UNIVERSITI TUNKU ABDUL RAHMAN

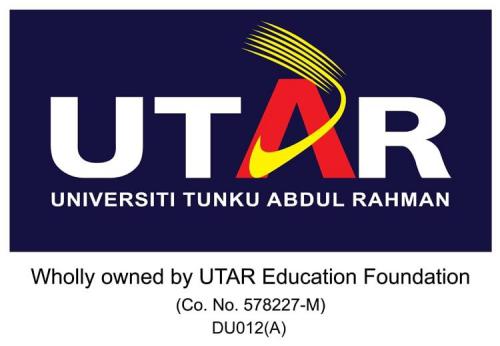
JANUARY 2023 TRIMESTER

**ASSIGNMENT**

**REPORT**

**Candidate is required to fill in ALL the information below:**

|  |  |  |  |
| --- | --- | --- | --- |
| Group Leader: (*Name, ID, Programme*) | **KUIK RUI LUAN 2103790 SE** | | |
| Group member 1: (*Name, ID, Programme*) | **HUI LE YUN 2105611 SE** | | |
| Group member 2: (*Name, ID, Programme*) | **KIM BEI ER 2103893 SE** | | |
| Group member 3: (*Name, ID, Programme*) | **NIVIKA PRASAD A/P KASHI NATH 2104910 SE** | | |
| Group member 4: (*Name, ID, Programme*) | **ZOE CHUA ZI YI 2103816 SE** | | |
| Faculty /Institute/ Centre: | **LKC FES** | Group number: | **G1** |
| Course Code: | **UECS1004/UECS1104** | Course Description: | **Programming and Problem Solving** |

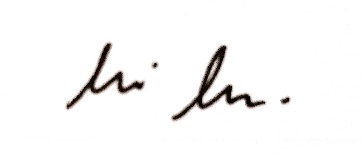


**DECLARATION STATEMENT**

We hereby solemnly and sincerely declare and confirm that we have read, understood and shall abide and comply with all laws, rules, regulations, guidelines and lawful instruction of the University and its staff in relation to the commencement of any assessment / examination during my programme of study in Universiti Tunku Abdul Rahman.

We hereby declare that our submission for all assessment / examination during our programme of study in the University shall be based on our original work, not plagiarised from any source(s) except for citations and quotations which have been duly acknowledged. We are fully aware that students who are suspected of violating this pledge are liable to be referred to the Examination Disciplinary Committee of the University.

We further certify that we have read and understand the above guidelines and agree to abide by the above declarations. The group leader signs this declaration statement on behalf of any and all group members.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group Leader and Member Names and Student I.D.:

KUIK RUI LUAN 2103790 (Group Leader)

HUI LE YUN 2105611

KIM BEI ER 2103893

NIVIKA PRASAD A/P KASHI NATH 2104910

ZOE CHUA ZI YI 2103816

Date: 20/4/2023

### TABLE OF CONTENTS

**DECLARATION .................................................................................................... 1**

**TABLE OF CONTENTS ....................................................................................... 2**

### CHAPTER

**1 INTRODUCTION TO SOLUTION DESIGN................................................ 3**

1.1 Introduction/Background ............................................................ 3

1.2 System Modules .......................................................................... 5

1.2.1 *Module 1: Registration* .................................................. 5

1.2.2 *Module 2: Login* ........................................................... 7

1.3 Structure Chart ........................................................................... 9

1.4 Flow Chart/Pseudocode ............................................................ 10

**2 C++ PROGRAM ............................................................................................. 36**

**3 SAMPLE OUTPUT ........................................................................................ 67**

3.1 *Module 1: Registration* ............................................................. 67

3.1.1 Personal Information Page ...................................... 68

3.1.2 Covid-19 Status Page .............................................. 71

3.1.3 Vaccination Status Page .......................................... 74

3.2 *Module 2: Login* ....................................................................... 80

3.2.1 Login as Admin Page .............................................. 81

3.2.2 Login as User Page ................................................. 89

**4 SAMPLE INPUT ............................................................................................ 93**

4.1 *Text File: Biodata.txt* ............................................................... 93

**TABLE OF TASK DISTRIBUTION................................................................... 95**

## 1 INTRODUCTION TO SOLUTION DESIGN

### Introduction/Background

COVID-19, commonly known as the coronavirus, is a highly contagious respiratory infection discovered in Wuhan, China in December 2019 and has since spread fast over the world, resulting in a pandemic. The virus spreads through the respiratory droplets of an infected person. If the patient’s situation is in serious, it can even lead to death, especially to the elderly or those with underlying medical conditions. To prevent the virus from spreading, governments and health organisations around the world have established standard operating procedures (SOP). Moreover, Covid-19 vaccines have also been created and are being disseminated globally to prevent Covid-19 infections.

This assignment aims to develop a Covid-19 management system to perform multi-tasking. For examples, it can categorise the users correctly based on the “Categories of Case Covid-19” guideline, suggests relevant actions to be taken and changes the category of users based on their Covid-19 test. With that, we can make sure the users handle Covid-19 cases in accordance with the standard operating procedure (SOP).

We have designed a Covid-19 management system using Microsoft Visual Studio, which provides a reliable and scalable platform for designing and running the program code. Through a simple and clear user interface, this program enables the Covid-19 Management System to handle Covid-19 cases systematically by following the standard operating procedure (SOP).

This Covid-19 management system can handle a large collection of user information. The system can execute tasks such as registering new users by retrieving the users’ personal information. The registered users can login into the system and the system may perform their profiles correctly. Moreover, the system will keep track of all the users’ categories, and update users’ information time by time.

Our solution includes several additional features such as the system will keep track with the users’ vaccination status, help users to register for the vaccine, display relevant reminders or suggested actions to the users in different categories and others to ensure that users follow the SOP. Additionally, the administrators of the system can view the total number of cases for each category in which the number of cases is up to date. The number of cases for each category is displayed in the table view as well as the horizontal bar graph view. This will provide an organised way of presenting data, making it easier to read. Besides, the administrators of the system can also view all users’ profiles to keep track of all users’ status.

Overall, our solution design intended to provide an effective and user-friendly system for managing Covid-19 cases. We hope this Covid-19 management system can help monitor the spread of the Covid-19 virus so that public health can be protected. The Covid-19 Management System will be further explained in depth in the next section. The explanation of the system will be covered in two parts, which are the register and login sections.

