UMS
UNIVERSITI MALAYSIA SABAH

KP14603 (OBJECT ORIENTED PROGRAMMING CONCEPTS) – SECTION 2
SEMESTER 2 2019/2020

INDIVIDUAL PROJECT

MUHAMMAD MIRZA MUQRI BIN MUSA
BI19270362

Introduction

In this project, I will be creating a Covid-19 Self-Assessment Tools. This tool will ask you questions such as your name, age, symptoms, pre-existing conditions and travel history, which will then tell you whether you have the symptoms of Covid-19 or no. It will also save the data to a file which then can be read or delete using the GUI.

Java Code

Project2.java

```
package project.pkg2;

public class Project2 {

    public static WelcomePage welcomePage;

    public static void main(String[] args) {
        welcomePage = new WelcomePage();

        welcomePage.openPage();
        openPage();
    }

    public static void openPage() {
        System.out.println("Opening page...");
    }

}
```

WelcomePage.java

```
package project.pkg2;

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;

public class WelcomePage extends JFrame {

    JPanel panel;
    JLabel title, disclaimer;
    JButton history, start, exit;

    ImageIcon logo;

    /*
    For reference: x coords, y coords, width, height
     */
    public void openPage() {
        setTitle("Covid-19 Self Assessment Tool");
        setSize(432, 700);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);

        panel = new JPanel();

        panel.setLayout(null);
        panel.setBackground(Color.decode("#04b3ad"));

        title = new JLabel("<html><div style='text-align: center;'>COVID-
19<br/>Self-Assessment Tool</div></html>", SwingConstants.CENTER);
        title.setFont(new Font(title.getFont().toString(), Font.BOLD, 14));
        title.setForeground(Color.WHITE);
        title.setBounds(80, 170, 250, 250);

        disclaimer = new JLabel("<html><div style='text-align: center;'>This
program is not<br>affiliated with any government.</div></html>",
SwingConstants.CENTER);
        disclaimer.setFont(new Font(title.getFont().toString(), Font.BOLD,
10));
        disclaimer.setForeground(Color.WHITE);
        disclaimer.setBounds(80, 500, 250, 250);

        start = new JButton("Start Assessment");
        start.setBounds(130, 340, 150, 30);
```

```
        designBtn(start, "#C4EF7B", Color.WHITE);
        startAssessmentBtn(start);

        history = new JButton("History");
        history.setBounds(130, 380, 150, 30);
        designBtn(history, "#93E48A", Color.WHITE);
        historyBtn(history);

        exit = new JButton("Exit");
        exit.setBounds(130, 420, 150, 30);
        designBtn(exit, "#37C5A6", Color.WHITE);
        exitBtn(exit);

        logo = new ImageIcon(getClass().getResource("image/logo.png"));

        // Adding images
        JLabel image = new JLabel(logo);
        image.setBounds(-10, 30, 432, 200);
        panel.add(image);

        // Adding all important items
        panel.add(title);
        panel.add(disclaimer);

        panel.add(start);
        panel.add(history);
        panel.add(exit);

        add(panel);

        setVisible(true);
    }

    public void designBtn(JButton button, String bg, Color text) {
        button.setBorder(null);
        button.setFocusPainted(false);
        button.setBackground(Color.decode(bg));
        button.setForeground(text);
    }

    public void startAssessmentBtn(JButton btn) {
        btn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                setVisible(false);
                new AssessmentPage();
            }
        });
    }

    public void exitBtn(JButton btn) {
        btn.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent ae) {
                dispose();
            }
        });
    }

    public void historyBtn(JButton btn) {
        btn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                setVisible(false);
                new HistoryPage(1);
            }
        });
    }
}
```

