hold some kind of data or information. Once data is stored in it, we are then able to call it later on and use it in different areas of our code. In Java Object Oriented Programming, a variable needs to be declared first, most importantly with a data type. A variable cannot be declared without a datatype. In the case above, fullName, Citizenship, age, etc are declared as String Variables. Meanwhile patientRec is declared as String array Variable.

There are two types of data types that a variable can be declared into, these include Primitive data types and Non-Primitive data types. Primitive data type include, byte, short, int, long, float, double, Boolean and char. Meanwhile on the other hand, non-primitive data type are data type like, String, Arrays. The difference between the two data types are that non-primitive data types can be Null, meanwhile primitive data types require there to be a value. Furthermore primitive datatypes have dynamic size while non-primitive have static size.

4.2 Control Structure

One of the fundamental concept of programming is the control structure. Is used to specify or control the flow of the system. allowing directing the system to other parts of the code if it meets a true in a certain argument. A control structure is made of one or multiple arguments to question the system on whether one part is true or false and thus directing them one way or the other. There are multiple control structures within java these include, If, else-if, Nested-if-else, switch case, etc.

4.2.1 If

```
if(username.equals(brokenLine[11]) == true && password.equals(brokenLine[12]) == true) {  
  
    return true;  
  
}
```

The If control structure, is an argument inside of coding that uses Boolean statements to determine whether it enters the if statement or not. If true, it does otherwise it passes to the next statement outside of the if function. In the case above, if username equals to the value of brokenLine[11] and password equals to brokenLine[12] is true, then enter and return true.

4.2.2 If-else

```
if(log.verify(log.getUsername(), log.getPassword())) {  
  
    System.out.println("Welcome");  
    log.setPatientID(log.patientBookmark(log.getUsername(), log.getPassword()));  
    log.loginActivity(log.getUsername());  
  
    setVisible(false);  
    patientmainmenu pmain = new patientmainmenu(log.getPatientID(), "A");  
    pmain.setVisible(true);  
  
} else{  
  
    System.out.println("Username and Password are incorrect");  
  
}
```

The If-else statement is the next step of an if statement. Whereby the if the argument in the if statement is false, it will pass to the next statement. But if the if statement has an added Else statement. If the argument is false, it enters the Else statement before exiting outside of the if statement.

4.2.3 Nested if-else

```
if(log.getVacname().equals("NA") == true) {  
  
    if(cen.vaccineApplication(vacidStr, locidStr) == true) {  
  
        String vacname = vac.getVaccine(vacidStr);  
        int vacdos = vac.getDosage(vacidStr);  
  
        String cenname = cen.getCenNameFull(locidStr);  
  
        System.out.println("Entered");  
        log.getVac(log.getPatientRec(), vacname, cenname, vacdos);  
  
        patientmainmenu frame = new patientmainmenu(log.getPatientID(), "D");  
        setVisible(false);  
        frame.setVisible(true);  
  
    } else {  
  
        System.out.println("We have encountered an error applying for this vaccine");  
  
    }  
  
} else {  
  
    System.out.println("You have already applied for a vaccine");  
  
}
```

The Nested If-Else is in simple terms an if-else function within another if-else function. Nested If-Else does not have a maximum number where a programmer can add. Combining these two functions makes it more secure and are able to determine better where the code enters to.

4.2.4 Switch Case

```
switch(currentPage) {  
  
    case "A":  
        jTabbedPanel.setSelectedIndex(0);  
  
    case "B":  
        jTabbedPanel.setSelectedIndex(1);  
  
    case "C":  
        jTabbedPanel.setSelectedIndex(2);  
  
    case "D":  
        jTabbedPanel.setSelectedIndex(3);  
  
    default:  
        jTabbedPanel.setSelectedIndex(1);  
  
}
```

The Switch Case is similar to a nested if else, If the value in the argument matches with one of the cases within the switch case, it will execute all the lines within that case only. Here the argument lies within the variable *currentPage* the switch case looks at the value of *currentPage* if it matches any of the cases. Such as "A", "B", "C", "D" if none matches, it will automatically execute the default. Otherwise

4.3 Looping Structure

Looping Structure is one of the fundamental programming concepts. The Looping concept simply describes a sequence of activity that is repeated again and again until meeting a result to break off the loop. Loops provide a simple way for a programmer to repeat a certain number of code multiple times without writing the code multiple times. The loop structure works by using an argument and asking every iteration until that argument is false. There are multiple types of Loops in Java, these include, while, for, etc.

4.3.1 While

```
while (Sc.hasNextLine()) {  
    Sc.nextLine();  
    count++;  
}
```

The While Function is a looping structure function that operates in a way similar to an if, except an if only operates once, a while operates until the argument becomes false. Thus looping into it again and determining true or false, if true, repeat and if false, exit. IN this case, the while loops argument is if the variable sc has a next line, if it does then proceed with the block of code.

4.4 Exception Handling

The Exception handling from Java is an incredibly powerful mechanism that is used to handle runtime errors. When the program is running and it encounters an error, instead of breaking the system, it can catch the error and continue the program as normal. There are multiple types of Exception Handling in Java, due to the many forms of runtime errors when creating a program many different types of Exception handling has to be used. Some of these include, IOException Handling, ParseException, ClassNotFoundException, etc.

Using the main keywords “Try” and “Catch” These two keywords create what is known as the Try-Catch Block. The Try-Catch Block is one of the exception Handling methods in java. Used within a method if a block of code might throw an exception the try catch block can catch the error. If caught, it permits the code on executing the rest of the code. Simply, in the code, the system will Try the block of code within the try block, then if an error were to occur, it will enter the catch and catch the error.

4.4.1 IOException

```
try {

    FileWriter loginfile = new FileWriter("src/txt/loginactivity.txt", true);

    Date date = new Date();
    SimpleDateFormat dmy = new SimpleDateFormat("dd/MM/yy");
    SimpleDateFormat time = new SimpleDateFormat("HH:mm");
    String currentDate = dmy.format(date);
    String currentTime = time.format(date);

    loginfile.write(currentDate + "\t" + currentTime + "\t" + username + "\n");

    loginfile.close();

} catch (IOException e) {
    System.out.println("An Error has occured");
    e.printStackTrace();
}
```

The IOException is an exception in java that occurs when an Input-Output (IO) operation causes an error. Usually occurring when handling files. Operations that involve a File such as Reading, Writing and etcetera.

4.4.2 ParseException

```
String fddateString = app.getFirstDoseApp();
try {

    Date fddate = dmy.parse(app.getFirstDoseApp());

    if(d2.compareTo(fddate) > 0) {
        app.setAppointmentDates(2, log.getPatientRec(), currentDate);
        app.bookAppointment(log.getPatientRec(), currentDate);
    } else {

        System.out.println("Please choose a date above the date of your first appointment");

    }

} catch (ParseException e) {
    e.printStackTrace();
}
```

ParseException is another exception that may occur in Java during runtime. The ParseException similarly like IOException handles exception that involves the parse() method. The parse method is used to extract data from an input and it converts it into the necessary information to form an object of the calling class. In the case above, we extract a date type and converts it into a dmy format.

4.5 Overriding

```
public boolean verify(String username, String password) {  
    return false;  
}
```

```
@Override  
public boolean verify(String username, String password) {...32 lines }
```

Overriding is one of abilities of java where a function with the same name is used. An annotation used to promote Polymorphism. One class may have a function with the same name but that method is different from the subclass. The subclass can then Override the superclass method and change the way they use the method.

4.6 Object Oriented Concepts

The object-oriented programming style (OOP) is a type of programming language that uses “Objects” in order to give them attributes and fields and methods that each *object* of the same type can do. An object in this programming style is representing an entity, of physical item, conceptual or a software type. Enclosed in a state and given a behaviour we can differentiate each object by their “Attributes” but are all able to “Behave” the same way.

The basic Object-Oriented Concepts that define what OO Programming is defined by 4 Concepts, Inheritance, Polymorphism, Abstraction and Encapsulation.

4.6.1 Inheritance

Inheritance by definition stands for the process of passing on information genetically from parent to child. Hence it is normal for family members to have similar characteristics and traits. In the case of Object-Oriented Programming, Inheritance is the process of passing on behaviours and attributes from parent class to child class. This will result in the child class also known as the subclass to share the same set of states and methods that the superclass, parent class.