## 1.2 System Modules

### 1.2.1 Module 1: Registration

In the register section, register\_main function is used to register all users. The register\_main function is one of the important functions to get personal data and the status of Covid-19 from the users. In the registration part, the personal information, Covid-19 status, and vaccination information are collected and the user will be classified into different categories of Covid-19 case. After retrieving all the information, the system will save the users’ data to “Biodata.txt” text file.

In the first part of the registration process, personal data such as the phone number, full name, username, NRIC, age, gender, race, home address, and password are collected from the user by using registration function. There are some restrictions for the input of the data. We perform data validation for all personal information of users to ensure the data entered by the users is correct. Each time the data is invalid, an error message will pop out and users will be required to re-enter the data. For the phone number part, a restriction where the number entered must be in 10 or 11 digits is to make sure the phone number is valid. For the full name, the name entered by the users must not contain any digit. Besides, users are asked to provide a username to login to the Covid-19 Management System in the future. Since the Malaysian NRIC length is in 14 letters, a limit of 14 words is set in the NRIC input part. Moreover, the age input limit is set between 0 and 150 to make sure the age is valid. Users are also asked to enter a valid gender, race and address. Lastly, for the password, a minimum of 8 letters are set to ensure the password entered by the users is secure and safe. Users have completed the first step of the registration process up to this point and will now go on to the second.

The second step aims to classify the users into different categories according to their Covid-19 status. In this part, get\_respd and categories functions will be called. The users need to answer yes or no for each question to know the first stage of classification of the category. The first question is asking about the medical condition of the users which has the possible chance to increase the risk of Covid-19. However, the second question is asking whether the users have traveled to any high Covid-19 transmission area recently. Besides, the users also will be asked whether they have attended any Covid-19 cluster-related events or areas and whether the users live or work in an area where the risk of Covid-19 transmission is quite high.

If more than 2 responses are yes, the users will be assigned to the suspected case. The following question being asked is whether the users have tested positive for covid-19 in the past two days. If yes, users will be categorized in category A which is a confirmed case. A message of 7 days of self-quarantine will pop out. The users are asked to quarantine for 7 days and does the self-test every day. If the users are tested negative on day 4, the users will be able to release earlier. If users have not been tested positive, the users will be further asked whether they have stayed with roommates or family members who tested positive for Covid-19. If yes, the users will be categorized into category B which is close contact. The users are advised to have self-quarantine for 3 days. The self-test result should be updated on day 1 and day 3 to prevent the spread of Covid-19. If the users are tested negative on day 3, the users will be released from self-quarantine. If users do not live with roommates or families who tested positive, the users will be categorized into category B which is a suspected case. A reminder will pop out to advise the users to protect themselves from Covid-19. The users need to update their Covid-19 status weekly. By following the standard operating procedure, the users must always wear a face mask and wash their hands frequently. Moreover, the mouth must always be covered when they sneeze or cough.

If the yes responses for the 4 questions in second part are less than 2, the users will be asked whether they have had covid symptoms in the last 14 days which are fever, cough, sore throat, body aches, or shortness of breath. If yes, the users will be assigned to category C which is low risk, but they need to update the self-test result for the next three days. If the answer is no, the users will also be assigned to category C which is low risk. The difference between symptomatic and asymptomatic users is the asymptomatic users do not need to update the self-test result every day. Up to here, users have completed the second step of the registration process.

The third part of the registration is retrieving the vaccination status of the users. This step involves vacStatus function. Initially, the users will be asked whether they have registered for vaccination before. If the users have taken the vaccine before, the system will collect the date of vaccination and the type of dose taken by calling dose\_question function. There are four options for the users to select the type of vaccine which are Pfizer, Sinovac, AstraZeneca, and Others. If the users did not take any vaccine before, they will be asked whether they wish to take the vaccine through register\_vac function. If yes, a successful registration for vaccination message will pop out and the users will be contacted if the appointment for the vaccination is made. If users do not want to register for vaccination, a suggestion message will pop out to let the users know the advantages of taking a vaccine and advise them to take the vaccine. The vacStatus function will continue to ask for second dose date and vaccine type if the users have taken the first dose of vaccine. If the users have not taken second dose of vaccine, they will be led to register\_vac function and same process of registering vaccination is repeated. After the users have provided their second dose information, the system will ask them for their date and vaccine type for third dose of vaccine. If the users have taken three of the vaccines, an appreciation message will be displayed to thank users for protecting themselves.

Once all the data are properly validated, the data entered by the users will be saved to “Biodata.txt” text file and will always be protected to avoid the leakage of confidential data from the users. Further analysis of the data entered will be done such as the total number of confirmed cases will be displayed and the number of users of registration.

### 1.2.2 Module 2: Login

In the login section, login function will be used to allow administrators and users to login the system. We will explain about this section in two parts which are admins and users. If the users choose to login as admin, they will need to enter the correct username and password for admin. If password incorrect, they will need to return to the login menu and login as user. The username and password of admin will only be known by the administrators who handle the system as the admin view might contain the confidentiality and privacy of personal and health information of every user.

The Covid-19 Management System contains administrative features which allow administrators to log in and view the number of Covid-19 cases in various categories such as confirmed case, suspected case, close contact, and low risk. The administrators may also view all users’ profiles to keep track of their status.

If the admin chooses to view the number of Covid-19 cases, the system will read data from the “Biodata.txt” text file which contains all users’ profiles and calculates the number of cases in each category based on the users’ Covid-19 classification. Category A consists of a number of cases from “Confirmed Case” whereas “Close Contact” and “Suspected Case” are considered as category B, and lastly category C consists of “Low Risk” cases. The number of cases in each category is then displayed in a tabular format and horizontal bar graph view. The table presents the number of cases in each category in an organized view while the horizontal bar graph allows the administrators to see the trends of the Covid-19 cases at a glance.

If the admin selects the second option, the system will display all registered users’ profiles. Each user’s name, age, IC number, gender, phone number, race, address, Covid-19 classification, and vaccination status are all displayed in each profile. With that, administrators can monitor the movement of each user in category A and B. Administrators can contact users who do not update their Covid-19 status and give warning to them so that they will follow the Standard Operating Procedure (SOP).