AssessmentPage.java

```
package project.pkg2;

import java.awt.Color;
import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javafx.scene.control.CheckBox;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AssessmentPage extends JFrame {

    JPanel panel;

    JTextField name, age;

    JLabel nameLbl, ageLbl, symptomsLbl, existingcondLbl, historyLbl,
preventionTips;

    JCheckBox fever, cough, fatigue, chestTightness, shortBreath,
            runnyNose, soreThroat, musclePain;

    JCheckBox highCholesterol, type2Diabetes, highBp, heart, stroke,
            chronicKidney;

    JCheckBox contactCovid, travelCountry, contactTravel;

    JButton submitBtn, backBtn;

    public AssessmentPage() {
        setTitle("Covid-19 Self Assessment Tool");
        setSize(760, 700);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);

        panel = new JPanel();

        panel.setLayout(null);
        panel.setBackground(Color.decode("#04b3ad"));

        // Name
        nameLbl = new JLabel("Full Name");
        nameLbl.setBounds(10, 10, 250, 10);
        nameLbl.setForeground(Color.WHITE);

        name = new JTextField(32);
```

```
        name.setBounds(10, 30, 250, 30);
        name.setBorder(null);

        // Age
        ageLbl = new JLabel("Age");
        ageLbl.setBounds(10, 70, 250, 30);
        ageLbl.setForeground(Color.WHITE);

        age = new JTextField(32);
        age.setBounds(10, 100, 250, 30);
        age.setBorder(null);

        // Symptoms
        symptomsLbl = new JLabel("Do you have any of the following
symptoms?");
        symptomsLbl.setBounds(10, 150, 250, 30);
        symptomsLbl.setForeground(Color.WHITE);

        fever = new JCheckBox("Fever");
        fever.setBounds(10, 180, 150, 30);

        cough = new JCheckBox("Cough");
        cough.setBounds(170, 180, 150, 30);

        fatigue = new JCheckBox("Fatigue");
        fatigue.setBounds(10, 220, 150, 30);

        chestTightness = new JCheckBox("Chest Tightness");
        chestTightness.setBounds(170, 220, 150, 30);

        shortBreath = new JCheckBox("Shortness of Breath");
        shortBreath.setBounds(10, 260, 150, 30);

        runnyNose = new JCheckBox("Runny Nose");
        runnyNose.setBounds(170, 260, 150, 30);

        soreThroat = new JCheckBox("Sore Throat");
        soreThroat.setBounds(10, 300, 150, 30);

        musclePain = new JCheckBox("Muscle Pain");
        musclePain.setBounds(170, 300, 150, 30);

        // Existing Condition
        existingcondLbl = new JLabel("Do you have the following existing
conditions?");
        existingcondLbl.setBounds(10, 350, 300, 30);
        existingcondLbl.setForeground(Color.WHITE);

        highCholesterol = new JCheckBox("High Cholesterol");
        highCholesterol.setBounds(10, 380, 150, 30);

        type2Diabetes = new JCheckBox("Type 2 Diabetes");
        type2Diabetes.setBounds(170, 380, 150, 30);
```

```
        highBp = new JCheckBox("High Blood Pressure");
        highBp.setBounds(10, 420, 150, 30);

        heart = new JCheckBox("Heart Disease");
        heart.setBounds(170, 420, 150, 30);

        stroke = new JCheckBox("Stroke");
        stroke.setBounds(10, 460, 150, 30);

        chronicKidney = new JCheckBox("Chronic Kidney");
        chronicKidney.setBounds(170, 460, 150, 30);

        // Epidemiological history
        historyLbl = new JLabel("Do you have the following epidemiological
history?");
        historyLbl.setBounds(10, 510, 300, 30);
        historyLbl.setForeground(Color.WHITE);

        contactCovid = new JCheckBox("Contact with COVID-19 confirmed
patient.");
        contactCovid.setBounds(10, 540, 310, 30);

        travelCountry = new JCheckBox("Travel or residence in an epidemic
area/country.");
        travelCountry.setBounds(10, 580, 310, 30);

        contactTravel = new JCheckBox("Contact with people who recently
traveled.");
        contactTravel.setBounds(10, 620, 310, 30);

        // Prevention Tips
        preventionTips = new JLabel("<html>Prevention Tips:"
                + "<br>◦ Minimize outdoor activities. Avoid going to areas
where COVID-19 is spreading and other crowded public places."
                + "<br><br>◦ Pay attention to personal protection and
maintan hand hygiene. It is recommended to wear surgical masks or N95 masks
when going out."
                + "<br><br>◦ Avoid touching surfaces in public places. Wash
your hands with hand sanitizer or soap under running water, or use alcohol
hand sanitizer. Cover your nose and mouth when sneezing or coughing."
                + "<br><br>◦ Actively measure your body temperature when you
feel feverish. If there is a child in the family, feel the child's forehead
twice a day and take the temperature of the child if necessary. If there are
suspected symptoms, put on a mask and seek medical consultation nearby."
                + "<br><br><br> Disclaimer: This results is for reference
only, and should not be relied upon as a basis for diagnosis and treatement.
Please decide the next steps based on your own situation and follow the
advise given by a doctor.</html>");
        preventionTips.setBounds(350, 10, 350, 390);
        preventionTips.setForeground(Color.WHITE);

        // Buttons
```

```
        submitBtn = new JButton("Submit Assessments");
        submitBtn.setBounds(350, 410, 160, 30);
        designBtn(submitBtn, "#5cb85c", Color.WHITE);
        onSubmit(submitBtn);

        backBtn = new JButton("Back");
        backBtn.setBounds(530, 410, 160, 30);
        designBtn(backBtn, "#d9534f", Color.WHITE);
        backBtn(backBtn);

        // Adding components to panel
        panel.add(nameLbl);
        panel.add(name);

        panel.add(ageLbl);
        panel.add(age);

        panel.add(symptomsLbl);
        panel.add(fever);
        panel.add(cough);
        panel.add(fatigue);
        panel.add(chestTightness);
        panel.add(shortBreath);
        panel.add(runnyNose);
        panel.add(soreThroat);
        panel.add(musclePain);

        panel.add(existingcondLbl);
        panel.add(highCholesterol);
        panel.add(type2Diabetes);
        panel.add(highBp);
        panel.add(heart);
        panel.add(stroke);
        panel.add(chronicKidney);

        panel.add(historyLbl);
        panel.add(contactCovid);
        panel.add(travelCountry);
        panel.add(contactTravel);

        panel.add(preventionTips);
        panel.add(submitBtn);
        panel.add(backBtn);

        for (Component c : panel.getComponents()) {
            if (c instanceof  JCheckBox) {
                c.setBackground(Color.decode("#37C5A6"));
                c.setForeground(Color.WHITE);
            }
        }

        add(panel);
```

```
        setVisible(true);
    }

    public void designBtn(JButton button, String bg, Color text) {
        button.setBorder(null);
        button.setFocusPainted(false);
        button.setBackground(Color.decode(bg));
        button.setForeground(text);
    }

    public void backBtn(JButton btn) {
        btn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                setVisible(false);
                Project2.welcomePage.setVisible(true);
            }
        });
    }

    public void onSubmit(JButton btn) {
        btn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
                int userAge = 0;
                String fullName = "";

                if (name.getText().length() <= 0) {
                    JOptionPane.showMessageDialog(panel, "Please enter your
full name.");
                    return;
                }
                else {
                    fullName = name.getText();
                }

                try {
                    userAge = Integer.parseInt(age.getText());
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(panel, "Please enter a
valid age.");
                    return;
                }

                Assessments assessments = new Assessments(fullName,
userAge);

                name.setText("");
                age.setText("");

                if (fever.isSelected()) {
                    assessments.getSymptoms().add("Fever");
                    fever.setSelected(false);
```

```
                }
                if (cough.isSelected()) {
                    assessments.getSymptoms().add("Cough");
                    cough.setSelected(false);
                }
                if (fatigue.isSelected()) {
                    assessments.getSymptoms().add("Fatigue");
                    fatigue.setSelected(false);
                }
                if (chestTightness.isSelected()) {
                    assessments.getSymptoms().add("Chest Tightness");
                    chestTightness.setSelected(false);
                }
                if (shortBreath.isSelected()) {
                    assessments.getSymptoms().add("Shortness of Breath");
                    shortBreath.setSelected(false);
                }
                if (runnyNose.isSelected()) {
                    assessments.getSymptoms().add("Runny Nose");
                    runnyNose.setSelected(false);
                }
                if (soreThroat.isSelected()) {
                    assessments.getSymptoms().add("Sore Throat");
                    soreThroat.setSelected(false);
                }
                if (musclePain.isSelected()) {
                    assessments.getSymptoms().add("Muscle Pain");
                    musclePain.setSelected(false);
                }

                if (highCholesterol.isSelected()) {
                    assessments.getPreExistingCondition().add("High
Cholesterol");
                    highCholesterol.setSelected(false);
                }
                if (type2Diabetes.isSelected()) {
                    assessments.getPreExistingCondition().add("Type 2
Diabetes");
                    type2Diabetes.setSelected(false);
                }
                if (highBp.isSelected()) {
                    assessments.getPreExistingCondition().add("High Blood
Pressure");
                    highBp.setSelected(false);
                }
                if (heart.isSelected()) {
                    assessments.getPreExistingCondition().add("Heart
Disease");
                    heart.setSelected(false);
                }
                if (stroke.isSelected()) {
                    assessments.getPreExistingCondition().add("Stroke");
                    stroke.setSelected(false);
```

```
                }
                if (chronicKidney.isSelected()) {
                    assessments.getPreExistingCondition().add("Chronic
Kidney");
                    chronicKidney.setSelected(false);
                }

                if (contactCovid.isSelected()) {
                    assessments.getEpidemiologicalHistory().add("Contact
with COVID-19 confirmed patient.");
                    contactCovid.setSelected(false);
                }
                if (travelCountry.isSelected()) {
                    assessments.getEpidemiologicalHistory().add("Travel or
residence in an epidemic area/country.");
                    travelCountry.setSelected(false);
                }
                if (contactTravel.isSelected()) {
                    assessments.getEpidemiologicalHistory().add("Contact
with people who recently traveled.");
                    contactTravel.setSelected(false);
                }

                assessments.showResult();
                assessments.save();
            }
        });
    }
}
```