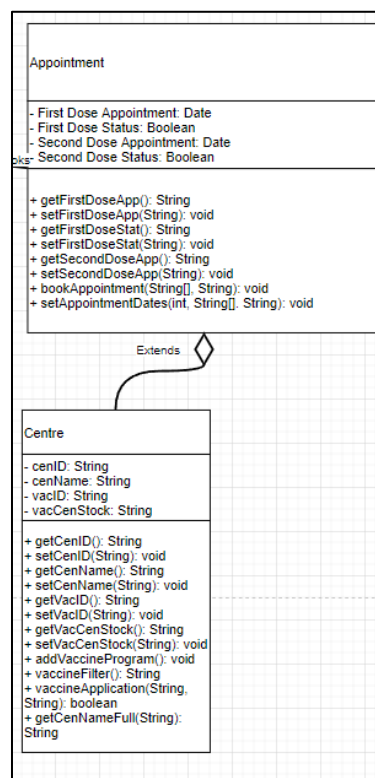
```
public class peopleLogin extends userLogin {
```

A child class can only inherit from one parent class by the use of the keyword *extends*, but a single parent class can share to multiple child class. There are two types of Inheritance, single inheritance and multiple inheritance.

4.6.1.1 Single Inheritance

As previously mentioned, Single Inheritance simply means that ONE parent has ONE child. A single superclass is sharing its methods with the subclass, resulting in the subclass being able to access the parent's. In the figure below, the Appointment class is an extension of the centre class.

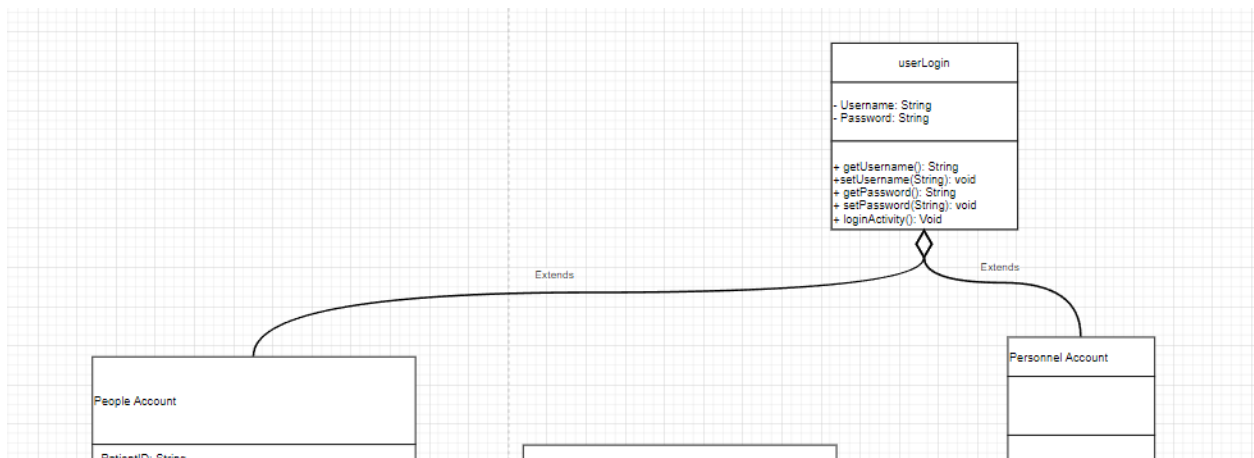
```
public class appointment extends centre{
```



4.6.1.2 Multiple Inheritance

On the other hand, Multiple Inheritance, is an Inheritance where the superclass is sharing its methods with MANY subclass. This is the only way Multiple Inheritance is achieved.

```
public class PersonnelLogin extends userLogin{  
  
public class peopleLogin extends userLogin {
```



In the case above the class userLogin passes down its attributes and methods to the subclass which are peopleLogin and personnelLogin. We can see it in each class declaration that both of them “extends userLogin”, allowing for both classes to use methods and attributes from the superclass.

4.6.2 Polymorphism

Object-Oriented Programming also has the concept known as Polymorphism. In the words of Thorben Janssen, Polymorphism describes a situations in which something occurs in similarly but in different forms. To clarify in terms of the code, its when one function is used but is used differently in same/separate classes. The separate classes using the same function uses it differently on its own, therefore this function is polymorphous.

4.6.2.1 Static Polymorphism

Static Polymorphism is a type of polymorphism concept, in this type, there are multiple methods of the same class that also uses the same name but has different set of parameters. Also known as method overloading. In the figure below, there are two methods named `setFullName()` but one has two parameters and the other only has one. Depending on the number of parameters when the method is called will decide which method to be used.

```
public void setFullName(String firstName, String lastName) {  
    this.fullName = firstName + " " + lastName;  
}  
  
public void setFullName(String fullName) {  
    this.fullName = fullName;  
}
```

4.6.2.2 Dynamic Polymorphism

The other type of the polymorphism concept is Dynamic Polymorphism. The key difference between Static Polymorphism and Dynamic Polymorphism is, in Dynamic the polymorphous method is placed in a inheritance hierarchy. A method found in its parent can be overridden in the child class and this method becomes polymorphous.

`verify()` method from `userLogin` class

```
public boolean verify(String username, String password) {  
    return false;  
}
```

Verify() method from peopleLogin class

```
@Override
public boolean verify(String username, String password) {

    //
    try {

        File patientfile = new File("src/txt/patientdata.txt");
        FileReader fr = new FileReader(patientfile);
        BufferedReader br = new BufferedReader(fr);
        String line;

        while((line=br.readLine()) != null) {

            String[] brokenLine = line.split("\t");

            if(username.equals(brokenLine[11]) == true && password.equals(brokenLine[12]) == true) {

                return true;

            }

        }
        fr.close();

    } catch (IOException e) {

        System.out.println("An Error has occurred");

        e.printStackTrace();

    }
    return false;

}
```

Verify() method from personnelLogin class

```
@Override
public boolean verify(String username, String password) {

    try {

        File PersonnelFile = new File("src/txt/PersonnelData.txt");
        FileReader fr = new FileReader(PersonnelFile);
        BufferedReader br = new BufferedReader(fr);
        String line;
        int found = 0;

        while((line=br.readLine()) != null) {

            String[] brokenLine = line.split("\t");
            if(username.equals(brokenLine[1]) == true && password.equals(brokenLine[4]) == true) {

                userLogin ul = new userLogin();
                ul.loginActivity(username);
                found += 1;
                fr.close();
                return true;

            }

        }
        fr.close();
        if (found == 0){
            System.out.println("Username or Password is incorrect.\nPlease try again.\t");
        }

    } catch (IOException e) {

        System.out.println("An Error has occurred");

    }
    return false;

}
```

From the figure above. We see three classes with the same function. userLogin, peopleLogin and personnelLogin class have the function verify. The superclass being userLogin is passing down the method of verify() but both the subclass, peopleLogin and personnelLogin overrides the method to suit their needs.

4.6.3 Encapsulation

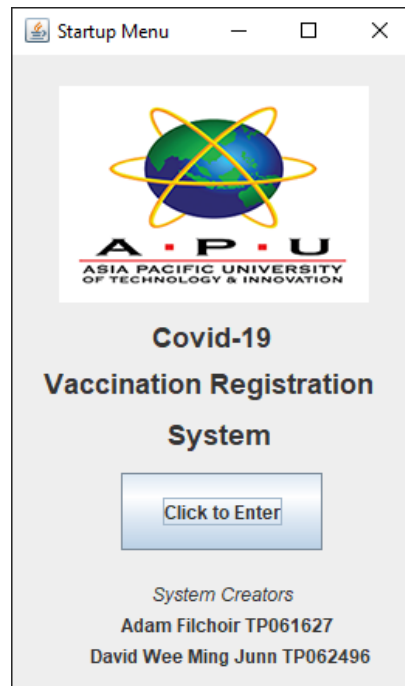
The Encapsulation Concept in Object Oriented is the concept of “Hiding” data inside of a single unit. These data are meant by the variables/characteristic that define that object. These variables will be hidden from other classes unless when called using their specific getters. A declared variable is hidden when added with the keyword *Private*. In the figure below, two String Variables is declared privately in this class.

```
private String username, password;
```