Move on to login as user part, the login function allows users to log into the system as a user, provided that the users registered their personal information before login. This action can be performed by calling chk\_user function. The system will need to read the users’ data from “Biodata.txt” text file. If the username entered by the users is not registered in the “Biodata.txt” text file, an error message will pop up and ask the users to register in main menu before login. If the username is found inside the text file, the user is prompted to enter the respective password for further confirmation. The entered password is then checked against the text file. If the password entered matches the username in the text file, a statement indicating a successful login will be displayed. If the entered password is incorrect, a message indicating that the password is incorrect will be displayed, and the user will be allowed to reset their password if they have forgotten it. To reset their password, a random number will be displayed as a reCAPTCHA, and the users need to enter the displayed number. If the entered number does not match the number displayed, a new random number will be displayed, and the user will be prompted to enter it again. If the entered number is correct, the user will then be led to change their password by entering a new one, which will be updated in the text file.

Once the users have successfully logged into the system, the system displays a menu with two options which to allow the users to update their Covid-19 status and to review their own profile, which includes their personal information such as name, age, NRIC, gender, phone number, race, address, Covid-19 status, and vaccine dose date and type. To update their Covid-19 status, update function will be called, and users need to answer the Covid-19 questions again so that the system can re-categorize the users to new Covid-19 category.

With that, this sums up the entire Covid-19 Management System we have developed.

## 1.3 Structure Chart

Diagram, Teams

Description automatically generated

## 1.4 Flowchart

main()

Diagram

Description automatically generated

register\_main()

A picture containing diagram

Description automatically generated

registration()

Diagram

Description automatically generated

registration()

Diagram

Description automatically generated

registration()

Diagram

Description automatically generated

registration()

Diagram

Description automatically generated

registration()

A picture containing text

Description automatically generated

get\_respd(\* ptr\_respd1, \* ptr\_respd2, \* ptr\_respd3, \* ptr\_respd4)

A picture containing text

Description automatically generated

categories(respd1, respd2, respd3, respd4, size)

Diagram, application

Description automatically generated

categories(respd1, respd2, respd3, respd4, size)

Diagram

Description automatically generated

categories(respd1, respd2, respd3, respd4, size)

A picture containing diagram

Description automatically generated

vacStatus()

Diagram

Description automatically generated

vacStatus()

Diagram

Description automatically generated

dose\_question(doseNum)

Diagram

Description automatically generated

dose\_question(doseNum)

Diagram

Description automatically generated

register\_vac(doseNum)

A picture containing diagram

Description automatically generated

login()

A picture containing diagram

Description automatically generated

login()

Diagram

Description automatically generated

login()

Diagram

Description automatically generated

login()

Diagram

Description automatically generated

login()

Diagram

Description automatically generated

login()

A picture containing diagram

Description automatically generated

chk\_user(\* index, i)

Diagram

Description automatically generated

chk\_user(\* index, i)

Diagram

Description automatically generated

chk\_user(\* index, i)

A picture containing diagram

Description automatically generated

update(size)

Letter

Description automatically generated

# 2 C++ Program

/\*

G1\_Kuik Rui Luan

Compiler: Microsoft Visual Studio

\*/

// Library Function

#include <iostream>

#include <cstring>

#include <cctype>

#include <string>

#include <iomanip>

#include <fstream>

#include <ctime>

#include <cstdlib>

using namespace std;

//Define number of question in get\_respd function

#define QUES\_NUM 4

// Declare USER structure type that represents all the biodata information of users

struct USER {

char name[100];

char phone\_num[100];

char nric[100];

char age[100];

char gender[100];

char race[100];

char address[100];

char username[100];

char password[100];

char classification[100];

char first\_dose\_date[100];

char first\_dose\_type[100];

char sec\_dose\_date[100];

char sec\_dose\_type[100];

char third\_dose\_date[100];

char third\_dose\_type[100];

};

USER user[500];

//Function Prototypes

//Register function prototypes

void register\_main(void);

void registration(void);

//Vaccination Status Function Prototype

void vacStatus(void);

void dose\_question(int doseNum);

void register\_vac(int doseNum);

//Login Function Prototype

void login(void);

void chk\_user(int\* index, int i);

//Covid Question Function Prototype

void get\_respd(int\* ptr\_respd1, int\* ptr\_respd2, int\* ptr\_respd3, int\* ptr\_respd4);

void categories(int respd1, int respd2, int respd3, int respd4, int size);

//Update Function Prototype

void update(int index);

// Main menu

int main(void)

{

bool loop = true;

do

{

system("cls");

cout << "-------------------------- \n";

cout << " COVID-19 MANAGEMENT \n";

cout << "--------------------------\n\n";

cout << "--MAIN MENU-- \n\n";

cout << "1. LOGIN \n";

cout << "2. REGISTER \n";

cout << "3. QUIT \n\n";

char choice;

// ask user to select an option from the main menu

cout << "SELECT AN OPTION: ";

cin >> choice;

if (choice == '1')

{

// if choice 1 is selected, login function will be called out

login();

}

else if (choice == '2')

{

/\* if choice 2 is selected, register function will be called out\*/

register\_main();

}

else if (choice == '3')

{

/\*

if choice 3 is selected, system will shut down and the loop will be terminated.

\*/

cout << "System is shutting down...." << endl;

loop = false;

}

else

{

/\*

if other than 1,2,3 is entered, system will show invalid option and ask user to select an option again.