Assessments.java

```
package project.pkg2;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class Assessments {

    private String name;

    private int age;

    private List<String> symptoms;
    private List<String> preExistingCondition;
    private List<String> epidemiologicalHistory;

    public Assessments(String name, int age) {
        this.name = name;
        this.age = age;

        symptoms = new ArrayList<>();
        preExistingCondition = new ArrayList<>();
        epidemiologicalHistory = new ArrayList<>();
    }

    public void showResult() {
        if (symptoms.size() >= 1 || preExistingCondition.size() >= 1 ||
epidemiologicalHistory.size() >= 1) {
            JOptionPane.showMessageDialog(null, "Contact your health care
provider.\nYour answers indicate that you have a possible symptoms of COVID-
19!");
        } else {
            JOptionPane.showMessageDialog(null, "Manage your symptoms at
home.\nYou do not have any symptoms that indicates for COVID-19.");
        }
    }

    public void save() {
        String data = name + ":" + age + ":";

        if (getSymptoms().size() >= 1) {
            for (int i = 0; i < getSymptoms().size(); i++) {
                if ((1 + i) == getSymptoms().size()) {
                    data += getSymptoms().get(i);
                } else {
                    data += getSymptoms().get(i) + "|";
                }
```

```
            }
        } else {
            data += "None";
        }

        data += ":";

        if (getEpidemiologicalHistory().size() >= 1) {
            for (int i = 0; i < getEpidemiologicalHistory().size(); i++) {
                if ((1 + i) == getEpidemiologicalHistory().size()) {
                    data += getEpidemiologicalHistory().get(i);
                } else {
                    data += getEpidemiologicalHistory().get(i) + "|";
                }
            }
        } else {
            data += "None";
        }

        data += ":";

        if (getPreExistingCondition().size() >= 1) {
            for (int i = 0; i < getPreExistingCondition().size(); i++) {
                if ((1 + i) == getPreExistingCondition().size()) {
                    data += getPreExistingCondition().get(i);
                } else {
                    data += getPreExistingCondition().get(i) + "|";
                }
            }
        } else {
            data += "None";
        }

        try {
            FileWriter fw = new FileWriter("storage.txt", true);
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter pw = new PrintWriter(bw);

            pw.append(data);
            bw.newLine();

            pw.close();
            bw.close();

        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Could not save data!");
        }
    }

    private String getName() {
        return name;
    }
```

```
    private int getAge() {
        return age;
    }

    public List<String> getEpidemiologicalHistory() {
        return epidemiologicalHistory;
    }

    public List<String> getPreExistingCondition() {
        return preExistingCondition;
    }

    public List<String> getSymptoms() {
        return symptoms;
    }
}
```

HistoryPage.java

```
package project.pkg2;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class Assessments {

    private String name;

    private int age;

    private List<String> symptoms;
    private List<String> preExistingCondition;
    private List<String> epidemiologicalHistory;

    public Assessments(String name, int age) {
        this.name = name;
        this.age = age;

        symptoms = new ArrayList<>();
        preExistingCondition = new ArrayList<>();
        epidemiologicalHistory = new ArrayList<>();
    }

    public void showResult() {
        if (symptoms.size() >= 1 || preExistingCondition.size() >= 1 ||
epidemiologicalHistory.size() >= 1) {
            JOptionPane.showMessageDialog(null, "Contact your health care
provider.\nYour answers indicate that you have a possible symptoms of COVID-
19!");
        } else {
            JOptionPane.showMessageDialog(null, "Manage your symptoms at
home.\nYou do not have any symptoms that indicates for COVID-19.");
        }
    }

    public void save() {
        String data = name + ":" + age + ":";

        if (getSymptoms().size() >= 1) {
            for (int i = 0; i < getSymptoms().size(); i++) {
                if ((1 + i) == getSymptoms().size()) {
                    data += getSymptoms().get(i);
                } else {
                    data += getSymptoms().get(i) + "|";
                }
```

```
            }
        } else {
            data += "None";
        }

        data += ":";

        if (getEpidemiologicalHistory().size() >= 1) {
            for (int i = 0; i < getEpidemiologicalHistory().size(); i++) {
                if ((1 + i) == getEpidemiologicalHistory().size()) {
                    data += getEpidemiologicalHistory().get(i);
                } else {
                    data += getEpidemiologicalHistory().get(i) + "|";
                }
            }
        } else {
            data += "None";
        }

        data += ":";

        if (getPreExistingCondition().size() >= 1) {
            for (int i = 0; i < getPreExistingCondition().size(); i++) {
                if ((1 + i) == getPreExistingCondition().size()) {
                    data += getPreExistingCondition().get(i);
                } else {
                    data += getPreExistingCondition().get(i) + "|";
                }
            }
        } else {
            data += "None";
        }

        try {
            FileWriter fw = new FileWriter("storage.txt", true);
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter pw = new PrintWriter(bw);

            pw.append(data);
            bw.newLine();

            pw.close();
            bw.close();

        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Could not save data!");
        }
    }

    private String getName() {
        return name;
    }
```

```
    private int getAge() {
        return age;
    }

    public List<String> getEpidemiologicalHistory() {
        return epidemiologicalHistory;
    }

    public List<String> getPreExistingCondition() {
        return preExistingCondition;
    }

    public List<String> getSymptoms() {
        return symptoms;
    }
}
```

Object Oriented Concept Implementation

Object and Classes

Most of my methods are using the object and classes concept. Using this methods, it allows me to create and object from another class and use its methods to store, get or set data. For example, I use this method on the main class (Project2.java), I created the instance of WelcomePage in there and make it public so other classes can have access to them, so that we don't need to create a new one every time we need to show the welcome page. Not only in Project2.java, I have used this method on every single classes on this project.

Interface and Inner Class

On this project, it also uses the Inner Class method. We can see this method was nested on HistoryPage.java called StorageFile. The main reason why I choose to make an inner class in HistoryPage is so that it will be much easier to maintain and read them & it can access a private and public data that are on the HistoryPage.

Encapsulation

Encapsulation can mostly be seen on the Assessments.java class. In this class, it stores the user input information's and save it to the local file. It will set the data using a setter and get the data using a get method. This class will also determine if the users need to contact a health care provider or manage their own symptoms.

Inheritance

This project uses inheritance for the JFrame. With this concept, it saves a lot of time and it makes it much easier to make a user interface. Instead of making a new class that does the same thing over and over again, it is best to use Inheritance because you only to inherit from an existing class.

Read and Write Implementation

The writing is mostly done on the Assessments.java class. The way that I saved the data is using a storage.txt file. Once the user submitted an assessment, the program will get all of the information's and then open the file using FileWriter, BufferredWriter and PrintWriter method. With all the data, I created a new string that divides all of the components using a string such as '|'. This will be important for reading the data.

For reading the data, the program will access the storage.txt from the directory of the program and check for every single line on the file. Every single lines contains a different assessment. And by using Split, it is able to determine which data is which. Meanings, it can understand how the data is saved and which data are the user name, age, symptoms, existing conditions and travel history. Finally, the program will display every single data that is stored on storage.txt.

User Manual

Step 1: Opening the Assessments Screen

Click the “Start Assessment” button to open up the assessments menu.