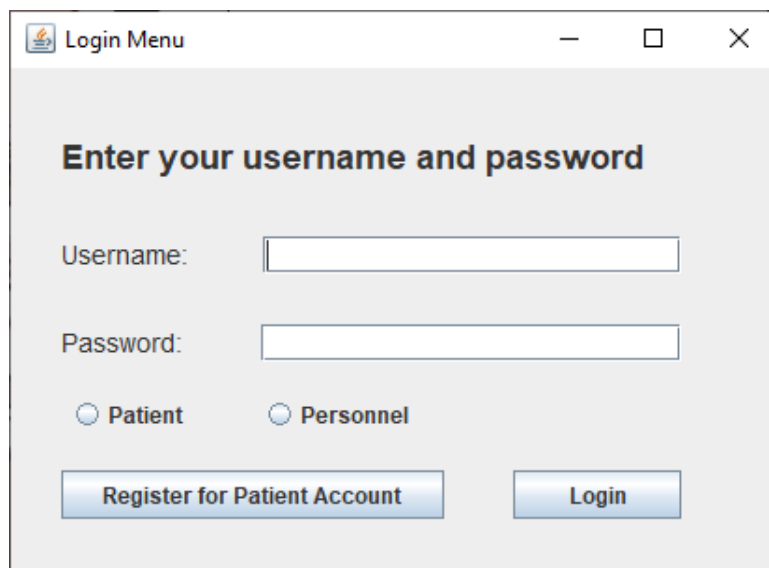
However, having just the keyword Private does not make it encapsulated, the data is hidden but there is no way to access it outside of the class and thus, we use getters and setters to achieve this. Completing the concept of Encapsulation. Each variable that was declared in the above figure has its getters and setters in the figure below.

```
public String getUsername() {  
    return username;  
}  
  
public void setUsername(String username) {  
    this.username = username;  
}  
  
public String getPassword() {  
    return password;  
}  
  
public void setPassword(String password) {  
    this.password = password;  
}
```

5.0 Sample Output Screen

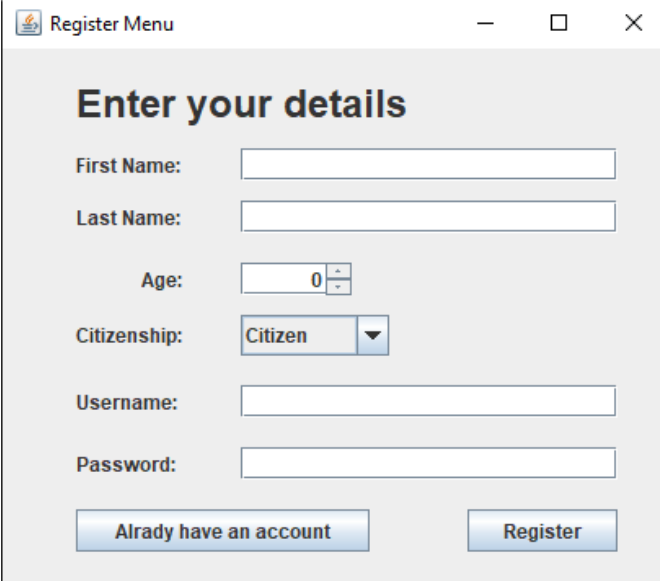


The start menu of the system. Only functions as a menu to begin the rest of menus. Displays the APU Logo along with the system creators.



The Login page that comes after the start menu. The users, both the Admin and the Patient both goes through this login menu. Differentiated by using the radio buttons. Only allows enter if Both Username and Password matches in the txt file of the respective patient or personnel buttons. Also allows to redirect to register menu if patient Does not have account.

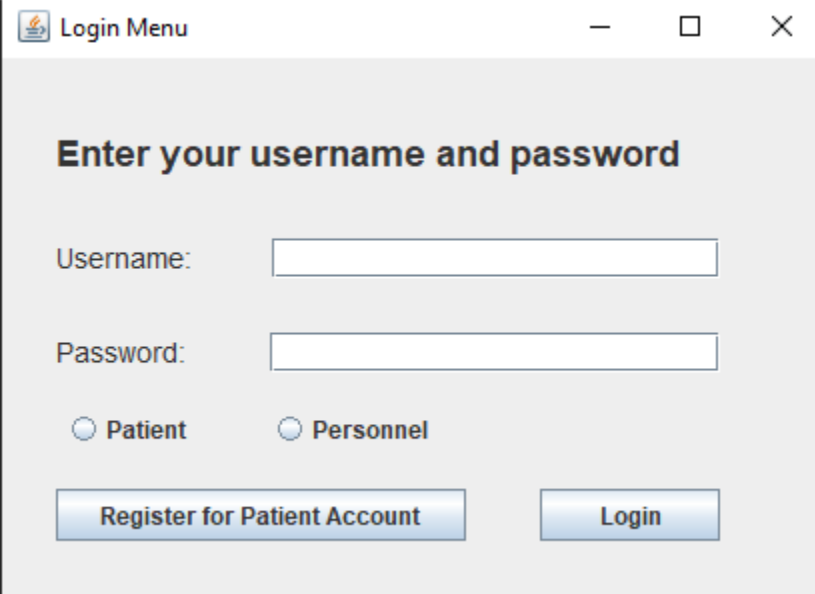
5.1 Patients/People



The 'Register Menu' window is a Java Swing application window titled 'Register Menu'. It contains a form titled 'Enter your details'. The form has the following fields and controls:

- First Name:** A text input field.
- Last Name:** A text input field.
- Age:** A numeric input field with a value of 0 and increment/decrement buttons.
- Citizenship:** A dropdown menu currently showing 'Citizen'.
- Username:** A text input field.
- Password:** A text input field.
- Buttons:** Two buttons at the bottom: 'Already have an account' and 'Register'.

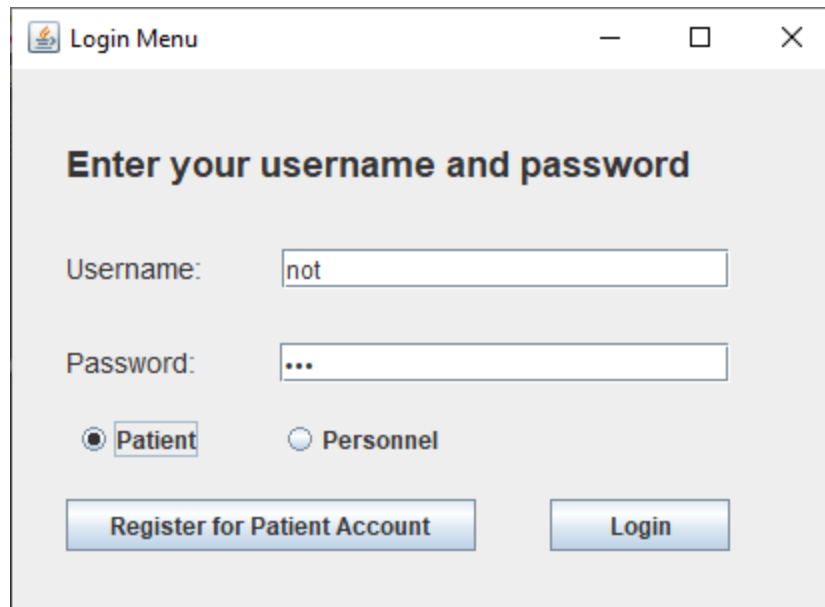
Register menu of patients. For patients to register, they need their First Name and Last name to make their full name, their age, citizenship which includes only Citizen or Non-Citizen, lastly their username and password to enter the system.