\*/

cout << "\nInvalid Option\n\n" << endl;

}

} while (loop);

return 0;

}

/\*

Register function which asks users to register their biodata, covid, and vaccination information

\*/

void register\_main(void)

{

system("cls");

cout << "=====================================================" << endl;

cout << " Register Here " << endl;

cout << "=====================================================" << endl;

cout << "\n";

char choice, proceed;

cout << "Do you wish to register? [Y]: ";

cin >> choice;

//change to uppercase before check the input

choice = toupper(choice);

if (choice == 'Y')

{

/\*

call registration function asks user to register their personal information

\*/

system("cls");

registration();

cout << "\nPress any key except ENTER to proceed to provide your Covid-19 information ..." << endl;

cin >> proceed;

/\*

if users agree to proceed, classify users to different covid - 19 categories by asking some covid - 19 question

Pre: users need to answer the personal information before answer the covid-19 question

\*/

if (proceed)

{

system("cls");

int respd1, respd2, respd3, respd4;

get\_respd(&respd1, &respd2, &respd3, &respd4);

categories(respd1, respd2, respd3, respd4, 0);

cout << "Press any key except ENTER to proceed to provide your vaccination infomation ..." << endl;

cin >> proceed;

/\*

if users agree to proceed, ask users about their vaccination information

Pre: users need to answer the personal information and covid-19 question before provide their vaccination information

\*/

if (proceed)

{

system("cls");

vacStatus();

//save all the data in the file after all information is provided

ofstream out\_file("Biodata.txt", ios::app);

if (!out\_file)

cout << "Error opening output file";

else

{

out\_file << user[0].name << "|" << user[0].phone\_num << "|" << user[0].nric << "|" << user[0].age << "|"

<< user[0].gender << "|" << user[0].race << "|" << user[0].address << "|" << user[0].username << "|"

<< user[0].password << "|" << user[0].classification << "|" << user[0].first\_dose\_date << "|" << user[0].first\_dose\_type << "|" << user[0].sec\_dose\_date << "|" << user[0].sec\_dose\_type << "|" << user[0].third\_dose\_date << "|" << user[0].third\_dose\_type << "\n";

cout << "\n";

cout << "Your data is saved. Thanks for registration." << endl;

}

out\_file.close();

}

}

}

else

cout << "Invalid input!" << endl;

cout << "Press any key except ENTER to proceed to return main menu ..." << endl;

cin >> proceed;

}

//Register function which asks users to register their biodata information

void registration(void)

{

cout << "=====================================================" << endl;

cout << " Register Here " << endl;

cout << "=====================================================" << endl;

cout << "\n";

//ask the user to enter the phone number

if (cin.peek() == '\n')

cin.ignore(256, '\n');

do

{

cout << "\nEnter the phone number: ";

cin.getline(user[0].phone\_num, 100);

fflush(stdin);

/\*check whether it is a number and the length of the phone number must be in 10 or 11\*/

if (strlen(user[0].phone\_num) != 10 && strlen(user[0].phone\_num) != 11)

{

cout << "Invalid phone number entered." << endl;

}

} while (strlen(user[0].phone\_num) != 10 && strlen(user[0].phone\_num) != 11);

//ask the user to enter the full name

bool validName;

do

{

cout << "\nEnter the full name: ";

cin.getline(user[0].name, 100);

fflush(stdin);

validName = true;

for (int i = 0; i < strlen(user[0].name); i++)

{

user[0].name[i] = toupper(user[0].name[i]);

/\*check the name entered by the user is valid and not contain number\*/

if (isdigit(user[0].name[i]))

{

validName = false;

break;

}

}

if (!validName)

{

cout << "Invalid name entered." << endl;

}

} while (!validName);

//ask the user to enter username

do

{

cout << "\nEnter the username: ";

cin.getline(user[0].username, 100);

fflush(stdin);

/\*check the length of the input is not equal to zero to make sure the username is entered\*/

if (strlen(user[0].username) == 0)

{

cout << "Invalid usename entered." << endl;

}

} while (strlen(user[0].username) == 0);

//ask the user to enter nric

do

{

cout << "\nEnter NRIC [xxxxxx-xx-xxxx]: ";

cin.getline(user[0].nric, 100);

fflush(stdin);

//check the length of the nric whether is 14

if (strlen(user[0].nric) != 14)

{

cout << "Invalid NRIC entered." << endl;

}

} while (strlen(user[0].nric) != 14);

//ask the user to enter the age

do

{

cout << "\nEnter the age: ";

cin.getline(user[0].age, 100);

fflush(stdin);

//check the age entered is between 0-150

if (atoi(user[0].age) < 0 || atoi(user[0].age) > 150)

{

cout << "Invalid age entered." << endl;

}

} while (atoi(user[0].age) < 0 || atoi(user[0].age) > 150);

//ask the user to enter the gender

do

{

cout << "\nEnter gender Male[M] Female [F]: ";

cin.getline(user[0].gender, 100);

for (int i = 0; user[0].gender[i] != '\0'; i++) {

user[0].gender[i] = toupper(user[0].gender[i]);

}

fflush(stdin);

//check the gender entered by the user is M or F

if (strcmp(user[0].gender, "M") != 0 && strcmp(user[0].gender, "F") != 0)

{

cout << "Invalid gender entered." << endl;

}

} while (strcmp(user[0].gender, "M") != 0 && strcmp(user[0].gender, "F") != 0);

//ask the user to enter the race

do

{

cout << "\n----------------------------------------------------------------" << endl;

cout << "| Malay [M] Chinese [C] Indian [I] |" << endl;

cout << "| Bumiputera Sabah [BSH] Bumiputera Sarawak [BSK] Others [O] |" << endl;

cout << "----------------------------------------------------------------" << endl;

cout << "\nEnter the race: ";

cin.getline(user[0].race, 100);

for (int i = 0; user[0].race[i] != '\0'; i++) {

user[0].race[i] = toupper(user[0].race[i]);

}

fflush(stdin);

//check the race entered is in the options

if (!(strcmp(user[0].race, "M") == 0 || strcmp(user[0].race, "C") == 0 || strcmp(user[0].race, "I") == 0 || strcmp(user[0].race, "BSH") == 0 || strcmp(user[0].race, "BSK") == 0 || strcmp(user[0].race, "O") == 0))

{

cout << "Invalid race entered." << endl;

}

} while (!(strcmp(user[0].race, "M") == 0 || strcmp(user[0].race, "C") == 0 || strcmp(user[0].race, "I") == 0 || strcmp(user[0].race, "BSH") == 0 || strcmp(user[0].race, "BSK") == 0 || strcmp(user[0].race, "O") == 0));

//ask the user to enter home address

do

{

cout << "\nEnter the home address: ";

cin.getline(user[0].address, 100);

fflush(stdin);

//check address entered is not equal to zero

if (strlen(user[0].address) == 0)

{

cout << "Invalid home address entered." << endl;

}

} while (strlen(user[0].address) == 0);

//ask the user to enter password

do

{

cout << "\nEnter the password of at least 8 characters or numbers: ";

cin.getline(user[0].password, 100);

fflush(stdin);