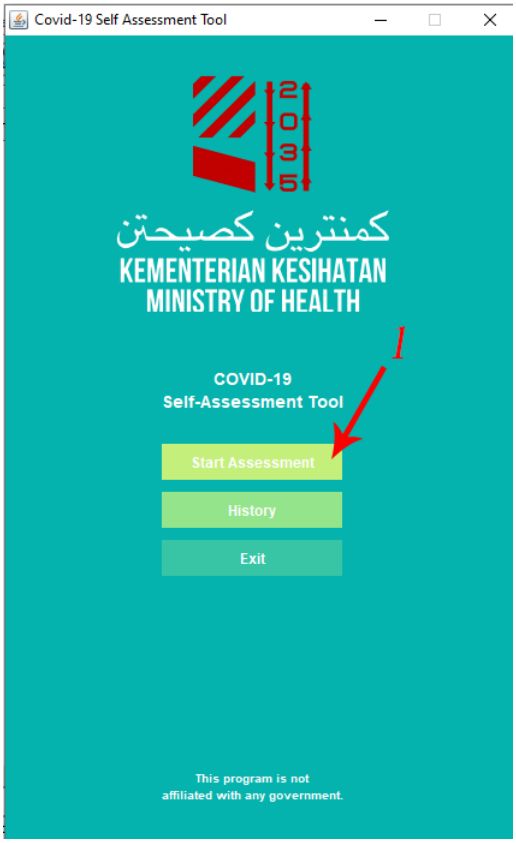

Figure 1 Start Assessment button

Step 2.1: Filling in the required information (User's Name)

In order to use the tool, you will need to enter your name. If you did not enter any names, you will be prompted with an error message asking you to enter a valid name (Figure x).

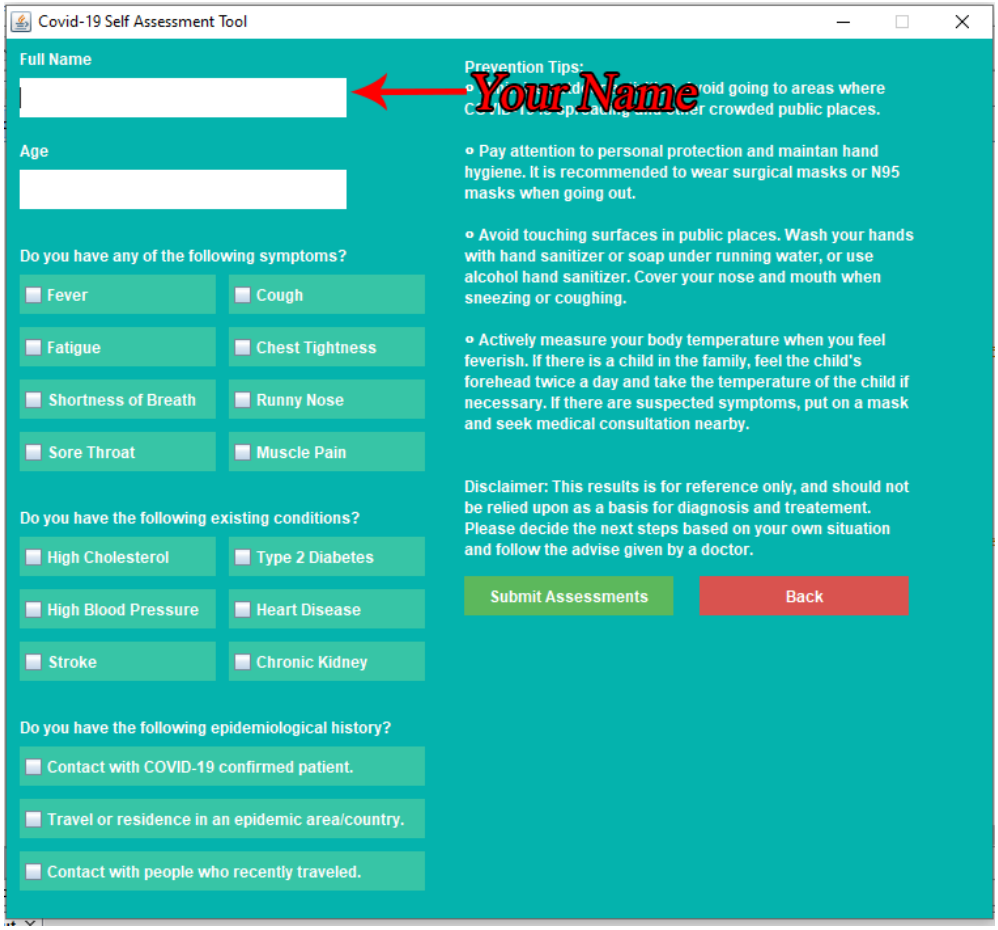

Figure 2 Full Name Input

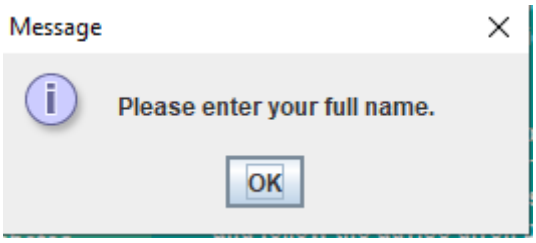

Figure 3 Error if there's no valid names

Step 2.2: Filling in the required information (User's Age)

After entering your name, you will need to enter your current age. If you enter any invalid age, invalid characters or an empty age, you will be prompted with an error message telling you to enter a valid age (Figure 5).