The 'Login Menu' window is a Java Swing application window titled 'Login Menu'. It contains a form titled 'Enter your username and password'. The form has the following fields and controls:

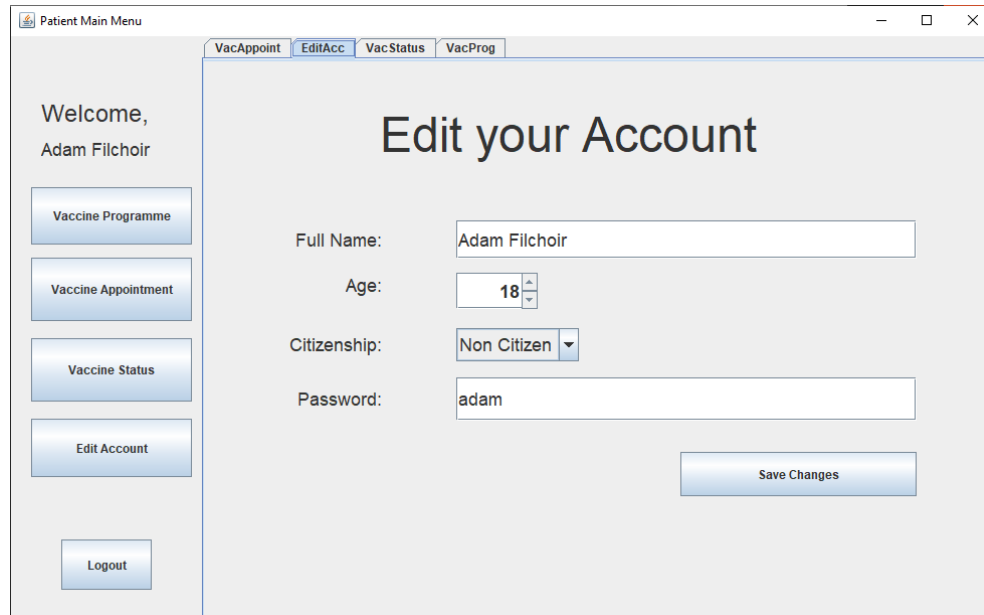
- Username:** A text input field.
- Password:** A text input field.
- Radio Buttons:** Two radio buttons labeled 'Patient' and 'Personnel'.
- Buttons:** Two buttons at the bottom: 'Register for Patient Account' and 'Login'.

Once Registered, or when clicking 'Already have an account' the system will redirect back to the login menu. Due to once registered does not mean it will directly login into the system. And for security purposes they must enter their username and password to login.



The image shows a window titled "Login Menu". It contains a heading "Enter your username and password". Below this, there are two input fields: "Username:" with the text "not" and "Password:" with three dots. There are two radio buttons: "Patient" (selected) and "Personnel". At the bottom, there are two buttons: "Register for Patient Account" and "Login".

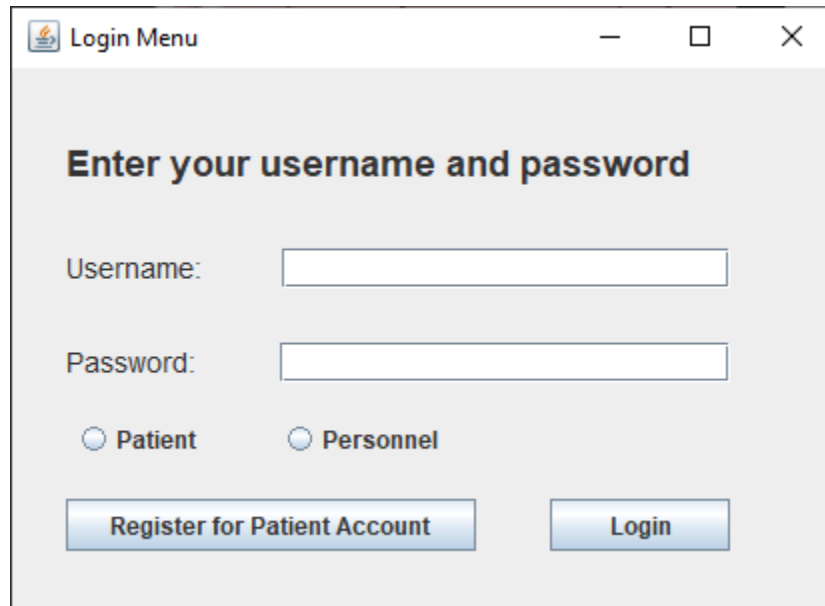
The figure above displays a user entering their username and password correctly and choosing the type of account to enter the system. If the username matches with the file from the patient registry and the password also does the same. It allows to login.



The image shows a window titled "Patient Main Menu". It has a sidebar on the left with buttons: "Vaccine Programme", "Vaccine Appointment", "Vaccine Status", "Edit Account", and "Logout". The main area has a heading "Edit your Account" and a form with fields: "Full Name:" (Adam Filchoir), "Age:" (18), "Citizenship:" (Non Citizen), and "Password:" (adam). There is a "Save Changes" button at the bottom right. The window also has tabs at the top: "VacAppoint", "EditAcc" (selected), "VacStatus", and "VacProg".

When the user correctly enters their username and password and the system recognizes both. When entering the system It will directly move to the edit account menu of the patient menu. On the left side of the system, the system welcomes the user into the system with their full name, furthermore it has the menu tabs that allows for navigation. And the logout button if the user were to log out of the system.

In this screen the user are able to edit their full name, age and citizenship and their password. The user is not able to change their username. Once edited to the way they want. They are able to save their changes by clicking the save changes button.



The image shows a Java Swing window titled "Login Menu". The window has a standard title bar with a minimize button, a maximize button, and a close button. The main content area has a light gray background. At the top, the text "Enter your username and password" is displayed in a bold, black font. Below this, there are two text input fields. The first is labeled "Username:" and the second is labeled "Password:". Below the password field, there are two radio buttons. The first is labeled "Patient" and the second is labeled "Personnel". At the bottom of the window, there are two buttons: "Register for Patient Account" and "Login".

If the user were to log out of the system it will redirect back to the user login menu allowing for another user to login into the system again.

The screenshot shows a web application window titled "Patient Main Menu". The left sidebar contains a welcome message "Welcome, Adam Filchoir" and five buttons: "Vaccine Programme", "Vaccine Appointment", "Vaccine Status" (which is selected), "Edit Account", and "Logout". The main content area has a tabbed interface with tabs for "VacAppoint", "EditAcc", "VacStatus" (selected), and "VacProg". The "VacStatus" tab displays the title "Vaccination Status" and four input fields: "Status:" with the value "Not Vaccinated", "Vaccine:" with the value "NA", "Dose 1 Appointment:" with the value "NA", and "Dose 2 Appointment:" with the value "NA". A "Refresh" button is located at the bottom right of the main content area.

If a new registered user were to login they would have no vaccine applied to their account and therefore no dose 1 or 2 appointment. Resulting their status to be Not Vaccinated. In this page, The user is able to refresh the page to refresh their data if data was altered but are not shown in the page.

The screenshot shows the same web application window, but the "VacProg" tab is selected. The main content area displays the title "Apply to get vaccinated". Below the title is a table with vaccine stock information:

Location ID	Location Name	Vaccine ID	Vaccine Stock
BJ	Bukit Jalil	PF	19900
BJ	Bukit Jalil	SV	10000
BJ	Bukit Jalil	CS	10200
HKL	Hospital Kuala Lu...	PF	20013
HKL	Hospital Kuala Lu...	SV	4296
HKL	Hospital Kuala Lu...	CS	9927

Below the table are two dropdown menus: "Location ID:" with "BJ" selected and "Vaccine ID:" with "PF" selected. A large "Apply for Vaccine" button is positioned at the bottom of the main content area.

The first thing the user should be doing is to apply for a vaccine that is located in the vaccine programme menu by clicking the vaccine programme menu button. In This page we are shown all of the Locations and their Location Id along with the vaccine that is provided by them and the amount of stock that location has of that vaccine. Here the user can apply for a vaccine by using the combo box and selecting the centre and what vaccine they would like to apply to. The user is unable to change their vaccine if already applied for one previously.