//check the password is in the length of 8

if (strlen(user[0].password) < 8)

{

cout << "Invalid password entered. Please enter at least 8 characters or numbers." << endl;

}

} while (strlen(user[0].password) < 8);

}

/\*

Login function which allows users to update covid-19 status or show profile

Login function which allows admin to view the number of cases in each category

Pre: users must register their personal information before login

\*/

void login(void)

{

char proceed;

bool loop = true;

while (loop)

{

char opt;

int i = 0;

int index = 0;

//open biodata text file to view user data

ifstream in\_file("Biodata.txt");

if (!in\_file)

cout << "Error opening input file";

//read data from the text file

else

{

while (in\_file.getline(user[i].name, 100, '|'))

{

in\_file.getline(user[i].phone\_num, 100, '|');

in\_file.getline(user[i].nric, 100, '|');

in\_file.getline(user[i].age, 100, '|');

in\_file.getline(user[i].gender, 100, '|');

in\_file.getline(user[i].race, 100, '|');

in\_file.getline(user[i].address, 100, '|');

in\_file.getline(user[i].username, 100, '|');

in\_file.getline(user[i].password, 100, '|');

in\_file.getline(user[i].classification, 100, '|');

in\_file.getline(user[i].first\_dose\_date, 100, '|');

in\_file.getline(user[i].first\_dose\_type, 100, '|');

in\_file.getline(user[i].sec\_dose\_date, 100, '|');

in\_file.getline(user[i].sec\_dose\_type, 100, '|');

in\_file.getline(user[i].third\_dose\_date, 100, '|');

in\_file.getline(user[i].third\_dose\_type, 100);

i++;

}

}

system("cls");

cout << "-------------------------------------------------\n";

cout << " Log In \n";

cout << "-------------------------------------------------\n";

cout << "1. Login as admin\n2. Login as user\n0. Exit\n" << endl;

cout << "Enter option: ";

cin >> opt;

//login as admin to view number of cases in each category and view users' profiles

if (opt == '1')

{

char admin[50];

cout << "Username: ";

cin >> admin;

if (strcmp("admin", admin) == 0)

{

char adminpass[50];

cout << "Password: ";

cin >> adminpass;

if (strcmp("hereisadmin", adminpass) == 0)

{

system("cls");

char adminopt;

cout << "Login as admin successfully!" << endl;

cout << "\n1. Display number of cases in each category" << endl;

cout << "2. View all users profile" << endl;

cout << "\nSelect an option: ";

cin >> adminopt;

if (adminopt == '1')

{

int catA = 0;

int catB = 0;

int catC = 0;

string line;

ifstream in\_file("Biodata.txt");

if (!in\_file)

cout << "Error opening input file";

else

{

/\*compute the number of cases in each category\*/

while (getline(in\_file, line)) {

if ((line.find("CONFIRMED CASE")) != string::npos) {

catA++;

}

else if ((line.find("SUSPECTED CASE")) != string::npos ||

(line.find("CLOSE CONTACT")) != string::npos) {

catB++;

}

else if ((line.find("LOW RISK")) != string::npos) {

catC++;

}

}

}

in\_file.close();

/\*display the number of case in each category in table view\*/

system("cls");

cout << "-----------------------------------------------------------" << endl;

cout << " Category Table" << endl;

cout << "-----------------------------------------------------------" << endl;

cout << "\n\n";

cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "|" << left << setw(10) << "Category|" << setw(25) << "Description" << setw(15) << "| Number of Case |" << endl;

cout << "|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "|" << setw(8) << " A |" << setw(26) << "CONFIRMED CASE" << setw(5) << "|\t" << catA << "\t\t|" << endl;

cout << "|" << setw(8) << " B |" << setw(26) << "SUSPECTED CASE," << setw(5) << "|\t\t\t|" << endl;

cout << "|" << setw(8) << " |" << setw(26) << "CLOSE CONTACT" << setw(5) << "|\t" << catB << "\t\t|" << endl;

cout << "|" << setw(8) << " C |" << setw(26) << "LOW RISK" << setw(5) << "|\t" << catC << "\t\t|" << endl;

cout << "|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

/\*display number of case in each category in horizontal bar graph view\*/

cout << "\n\n\n";

cout << "------------------------------------------------------" << endl;

cout << " Horizontal Bar Graph" << endl;

cout << "------------------------------------------------------" << endl;

cout << "\n\n";

cout << "\nA ";

for (int row = 0; row < 2; row++)

{

for (int i = 0; i < catA; i++)

{

cout << "\*";

}

cout << endl;

cout << " ";

}

cout << "\nB ";

for (int row = 0; row < 2; row++)

{

for (int i = 0; i < catB; i++)

{

cout << "\*";

}

cout << endl;

cout << " ";

}

cout << "\nC ";

for (int row = 0; row < 2; row++)

{

for (int i = 0; i < catC; i++)

{

cout << "\*";

}

cout << endl;

cout << " ";

}

cout << endl;

}

else if (adminopt == '2')

{

system("cls");

for (int j = 0; j < i; j++)

{

cout << "User " << j + 1 << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "|" << right << setw(20) << "Name: " << setw(30) << user[j].name << "|" << setw(20) << "Age: " << setw(30) << user[j].age << "|" << endl;

cout << "|" << setw(20) << "IC: " << setw(30) << user[j].nric << "|" << setw(20) << "Gender: " << setw(30) << user[j].gender << "|" << endl;

cout << "|" << setw(20) << "Phone Number: " << setw(30) << user[j].phone\_num << "|" << setw(20) << "Race: " << setw(30) << user[j].race << "|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "|" << setw(20) << "Address: " << setw(60) << user[j].address << " |" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "|" << setw(20) << "Status: " << setw(30) << user[j].classification << "|" << setw(51) << "|" << endl;

cout << "|" << setw(20) << "First Dose Date: " << setw(30) << user[j].first\_dose\_date << "|" << setw(20) << "First Dose Type: " << setw(30) << user[j].first\_dose\_type << "|" << endl;

cout << "|" << setw(20) << "Second Dose Date: " << setw(30) << user[j].sec\_dose\_date << "|" << setw(20) << "Second Dose Type: " << setw(30) << user[j].sec\_dose\_type << "|" << endl;

cout << "|" << setw(20) << "Third Dose Date: " << setw(30) << user[j].third\_dose\_date << "|" << setw(20) << "Third Dose Type: " << setw(30) << user[j].third\_dose\_type << "|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