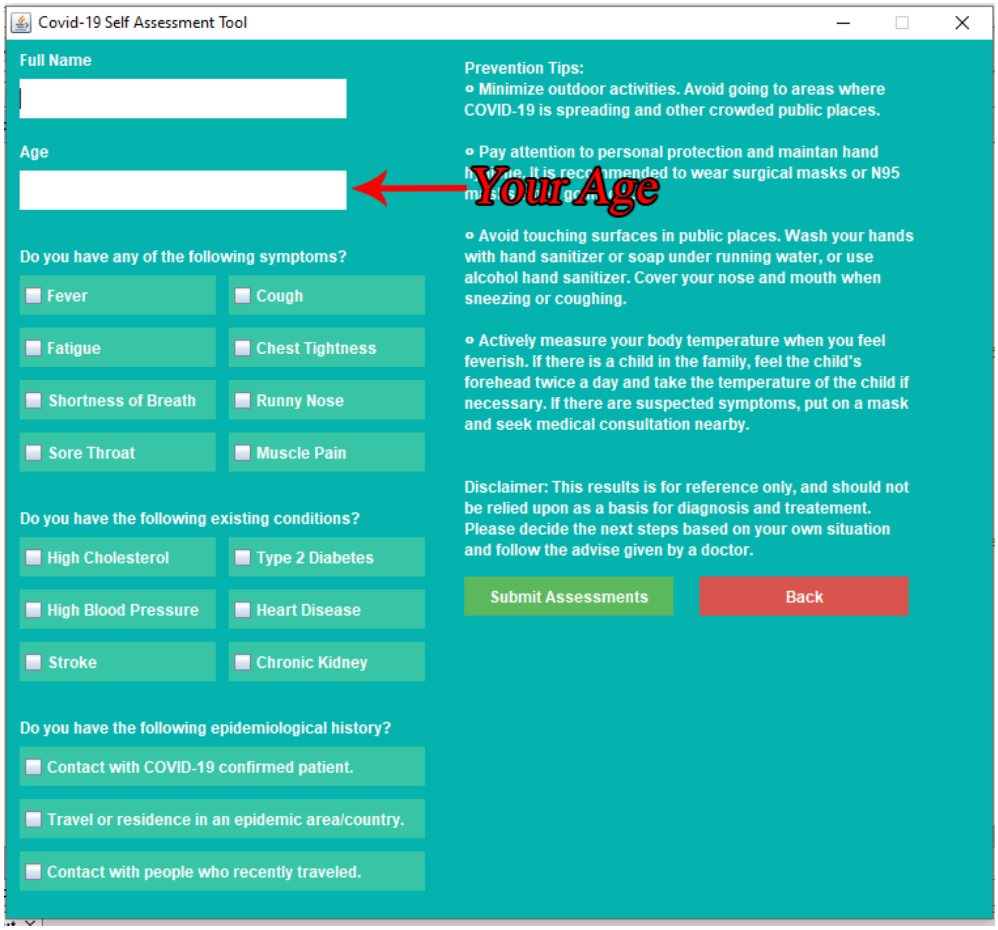

Figure 4 Age Input

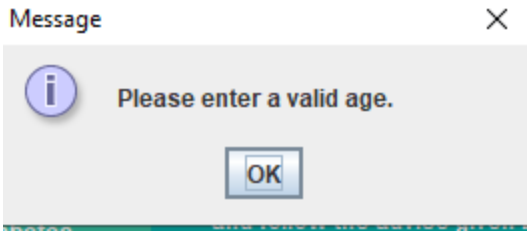

Figure 5 Error if user enters an invalid age

Step 2.3: Filling in the required information (User's Symptoms)

If you have any symptoms that is on the list, please do check them. If you don't have any of the symptoms listed on, you may ignore this sections.

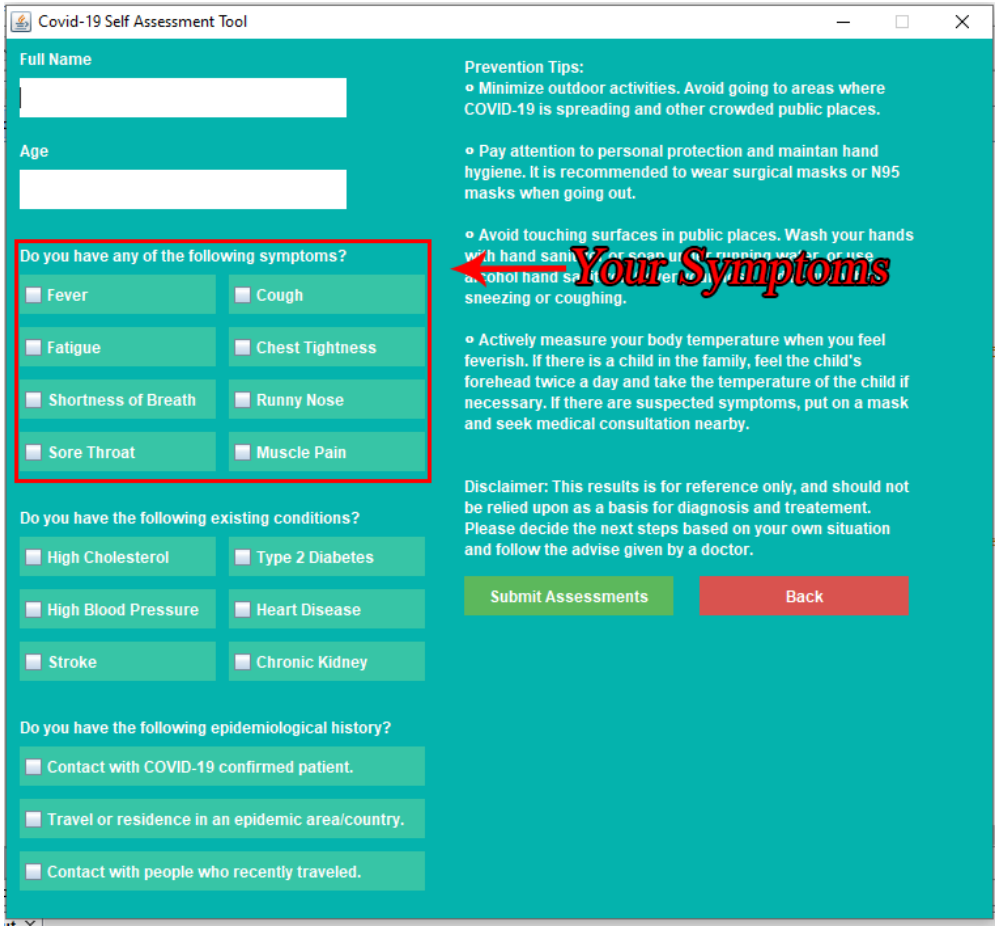

Figure 6 List of symptoms

Step 2.4: Filling in the required information (User's Pre-Existing Conditions)

If you have any pre-existing conditions that are listed on the form, please do check them. But if you don't have any of them, you can skip this part as well.

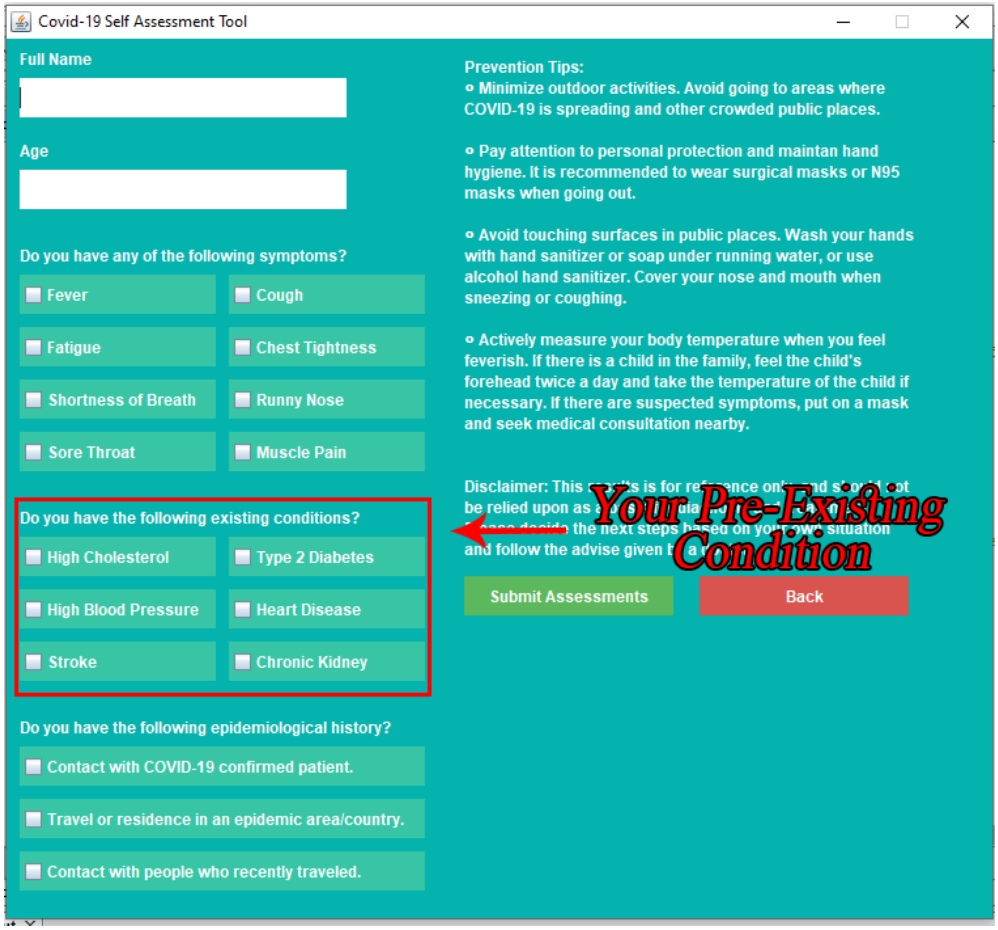

Figure 7 List of pre-existing conditions

Step 2.5: Filling in the required information (User's Epidemiological/Travel History)

If you have traveled in an epidemic area/country or you have been in contact with Covid-19 confirmed patient or been in contact with people who recently traveled, check any of the following epidemiological history. If you do not have any of the following, you may skip this part.