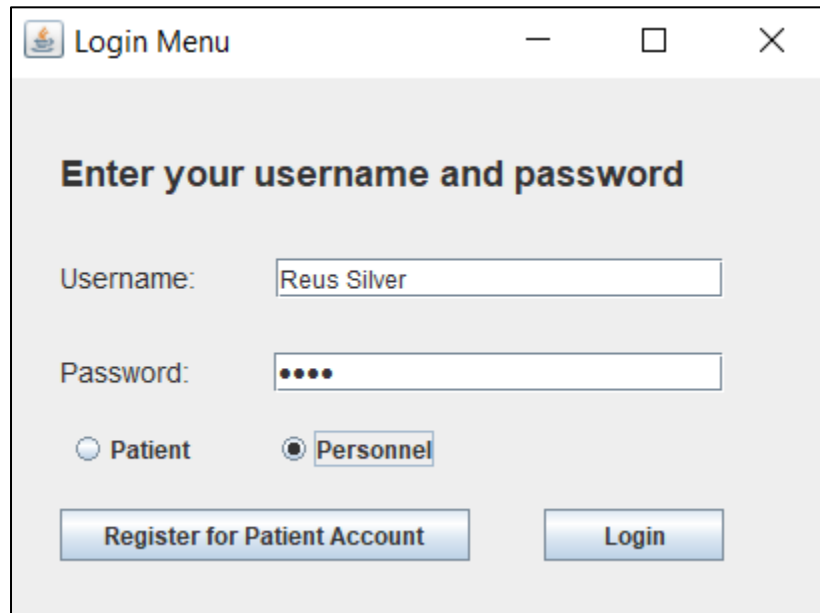
The screenshot shows a web application window titled "Patient Main Menu". It has a sidebar on the left with a "Welcome, Adam Filchoir" message and buttons for "Vaccine Programme", "Vaccine Appointment", "Vaccine Status", "Edit Account", and "Logout". The main content area has tabs for "VacAppoint", "EditAcc", "VacStatus", and "VacProg". The "VacAppoint" tab is active, displaying the "Book an Appointment" page. This page contains two date input fields, both showing "12/4/21", each with a calendar icon. To the right of these fields are buttons for "Book 1st Appointment", "Book 2nd Appointment", and a "Refresh" button.

Once the user have chosen a vaccine and has applied for one, they are able to book an appointment. Depending on the number of doses they are able to book that number of appointment. The maximum doses a vaccine has is 2. A user can only book the 2nd appointment if they have already booked for the 1st appointment and the vaccine allows them to.

The screenshot shows a web application window titled "Patient Main Menu". It features a sidebar with a welcome message "Welcome, Adam Filchoir" and four buttons: "Vaccine Programme", "Vaccine Appointment", "Vaccine Status", and "Edit Account". At the bottom of the sidebar is a "Logout" button. The main content area has a tabbed interface with four tabs: "VacAppoint", "EditAcc", "VacStatus" (which is active), and "VacProg". The "VacStatus" tab displays the title "Vaccination Status" and four input fields: "Status:" with the value "Not Vaccinated", "Vaccine:" with the value "Pfizer", "Dose 1 Appointment:" with the value "23/12/21", and "Dose 2 Appointment:" with the value "30/12/21". A "Refresh" button is located at the bottom right of the main content area.

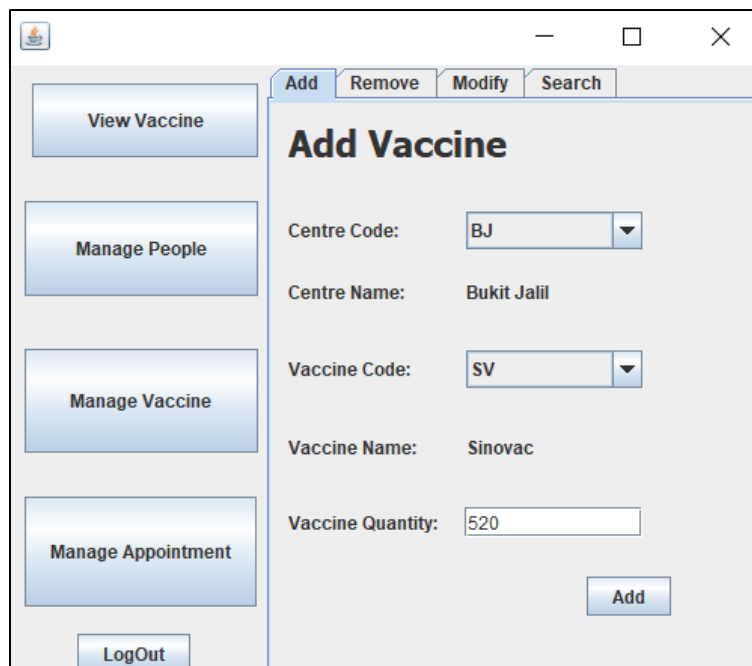
Once all the data the user has chosen when returning to the vaccine status and clicking refresh, the data will appear to the user.

5.2 Personnel



The Login Menu window has a title bar with a standard icon, a minus button, a maximize button, and a close button. The main content area has a light gray background. At the top, it says "Enter your username and password" in bold. Below this, there are two text input fields. The first is labeled "Username:" and contains the text "Reus Silver". The second is labeled "Password:" and contains four dots. Below the password field, there are two radio buttons. The first is labeled "Patient" and is unselected. The second is labeled "Personnel" and is selected. At the bottom, there are two buttons: "Register for Patient Account" and "Login".

To login as a personnel, the personnel box must be checked and the username and password must be identical to the ones in the data.



The Add Vaccine window has a title bar with a standard icon, a minus button, a maximize button, and a close button. The main content area has a light gray background. On the left side, there is a vertical sidebar with four buttons: "View Vaccine", "Manage People", "Manage Vaccine", and "Manage Appointment". At the bottom of the sidebar is a "LogOut" button. The main area has a tabbed interface with four tabs: "Add", "Remove", "Modify", and "Search". The "Add" tab is selected. The "Add Vaccine" form has four fields: "Centre Code:" with a dropdown menu showing "BJ", "Centre Name:" with the text "Bukit Jalil", "Vaccine Code:" with a dropdown menu showing "SV", and "Vaccine Name:" with the text "Sinovac". Below these fields is a "Vaccine Quantity:" field with the text "520". At the bottom right of the form is an "Add" button.

Personnel will be directed to one of the three main features which is vaccine management. Through here, the personnel can view vaccine, add vaccine, remove vaccine, modify vaccine, and search vaccine. The vaccine supply can be managed through this graphical user interface(gui). The other tabs will be shown in appendix as there's not much difference in appearance.

The screenshot shows a web application interface for managing people. On the left, there are four buttons: 'View People', 'Manage People', 'Manage Vaccine', and 'Manage Appointment'. At the bottom left is a 'LogOut' button. The main area has tabs for 'Search', 'Register', and 'Modify'. Below the tabs is a table with 13 columns: ID, Citizenship, Name, Age, Status, Vac Name, F1 Date, F1 Status, F2 Date, F2 Status, Vac Location, Username, and Password. The table contains 13 rows of data. Below the table, there is a search bar with a dropdown for 'ID' and a text input field containing 'Exp: Naruto', followed by a 'Search' button.

ID	Citizenship	Name	Age	Status	Vac Name	F1 Date	F1 Status	F2 Date	F2 Status	Vac Location	Username	Password
1001	Non Citizen	Zidan	18	Not Vaccinated	CansSinoBio	NA	Not Done	Not Applicable	Not Applicable	Bukit Jalil	what	this
1002	Non Citizen	Dex Ter	3	Not Vaccinated	CansSinoBio	NA	Not Done	Not Applicable	Not Applicable	Bukit Jalil	dexy	doo
1003	Citizen		0	Not Vaccinated	Pfizer	NA	Not Done	NA	Not Done	Hospital Kuala Lumpur	no	name
1004	Non Citizen	ad	7	Not Vaccinated	Pfizer	30/11/21	Approved	22/12/21	Not Done	Bukit Jalil	not	now
1005	Citizen	Lesly Noodle	0	Not Vaccinated	Pfizer	23/12/21	Approved	NA	Not Done	Hospital Kuala Lumpur	lov	you
1006	Citizen		0	Not Vaccinated	CansSinoBio	16/12/21	Approved	Not Applicable	Not Applicable	Bukit Jalil	ally	log
1007	Non Citizen	Water bottle	3	Not Vaccinated	CansSinoBio	NA	Not Done	Not Applicable	Not Applicable	Bukit Jalil	not	mine
1008	Citizen	first last	18	Not Vaccinated	NA	NA	Not Done	NA	Not Done	NA	Boieng101	hola
1009	Citizen	wwwwww...	0	Vaccinated	Pfizer	28/11/21	Approved	28/11/21	Done	Bukit Jalil	your	rouy
1010	Citizen		0	Not Vaccinated	NA	NA	Not Done	NA	Not Done	NA	test	test
1011	Citizen	Trying~	10	Not Vaccinated	Pfizer	NA	Not Done	NA	Not Done	NA	try	try
1012	Citizen	Naruto	17	Vaccinated	Pfizer	22/11/2021	Done	11/12/21	Done	Mid Valley	Boruto	burrito
1013	Non Citizen	Gracias	24	Not Vaccinated	Pfizer	28/12/2021	Not Done	NA	Not Done	Mid Valley	Aloha	Luna

This is the second main feature which is manage the people's data. The personnel will be able to view, search and modify people's data. The personnel is also able to register people into the system.