}

}

else

{

cout << "Invalid input!" << endl;

}

}

else

{

cout << "Password incorrect!" << endl;

}

}

else

{

cout << "You are not an admin! Please login as user." << endl;

}

cout << "\nPress any key except ENTER to proceed to return login menu ..." << endl;

cin >> proceed;

}

//login as user

else if (opt == '2')

{

chk\_user(&index, i);

if (index != 0)

{

system("cls");

char choice1;

cout << "Login Successful !" << endl;

cout << "\n1. Update Covid-19 Status " << endl;

cout << "2. Show profile" << endl;

cout << "\nSelect an option: ";

cin >> choice1;

//update Covid-19 status

if (choice1 == '1')

{

int position = index - 1;

update(position);

ofstream out\_file("Biodata.txt");

if (!out\_file)

cout << "Error opening output file";

else

{

for (int j = 0; j < i; j++)

{

out\_file << user[j].name << "|" << user[j].phone\_num << "|" << user[j].nric << "|" << user[j].age << "|" << user[j].gender << "|" << user[j].race << "|" << user[j].address << "|" << user[j].username << "|" << user[j].password << "|" << user[j].classification << "|" << user[j].first\_dose\_date << "|" << user[j].first\_dose\_type << "|" << user[j].sec\_dose\_date << "|" << user[j].sec\_dose\_type << "|" << user[j].third\_dose\_date << "|" << user[j].third\_dose\_type << "\n";

}

}

out\_file.close();

}

//display user's profile

else if (choice1 == '2')

{

system("cls");

cout << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "|" << right << setw(20) << "Name: " << setw(30) << user[index - 1].name << "|" << setw(20) << "Age: " << setw(30) << user[index - 1].age << "|" << endl;

cout << "|" << setw(20) << "IC: " << setw(30) << user[index - 1].nric << "|" << setw(20) << "Gender: " << setw(30) << user[index - 1].gender << "|" << endl;

cout << "|" << setw(20) << "Phone Number: " << setw(30) << user[index - 1].phone\_num << "|" << setw(20) << "Race: " << setw(30) << user[index - 1].race << "|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "|" << setw(20) << "Address: " << setw(60) << user[index - 1].address << " |" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "|" << setw(20) << "Status: " << setw(30) << user[index - 1].classification << "|" << setw(51) << "|" << endl;

cout << "|" << setw(20) << "First Dose Date: " << setw(30) << user[index - 1].first\_dose\_date << "|" << setw(20) << "First Dose Type: " << setw(30) << user[index - 1].first\_dose\_type << "|" << endl;

cout << "|" << setw(20) << "Second Dose Date: " << setw(30) << user[index - 1].sec\_dose\_date << "|" << setw(20) << "Second Dose Type: " << setw(30) << user[index - 1].sec\_dose\_type << "|" << endl;

cout << "|" << setw(20) << "Third Dose Date: " << setw(30) << user[index - 1].third\_dose\_date << "|" << setw(20) << "Third Dose Type: " << setw(30) << user[index - 1].third\_dose\_type << "|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

in\_file.close();

}

else

{

cout << "Invalid input!" << endl;

}

}

cout << "\nPress any key except ENTER to return login menu ..." << endl;

cin >> proceed;

}

//return to main menu

else if (opt == '0')

{

loop = false;

}

else

{

cout << "Invalid input!\n" << endl;

cout << "\nPress any key except ENTER to proceed to return login menu ..." << endl;

cin >> proceed;

}

}

}

/\*

Ask for users' vaccination information about dose 1, 2 and 3

If users do not register for vaccination, ask users if the users want to register for receiving vaccination

Pre: users must register their personal information and answer the Covid-19 question before providing vaccination information

\*/

void vacStatus(void)

{

cout << "-------------------------------------------------\n";

cout << " Vaccination Status \n";

cout << "-------------------------------------------------\n";

int i = 0;

char regVac, ansdose;

int doseNum = 1;

bool loop = true;

while (loop)

{

do

{

cout << "Have you registered for vaccination?" << endl;

cout << "1. Yes\n2. No\n0. Exit" << endl;

cout << "Enter your answer: ";

cin >> regVac;

if (regVac == '0')

break;

//enter dose one information

else if (regVac == '1')

{

system("cls");

cout << "Please key in the information of your dose 1.\n";

dose\_question(doseNum);

doseNum++;

cout << "\nHave you taken dose " << doseNum << "? " << endl;

cout << "1. Yes\n2. No\n";

cout << "\nEnter your answer: ";

cin >> ansdose;

}

//ask user to register for dose one

else if (regVac == '2')

{

register\_vac(doseNum);

}

else

cout << "Invalid input!" << endl;

} while (regVac != '1' && regVac != '2' && regVac != '0');

/\*ask user about dose two information after user has provided dose one information\*/

if (doseNum == 2)

{

do

{

//enter dose two information

if (ansdose == '1')

{

system("cls");

dose\_question(doseNum);

doseNum++;

cout << "\nHave you taken dose " << doseNum << "? " << endl;

cout << "1. Yes\n2. No\n";

cout << "\nEnter your answer: ";

cin >> ansdose;

}

//ask user to register for dose two

else if (ansdose == '2')

{

register\_vac(doseNum);

}

/\*ask again the question if the users key in invalid input\*/

else

{

cout << "Invalid input!" << endl;

cout << "\nHave you taken dose " << doseNum << "? " << endl;

cout << "1. Yes\n2. No\n" << endl;

cout << "\nEnter your answer: ";

cin >> ansdose;

}

} while (ansdose != '1' && ansdose != '2');

}

/\*ask user about dose three information after user has provided dose one and two information\*/

if (doseNum == 3)

{

do

{

//enter dose three information

if (ansdose == '1')

{

system("cls");

dose\_question(doseNum);

cout << "\nThank you for protecting yourself and the community from Covid-19!" << endl;

cout << "Together we stand, together we fight!\n" << endl;

}

//ask user to register for dose three

else if (ansdose == '2')