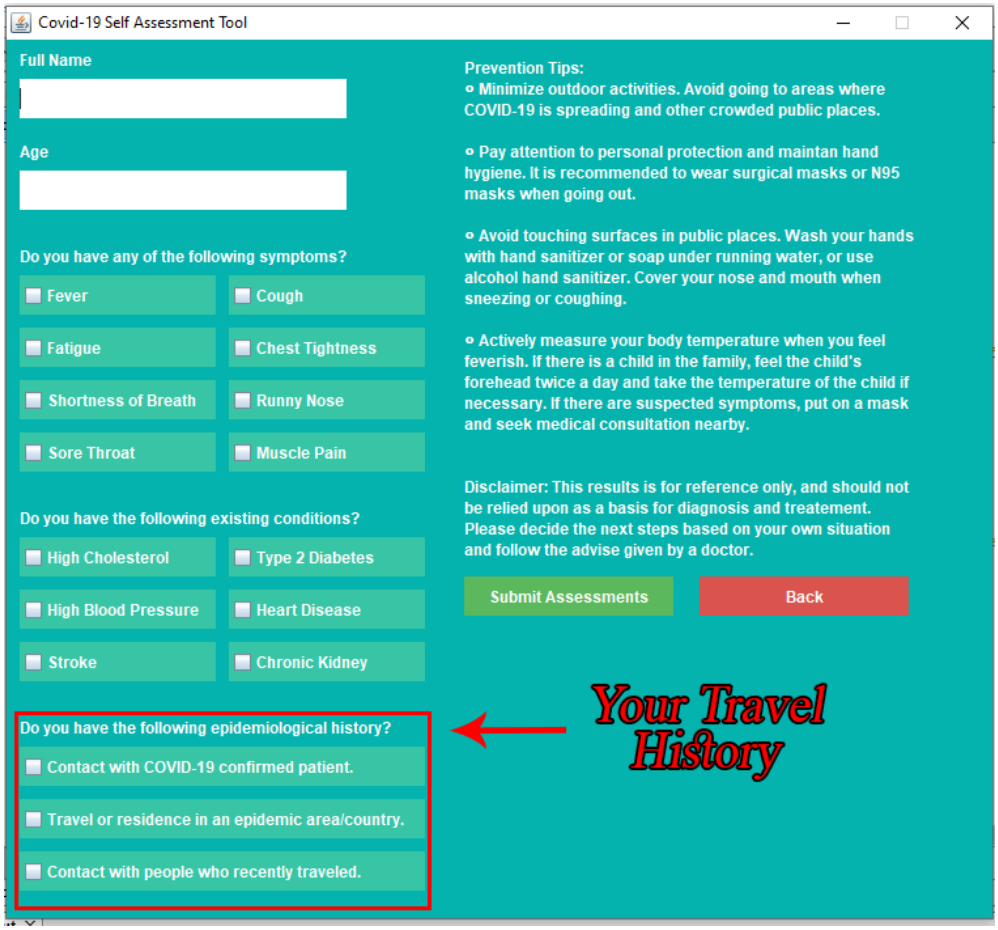

Figure 8 List of epidemiological/travel history

Step 2.6: Submitting the form for assessments

After you have fill in every details on the assessments tools, click the Submit Assessments tools and a message box will appear to let you know if you need to contact a health provider (Figure 11) or you can manage your symptoms at home (Figure 10). If you want to cancel your assessments, do not click the Submit Assessments button instead, click the Back button (Figure X).