The screenshot shows the same web application interface as the previous one, but with the 'Search' tab selected. The search bar at the bottom now has a dropdown menu set to 'Citizenship' and a text input field containing 'Citizen'. The table below shows only the rows where the citizenship is 'Citizen'.

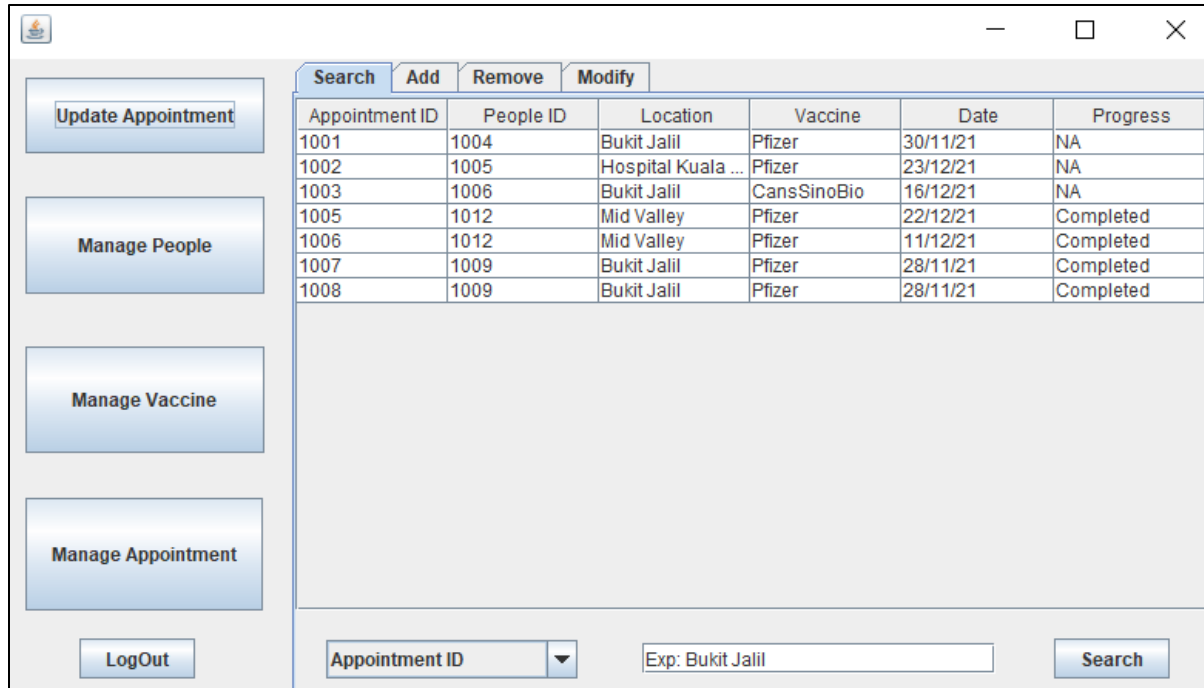
ID	Citizenship	Name	Age	Status	Vac Name	F1 Date	F1 Status	F2 Date	F2 Status	Vac Location	Username	Password
1003	Citizen		0	Not Vaccinated	Pfizer	NA	Not Done	NA	Not Done	Hospital Kuala Lumpur	no	name
1005	Citizen	Lesly Noodle	0	Not Vaccinated	Pfizer	23/12/21	Approved	NA	Not Done	Hospital Kuala Lumpur	lov	you
1006	Citizen		0	Not Vaccinated	CansSinoBio	16/12/21	Approved	Not Applicable	Not Applicable	Bukit Jalil	ally	log
1008	Citizen	first last	18	Not Vaccinated	NA	NA	Not Done	NA	Not Done	NA	Boieng101	hola
1009	Citizen	wwwwww...	0	Vaccinated	Pfizer	28/11/21	Approved	28/11/21	Done	Bukit Jalil	your	rouy
1010	Citizen		0	Not Vaccinated	NA	NA	Not Done	NA	Not Done	NA	test	test
1011	Citizen	Trying~	10	Not Vaccinated	Pfizer	NA	Not Done	NA	Not Done	NA	try	try
1012	Citizen	Naruto	17	Vaccinated	Pfizer	22/11/2021	Done	11/12/21	Done	Mid Valley	Boruto	burrito

This is a demonstration of the search function in people management. The personnel can search based on their desired criteria. In this case, citizenship is chosen, and only those who are citizens are displayed.

This is the registration feature for personnel. People's data can be inserted through here, to register them for the vaccination programme.

The small figure on the left is what the modify gui initially looks like. As you can see, all the text boxes are empty.

After writing down the people's ID and press enter, their information will show up correspondingly. This gui helps personnel to modify people's data.

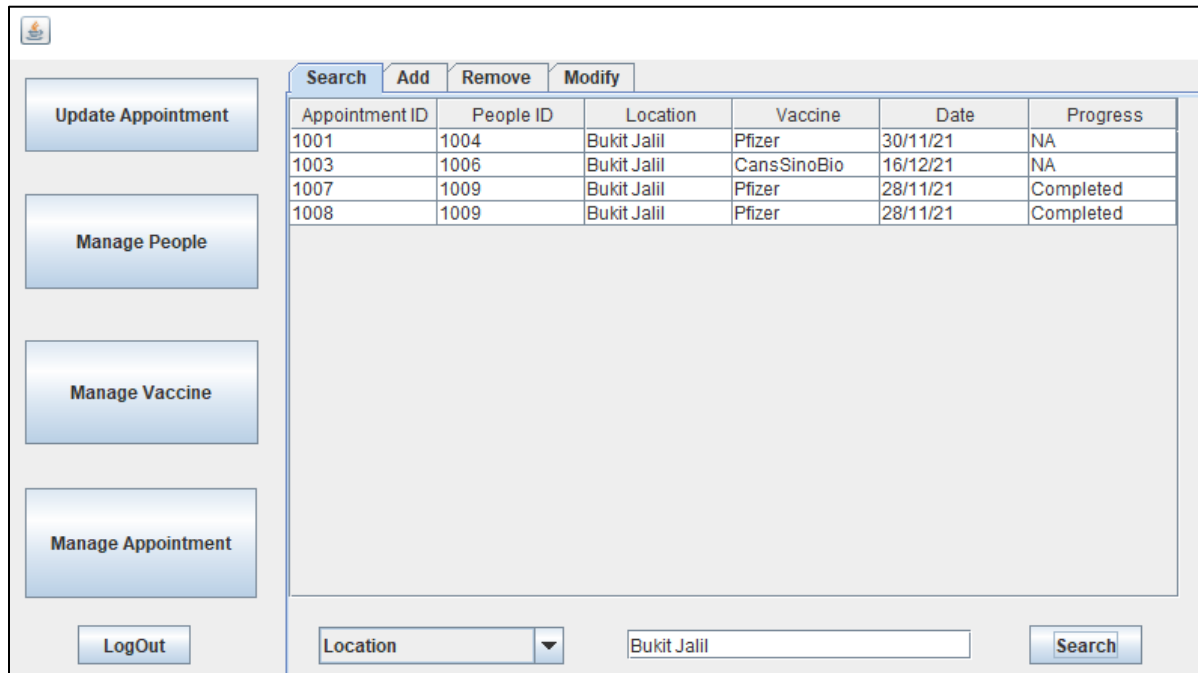


The screenshot shows a web application window titled "Appointment Management". On the left, there are four buttons: "Update Appointment", "Manage People", "Manage Vaccine", and "Manage Appointment", followed by a "LogOut" button at the bottom. The main area contains a table with the following data:

Appointment ID	People ID	Location	Vaccine	Date	Progress
1001	1004	Bukit Jalil	Pfizer	30/11/21	NA
1002	1005	Hospital Kuala ...	Pfizer	23/12/21	NA
1003	1006	Bukit Jalil	CansSinoBio	16/12/21	NA
1005	1012	Mid Valley	Pfizer	22/12/21	Completed
1006	1012	Mid Valley	Pfizer	11/12/21	Completed
1007	1009	Bukit Jalil	Pfizer	28/11/21	Completed
1008	1009	Bukit Jalil	Pfizer	28/11/21	Completed

Below the table, there is a search bar with a dropdown menu labeled "Appointment ID" and a text input field containing "Exp: Bukit Jalil". A "Search" button is located to the right of the input field.