{

register\_vac(doseNum);

}

/\*ask again the question if the users key in invalid input\*/

else

{

cout << "Invalid input!" << endl;

cout << "\nHave you taken dose " << doseNum << "? " << endl;

cout << "1. Yes\n2. No\n";

cout << "\nEnter your answer: ";

cin >> ansdose;

}

} while (ansdose != '1' && ansdose != '2');

}

loop = false;

}

}

/\*

Ask for users' either 1, 2 or 3 dose date and vaccine type they have received

Display the dose question with dose number, doseNum accordingly

\*/

void dose\_question(int doseNum)

{

int vac\_type, day, month, year;

cout << "Dose " << doseNum << ": " << endl;

do {

cout << "Enter the date of your dose (DD MM YYYY): ";

cin >> day >> month >> year;

//validate whether the date entered is in correct format

if (day > 31 || month > 12 || !(year >= 2020 && year <= 2023)) {

cout << "Invalid date! Please key in the date in correct format." << endl;

}

//assign the day, month and year to 1, 2 or 3 dose date

else

{

if (doseNum == 1)

{

string date = to\_string(day) + "/" + to\_string(month) + "/" + to\_string(year);

strcpy\_s(user[0].first\_dose\_date, date.c\_str());

}

else if (doseNum == 2)

{

string date = to\_string(day) + "/" + to\_string(month) + "/" + to\_string(year);

strcpy\_s(user[0].sec\_dose\_date, date.c\_str());

}

else

{

string date = to\_string(day) + "/" + to\_string(month) + "/" + to\_string(year);

strcpy\_s(user[0].third\_dose\_date, date.c\_str());

}

}

} while (day > 31 || month > 12 || !(year >= 2020 && year <= 2023));

string vacTypeLst[4] = { "AstraZenca", "Pfizer", "Sinovac", "Others" };

do {

cout << "\nEnter the vaccine type\n<0> AstraZenca\n<1> Pfizer\n<2> Sinovac\n<3> Others\n";

cout << "Please enter the digit: ";

cin >> vac\_type;

//validate whether users enter valid vaccine type

if (vac\_type < 0 || vac\_type > 3) {

cout << "Invalid vaccine type! Please key in the digit according to your vaccine type.\n" << endl;

}

//assign the vaccine type to 1, 2 or 3 dose type

else

{

if (doseNum == 1)

{

const char\* vacType = vacTypeLst[vac\_type].c\_str();

strcpy\_s(user[0].first\_dose\_type, vacType);

}

else if (doseNum == 2)

{

const char\* vacType = vacTypeLst[vac\_type].c\_str();

strcpy\_s(user[0].sec\_dose\_type, vacType);

}

else

{

const char\* vacType = vacTypeLst[vac\_type].c\_str();

strcpy\_s(user[0].third\_dose\_type, vacType);

}

cout << vacTypeLst[vac\_type] << endl;

cout << "\n";

}

} while (vac\_type < 0 || vac\_type > 3);

}

/\*

Ask users whether the users want to register for 1, 2 or 3 dose

Display the register vaccine question with dose number, doseNum accordingly

\*/

void register\_vac(int doseNum)

{

int reg\_ans;

do

{

cout << "\nDo you want to register for dose " << doseNum << "?" << endl;

cout << "1. Yes\n2. No\n";

cout << "Enter your answer: ";

cin >> reg\_ans;

//Display successful registration statements

if (reg\_ans == 1)

{

cout << "\nThanks for your registration! We will contact you in a few days for further confirmation." << endl;

}

/\*Display information and importance of vaccination to encourage users to register for vaccination\*/

else if (reg\_ans == 2)

{

cout << "\nWhat You Need To Know About COVID-19 Vaccine\n\n";

cout << "Getting vaccinated is one of the most crucial steps in protecting yourself and\n";

cout << "your loved ones from COVID-19. It is also to achieve herd immunity for the greater good.\n";

cout << "Studies have shown that getting vaccinated against COVID-19 can lower your risk of\n";

cout << "getting and spreading the virus that causes COVID-19. Vaccines also help reduce the risk of\n";

cout << "severe illnesses and deaths from COVID-19 infections among people who are fully vaccinated.\n\n";

cout << "Your loved ones need you.\n";

cout << "Get your COVID-19 vaccine to make sure you can be there for them!\n";

}

else

cout << "Invalid input!\n";

} while (reg\_ans != 1 && reg\_ans != 2);

}

/\*

Check whether username is registered before

If username is valid, validate the password is correctly match with the password registered before

Allow users to re-enter the password if the users forgot their password

\*/

void chk\_user(int\* index, int i)

{

char proceed;

bool loginSuccessful = false;

while (!loginSuccessful) {

char username[50];

cout << "Username: ";

cin >> username;

bool usernameFound = false;

for (int pos = 0; pos < i; pos++) {

/\*check whether the username entered is registered in the biodata text file\*/

if (strcmp(user[pos].username, username) == 0)

{

usernameFound = true;

char password[50];

cout << "Password: ";

cin >> password;

/\*validate the password entered is matched correctly with username in biodata text file\*/

if (strcmp(user[pos].password, password) == 0)

{

cout << "Login successfully!" << endl;

loginSuccessful = true; // exit the loop since login is successful

\*index = pos + 1;

}

else

{

cout << "\nPassword incorrect!" << endl;

char forgotpass;

cout << "\n\nForgot password? [Y]es [N]o : ";

cin >> forgotpass;

if (forgotpass == 'y' || forgotpass == 'Y')

{

int random\_number\_entered;

int random\_number = 0;

/\*use random number to display recaptcha and ask user to enter the recaptcha to ensure the user is human\*/

do

{

srand(time(NULL));

random\_number = rand() % 1000 + 1;

cout << random\_number << endl;

cout << "\nEnter the number that show on screen: ";

cin >> random\_number\_entered;

if (random\_number\_entered != random\_number)

cout << "Please retry...\n";

} while (random\_number != random\_number\_entered);

//ask user to enter new password

cout << "\nNew password: ";

if (cin.peek() == '\n')

cin.ignore(256, '\n');

cin.getline(user[pos].password, 100);

//update biodata text file

ofstream out\_file("Biodata.txt", ios::out);

if (!out\_file)

cout << "Error opening output file";

else

{

for (int j = 0; j < i; j++)