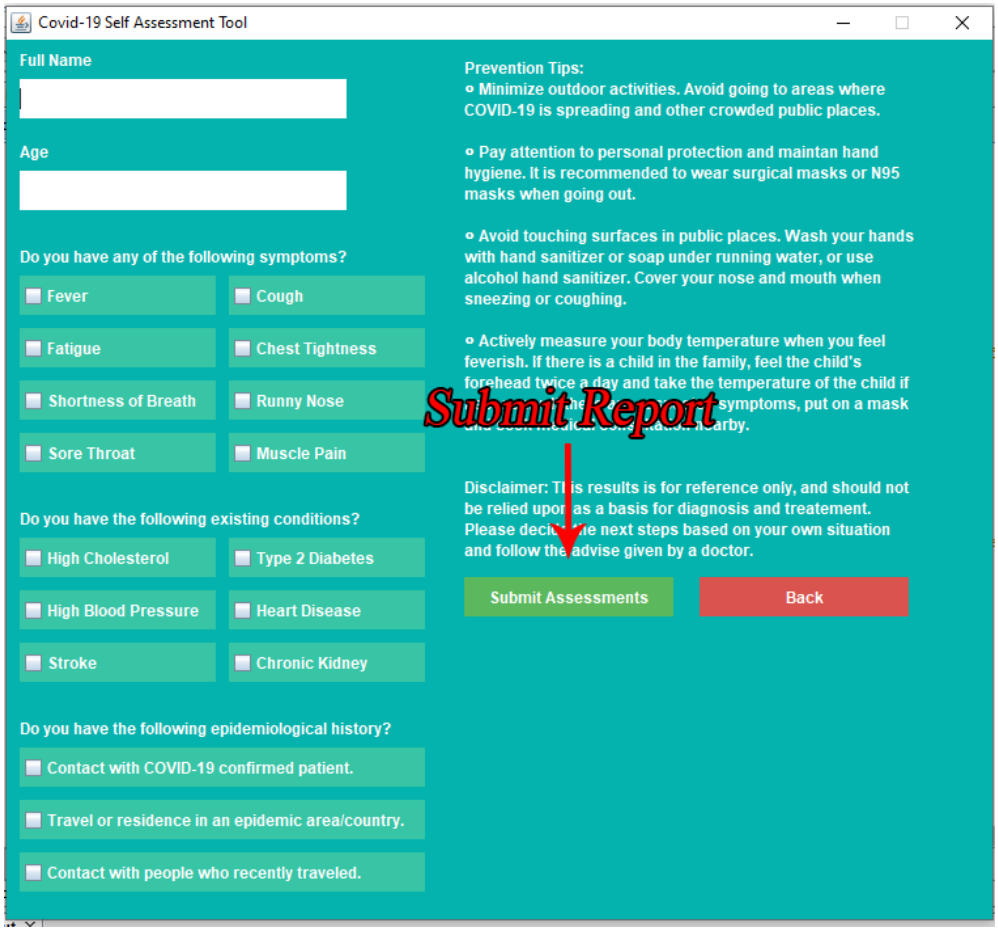

Figure 9 Submit your assessments

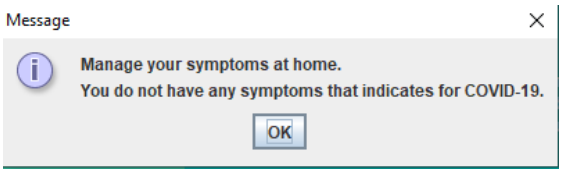

Figure 10 Assessments tools result #1

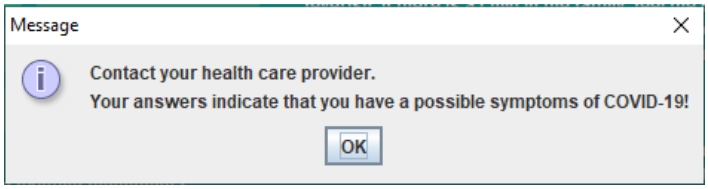

Figure 11 Assessments tools result #2

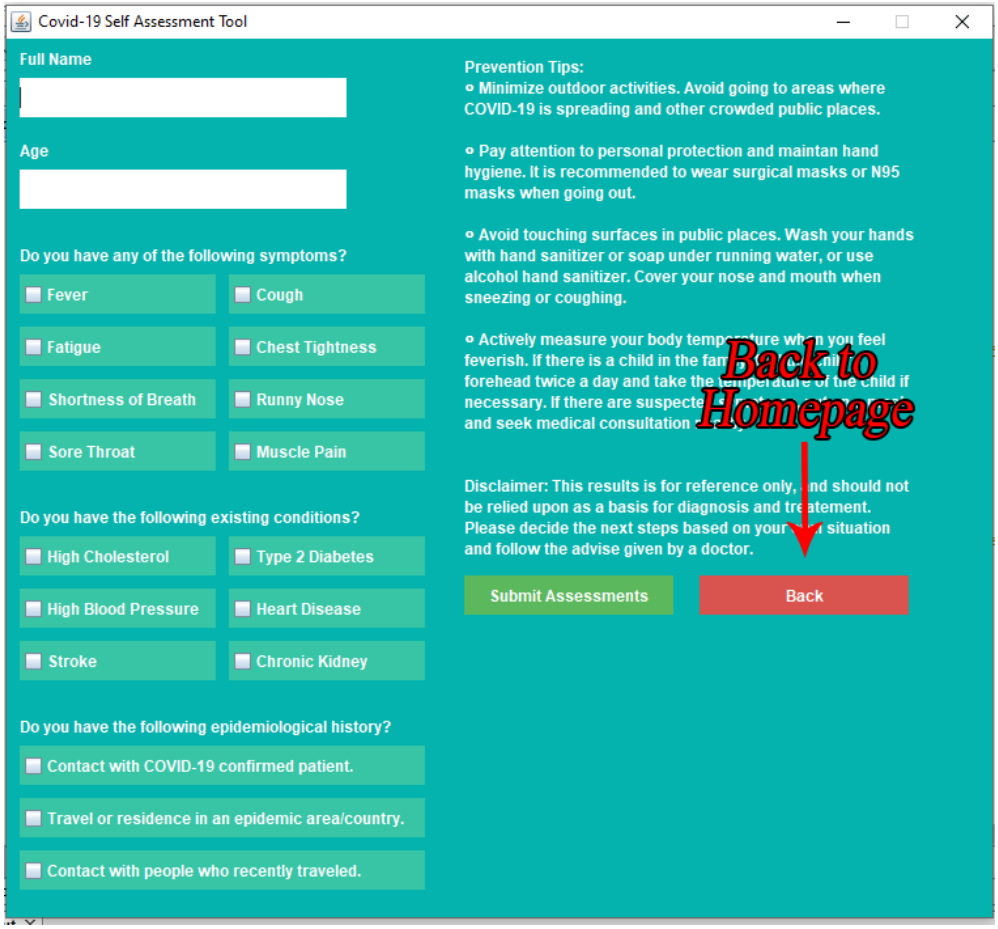

Figure 12 Cancel your assessments

Now that you have done everything, you may close the applications (Step 4.1). If you would like to see or remove the assessments history, please refer to User Manual 3.1 for further details on how to perform a certain action.

Step 3.1: Accessing the Assessments history

Click the History button that is located on the Homepage and the History page will pop up.

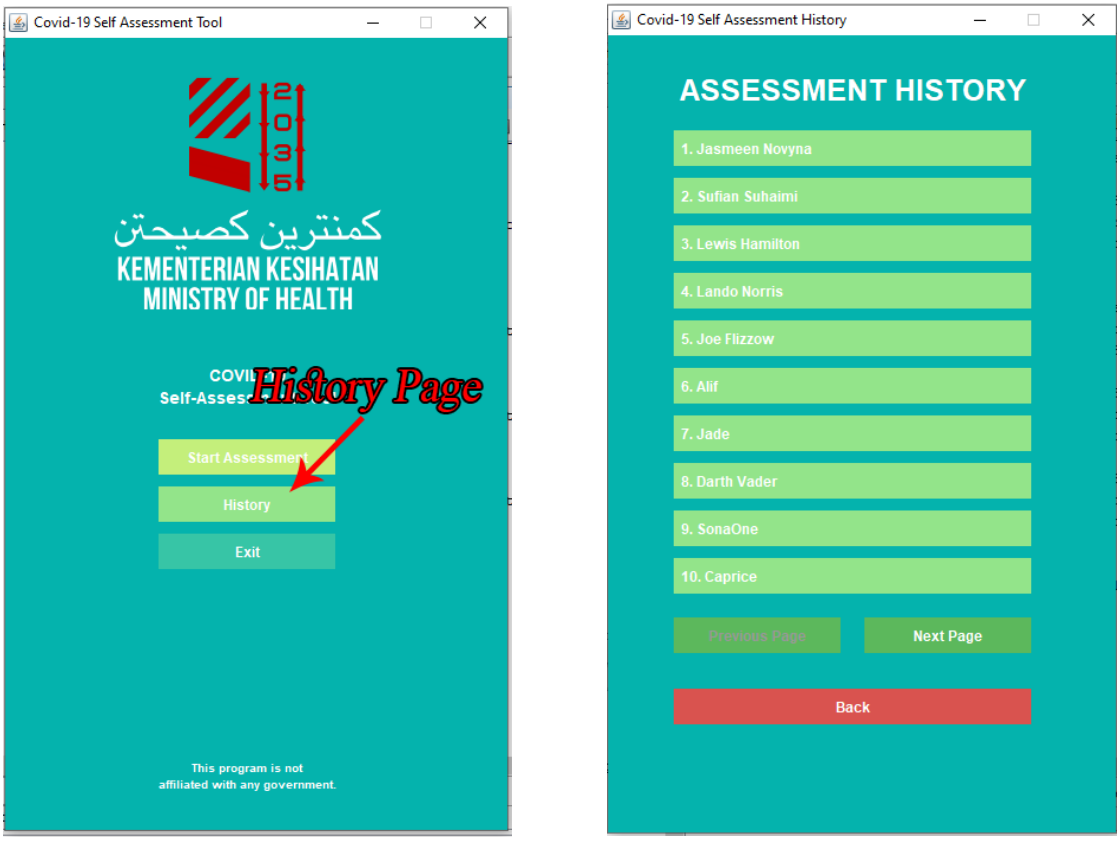

Figure 13 On the left is the homepage and on the right side is what the history page looks like.