This is the third and last main feature, appointment management. Here, the personnel can search, view, add, remove, and modify appointments.



The screenshot shows the same web application window, but the search results are filtered by location. The table now displays only appointments at "Bukit Jalil":

Appointment ID	People ID	Location	Vaccine	Date	Progress
1001	1004	Bukit Jalil	Pfizer	30/11/21	NA
1003	1006	Bukit Jalil	CansSinoBio	16/12/21	NA
1007	1009	Bukit Jalil	Pfizer	28/11/21	Completed
1008	1009	Bukit Jalil	Pfizer	28/11/21	Completed

The search bar now has a dropdown menu labeled "Location" and a text input field containing "Bukit Jalil". The "Search" button remains.

This is a demonstration of the gui's search capability. Referring to the figure above, only data which the location is "Bukit Jalil" will be displayed.

ID	Citizenship	Name	Age	Status	Vac Name	F1 Date	F1 Status	F2 Date	F2 Status	Vac Location	Username	Password
1001	Non Citizen	chuan	10	Not Vaccinat...	CansinoBio	NA	Not Done	Not Applicab...	Not Applicable	Bukit Jalil	wrat	aris
1002	Non Citizen	Dex Ter	3	Not Vaccinat...	CansinoBio	NA	Not Done	Not Applicab...	Not Applicable	Bukit Jalil	dexy	doo
1003	Citizen		0	Not Vaccinat...	Pfizer	NA	Not Done	NA	Not Done	Hospital Ku...	no	name
1004	Non Citizen	ad	7	Not Vaccinat...	Pfizer	30/11/21	Approved	22/12/21	Not Done	Bukit Jalil	not	now
1005	Citizen	Lesly Noodle	0	Not Vaccinat...	Pfizer	23/12/21	Approved	NA	Not Done	Hospital Ku...	lov	you
1006	Citizen		0	Not Vaccinat...	CansinoBio	16/12/21	Approved	Not Applicab...	Not Applicable	Bukit Jalil	ally	log
1007	Non Citizen	Water bottle	3	Not Vaccinat...	CansinoBio	NA	Not Done	Not Applicab...	Not Applicable	Bukit Jalil	not	mine
1008	Citizen	first last	18	Not Vaccinat...	NA	NA	Not Done	NA	Not Done	NA	Boieng101	hola
1009	Citizen	wwwwww...	0	Vaccinated	Pfizer	28/11/21	Approved	28/11/21	Done	Bukit Jalil	your	rouy
1010	Citizen		0	Not Vaccinat...	NA	NA	Not Done	NA	Not Done	NA	test	test
1011	Citizen	Trying~	10	Not Vaccinat...	Pfizer	NA	Not Done	NA	Not Done	NA	try	try
1012	Citizen	Naruto	17	Vaccinated	Pfizer	22/11/2021	Done	11/12/21	Done	Mid Valley	Boruto	burrito
1013	Non Citizen	Gracias	24	Not Vaccinat...	Pfizer	28/12/2021	Not Done	NA	Not Done	Mid Valley	Aloha	Luna
1014	Citizen	Taylor	57	Not Vaccinat...	Sinovac	22/12/2021	Approved	NA	Not Done	Mid Valley	Taylor	Swift

In the add feature, people's information is displayed for the personnel's convenience. It has a search function as well. The displayed information is updated after ID-1014 is added to the appointment, which is the recently registered Taylor.

Appointment ID	People ID	Location	Vaccine	Date	Progress
1001	1004	Bukit Jalil	Pfizer	30/11/21	NA
1002	1005	Hospital Kuala ...	Pfizer	23/12/21	NA
1003	1006	Bukit Jalil	CansinoBio	16/12/21	NA
1005	1012	Mid Valley	Pfizer	22/12/21	Completed
1006	1012	Mid Valley	Pfizer	11/12/21	Completed
1007	1009	Bukit Jalil	Pfizer	28/11/21	Completed
1008	1009	Bukit Jalil	Pfizer	28/11/21	Completed
1009	1014	Mid Valley	Sinovac	22/12/2021	NA

As you can see from the display feature in remove feature, Taylor(ID1014) is added into the appointment. Here, personnel can remove the appointments, or essentially cancelling the appointments.

Appointment ID	People ID	Location	Vaccine	Date	Progress
1001	1004	Bukit Jalil	Pfizer	30/11/21	NA
1002	1005	Hospital Kuala ...	Pfizer	23/12/21	NA
1003	1006	Bukit Jalil	CansSinoBio	16/12/21	NA
1005	1012	Mid Valley	Pfizer	22/12/21	Completed
1006	1012	Mid Valley	Pfizer	11/12/21	Completed
1007	1009	Bukit Jalil	Pfizer	28/11/21	Completed
1008	1009	Bukit Jalil	Pfizer	28/11/21	Completed
1009	1014	Mid Valley	Sinovac	22/12/2021	Completed

This is the last feature of appointment management. The personnel can modify the progress of the appointment from NA to completed and vice-versa. If completed, the add feature will automatically add appointment for the second vaccine when the add feature is executed.

6.0 Additional Features

6.1 String[]

```
String[] brokenLine = line.split("\t");
```

String[] is the data type for array. Many individual strings can be inserted into one array.

6.2 equals()

```
if(item.equals(brokenLine[col_num]) == true && brokenLine[5].equals("NA")) {  
    fr.close();  
    return true;  
}
```

equals() helps to check if two individual data are identical. Referring to the figure above, it checks whether item is identical to brokenLine[col_num], it will return true if match and false if non-identical.

6.3 split()

```
String[] brokenLine = line.split("\t");
```

split() is capable of splitting a string based on an argument. For example, the string line is separated by “\t”. Hence if there’s two “\t”, three strings will be produced from that split().

6.4 Arrays.stream().collect(Collectors.joining())

```
appointment_content[i] = Arrays.stream(line_content).collect(Collectors.joining("\t"));
```

Arrays.stream(arg1).collect(Collectors.joining(arg2)) joins strings of an array together to form a single string. Each string in arg1 array is joined with arg2 between them.

arg1 = line_content.

arg2 = “\t”.

6.5 length



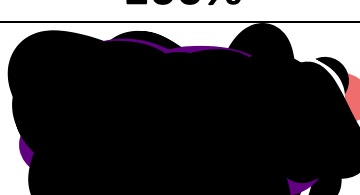

```
for (int i = 0; i < brokenLine.length; i++){
```

arg.length helps to find the size of an array. This ensures that the index never goes out of range and causes an error.

Conclusion

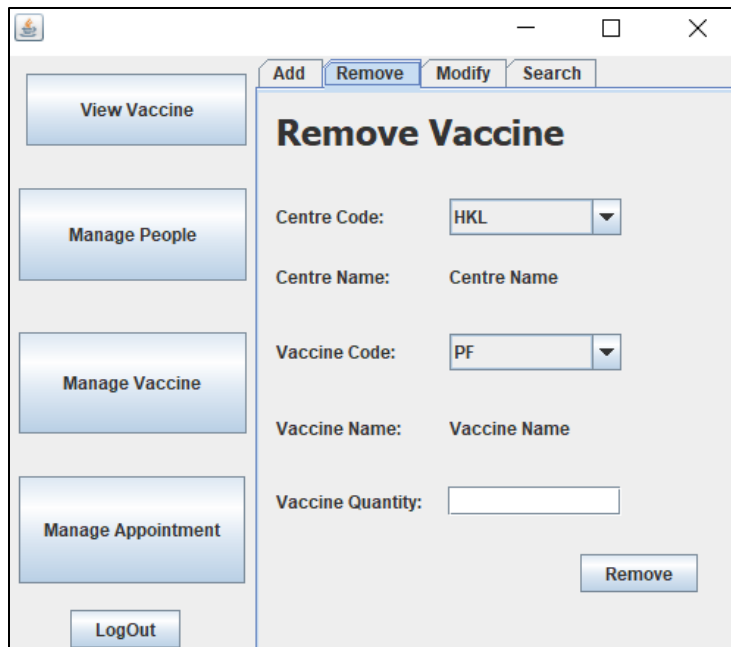
In conclusion, we have applied Object Oriented Concepts in the form of Java Programming within this System. The System created named “Covid-19 Vaccine Registration System” Has the required functions for the personnel and the people of the Country of Malaysia to use. Allowing Citizens or Non-Citizens to register for an account and apply for a vaccine. On The other hand, the Personnel is able to modify the records of Vaccine, Centre and People Data.

Work Load Matrix

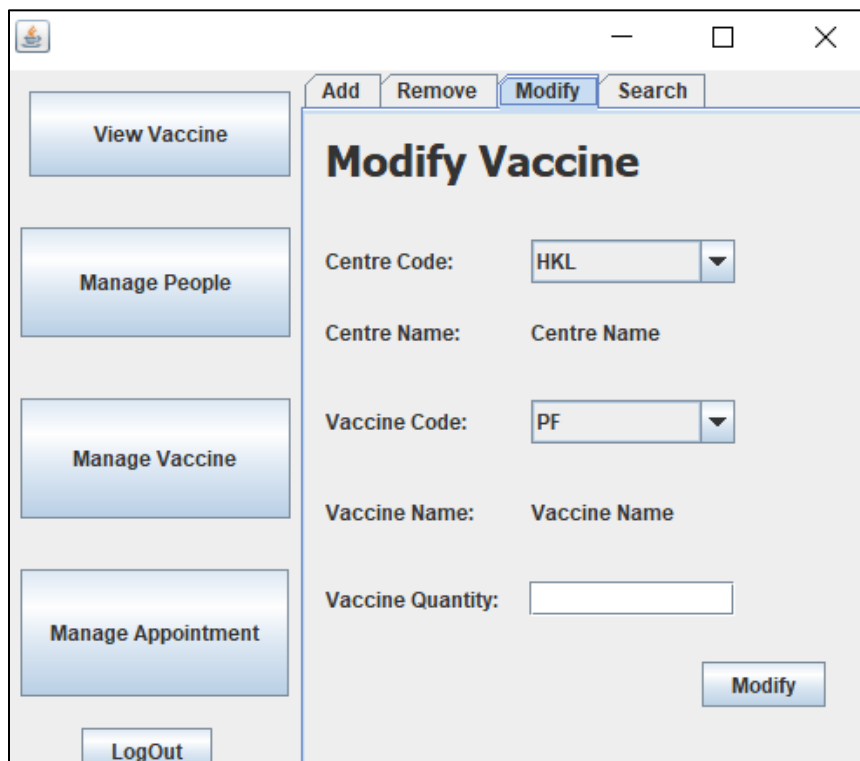
RESPONSIBILITY ASSIGNMENT MATRIX		
COMPONENTS		David Wee Ming Junn 
Admin access page		50%
		50%
People access page	50%	
	50%	
TOTAL	100%	100%
STUDENT SIGNATURE		

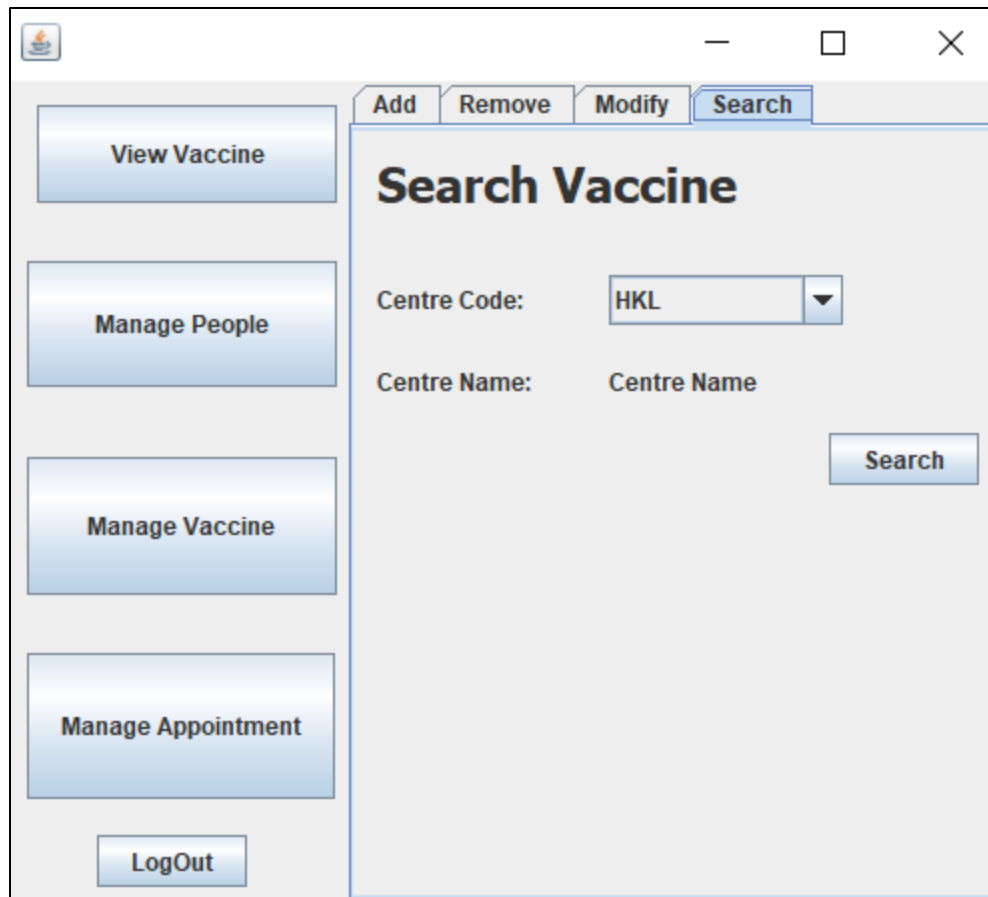
[illegible]

34 of 37



(Figure 7.2: Remove Vaccine gui)

(Figure 7.3: *Modify Vaccine* gui)



(Figure 7.4: *Search Vaccine gui*)

References

- Data Flair. (n.d.). *Loops in Java(for, while, do-while) Coding with Easy Method*. Retrieved from <https://data-flair.training/blogs/loops-in-java/>
- GeeksforGeeks. (2021, June 28). *geeksforgeeks.org*. Retrieved from Overriding in Java: <https://www.geeksforgeeks.org/overriding-in-java/>
- Herrity, K. (2021, October 6). *What Is Object Oriented Programming? The Four Basic Concepts of OOP*. Retrieved from indeed.com: <https://www.indeed.com/career-advice/career-development/what-is-object-oriented-programming>
- HowToDoInJava. (2020, August 30). *Java 8 – Join String Array – Convert Array to String*. Retrieved from <https://howtodoinjava.com/java8/java-8-join-string-array-example/>
- Janssen, T. (2017, December 22). *OOP Concepts for Beginners: What is Polymorphism*. Retrieved from Stackify.com: <https://stackify.com/oop-concept-polymorphism/>
- javaTpoint. (n.d.). *Exception Handling in Java*. Retrieved from javaTpoint: <https://www.javatpoint.com/exception-handling-in-java#:~:text=Exception%20Handling%20is%20a%20mechanism%20to%20handle%20runtime,the%20TV%27s%20watch%20history%20and%20influence%20TV%20recommen%20dations.>
- javaTpoint. (n.d.). *Java String length()*. Retrieved from <https://www.javatpoint.com/java-string-length>
- javaTpoint. (n.d.). *javatpoint.com*. Retrieved from Java Try-Catch Block: <https://www.javatpoint.com/try-catch-block>
- Seetha. (2018, February 5). *What are basic Object oriented programming concepts?* Retrieved from tutorialspoint.com: <https://www.tutorialspoint.com/What-are-basic-Object-oriented-programming-concepts>
- tutorialspoint. (n.d.). *Java-Encapsulation*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/java/java_encapsulation.htm
- w3schools. (n.d.). *Java Data Types*. Retrieved from w3schools.com: https://www.w3schools.com/java/java_data_types.asp