Step 3.2: Reading the assessments history

You can read/access them by clicking on their names. If you could not find any names that you are looking for, they are probably on the next page. Click the next page to reveal more list (Figure X).

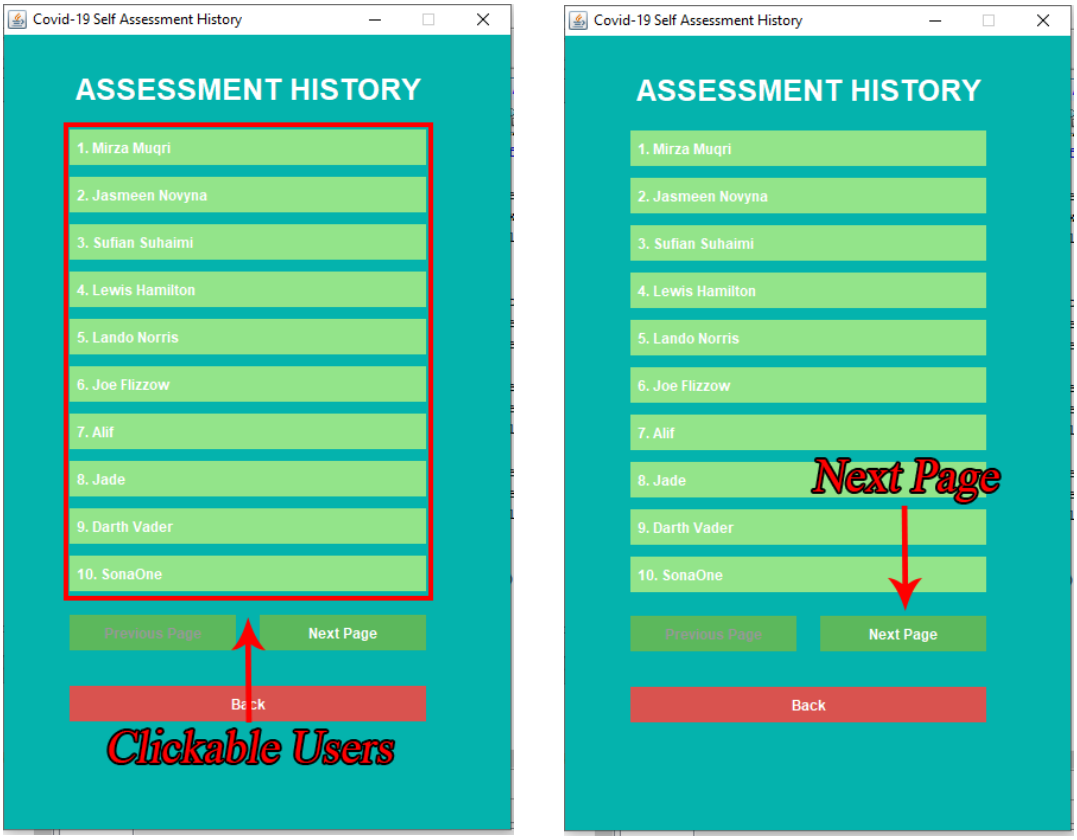

Figure 14 Click any of the users to get more info about their assessments

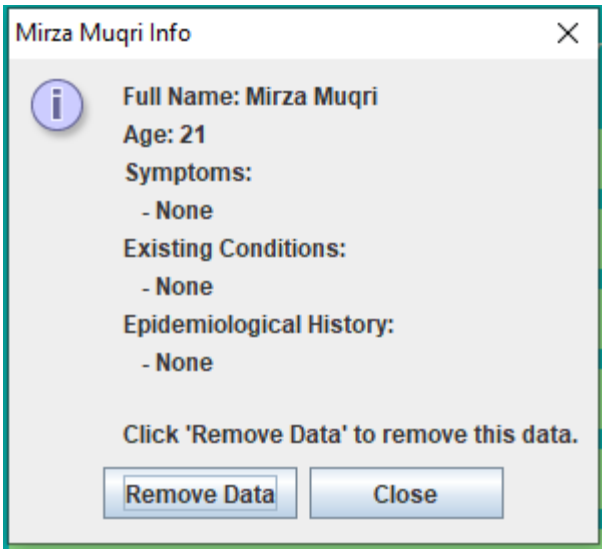

Figure 15 User Informations

Step 3.3: Deleting user assessments

You may delete any of the previous assessments by doing step 3.2 and clicking the Remove Data button from the user info message box.

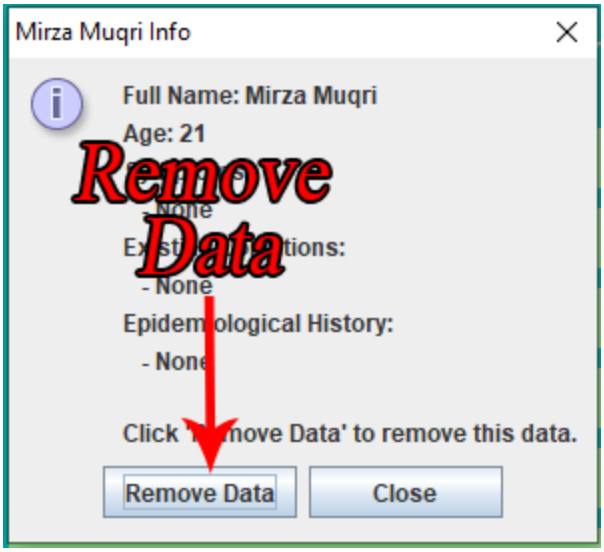

Figure 16 Removing assessments from the history

Figure 17 Before and After a deleted data

Step 4.1: Closing the application

There are multiple ways of closing the applications. You can always close the applications using the X button on top right of the applications or by clicking the Exit button on the homepage.

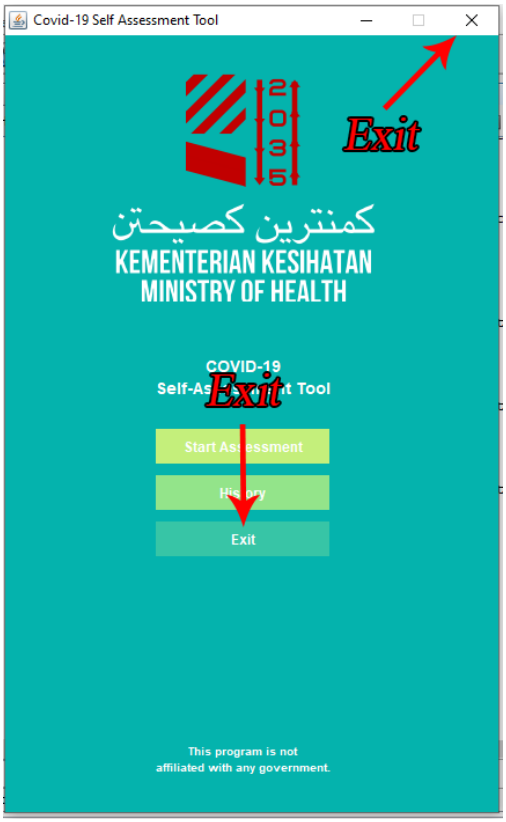

Figure 18 Close button located on the application