{

out\_file << user[j].name << "|" << user[j].phone\_num << "|" << user[j].nric << "|" << user[j].age << "|" << user[j].gender << "|" << user[j].race << "|" << user[j].address << "|" << user[j].username << "|" << user[j].password << "|" << user[j].classification << "|" << user[j].first\_dose\_date << "|" << user[j].first\_dose\_type << "|" << user[j].sec\_dose\_date << "|" << user[j].sec\_dose\_type << "|" << user[j].third\_dose\_date << "|" << user[j].third\_dose\_type << endl;

}

}

out\_file.close();

cout << "Password changed! You may log in again with the new password. \n" << endl;

}

else

cout << "Please login again.\n" << endl; /\* exit the loop to login again since username was found, but password was incorrect\*/

}

}

}

if (!usernameFound) {

cout << "Username not found! Please return to the main menu to register an account.\n" << endl;

loginSuccessful = true; // exit the loop since username was not found

}

}

}

/\*

Ask users about 4 Covid-19 questions and receive 4 answers from users

4 responses from users are returned with pass-by-address method using pointers

\*/

void get\_respd(int\* ptr\_respd1, int\* ptr\_respd2, int\* ptr\_respd3, int\* ptr\_respd4)

{

cout << "\n\nPlease answer the following question.\n";

cout << "YES[1] NO[2]\n\n";

cout << "1. Do you have any underlying medical conditions that may increase your risk of severe illness from COVID-19, such as heart disease, diabetes, or lung disease? : ";

cin >> \*ptr\_respd1;

cout << "\n\n2. Have you recently traveled to an area with high COVID-19 transmission rates? : ";

cin >> \*ptr\_respd2;

cout << "\n\n3. Have you been to any COVID-19 cluster related events or areas? : ";

cin >> \*ptr\_respd3;

cout << "\n\n4. Do you live or work in an area where COVID-19 transmission is a high danger, such as a nursing home or healthcare facility? : ";

cin >> \*ptr\_respd4;

}

/\*

Calculate the number of 'yes' response and 'no' response from the users in previous get\_respd function

Use switch statements to count the responses

After the responses are calculated, users need to answer more questions so that the system can classify users to correct categories

Reminders will be generated to remind users follow SOP

\*/

void categories(int respd1, int respd2, int respd3, int respd4, int size)

{

char proceed;

int yes = 0;

int no = 0;

string classLst[4] = { "CONFIRMED CASE","SUSPECTED CASE","CLOSED CONTACT","LOW RISK" };

switch (respd1)

{

case 1:

yes++;

break;

case 2:

no++;

break;

default:

cout << "Invalid input.";

break;

}

switch (respd2)

{

case 1:

yes++;

break;

case 2:

no++;

break;

default:

cout << "Invalid input.";

break;

}

switch (respd3)

{

case 1:

yes++;

break;

case 2:

no++;

break;

default:

cout << "Invalid input.";

break;

}

switch (respd4)

{

case 1:

yes++;

break;

case 2:

no++;

break;

default:

cout << "Invalid input.";

break;

}

/\*

Categorise users to category A - Confirmed case or category B - Suspected case / Close contact

If users have been tested positive, users is categorised to category A and users are asked to quanrantine 7 days

If users not tested positive but having symptoms or close contact, users are categorised to category B and users are asked to quarantine 3 days

If users have not been tested positive and do not have any symptom and do not close contact, users are under category B also but users do not need to quarantine

\*/

if (yes >= 2)

{

int respd\_catA;

cout << "You have been assigned to suspected case category, please answer the following question" << endl;

cout << "\nHave you been tested positive for Covid-19 in the past two days? Yes[1], No[2]" << endl;

cin >> respd\_catA;

if (respd\_catA == 1)

{

cout << "\n---------------------------------------------------------------------------------------------------------------------";

cout << "\nYou are in category A -- CONFIRMED CASE" << endl;

cout << "\nPlease self-quarantine for 7 days to safeguard others around you and prevent the spread of COVID-19." << endl;

cout << "\nReminder: Please update your COVID-19 self-test result everyday for the next 7 days." << endl;

cout << "\n\*NOTE\* <You might be released earlier if you have tested negative on Day 4.>" << endl;

const char\* classification = classLst[0].c\_str();

strcpy\_s(user[size].classification, classification);

}

else

{

int respd\_catB1;

int respd\_catB2;

cout << "\nHave you had any COVID-19 symptoms in the last 14 days, such as fever, cough, sore throat, bodyache, or shortness of breath?" << endl;

cout << "YES[1] NO[2]\n";

cout << "\nAns: ";

cin >> respd\_catB1;

if (respd\_catB1 == 1)

{

cout << "\n---------------------------------------------------------------------------------------------------------------------";

cout << "\nYou are in category B -- SUSPECTED CASE" << endl;

cout << "\nPlease self-quarantine for 3 days to safeguard others around you and prevent the spread of COVID-19." << endl;

cout << "\nReminder: Please update your COVID-19 self-test result on day 1 and day 3." << endl;

cout << "\n\*NOTE\* <You may be released from quarantine if your COVID-19 self-test result on day 3 is negative and symptoms have improved.>" << endl;

cout << "\n\*NOTE\* <You may consult with a doctor if you continue to have symptoms on day 3 even if the COVID-19 self-test result is negative.>" << endl;

const char\* classification = classLst[1].c\_str();

strcpy\_s(user[size].classification, classification);

}

else

{

cout << "\nDo you staying with roommates or family members who tested positive Covid-19?" << endl;

cout << "Yes[1] No[2]\n";

cout << "\nAns: ";

cin >> respd\_catB2;

if (respd\_catB2 == 1)

{

cout << "\n---------------------------------------------------------------------------------------------------------------------";

cout << "\nYou are suspected to category B -- CLOSE CONTACT" << endl;

cout << "\nPlease self-quarantine for 3 days to safeguard others around you and prevent the spread of COVID-19." << endl;

cout << "\nReminder: Please update your COVID-19 self-test result on day 1 and day 3." << endl;

cout << "\n\*NOTE\* <You may be released from quarantine if your COVID-19 self-test result on day 3 is negative.>" << endl;

const char\* classification = classLst[2].c\_str();

strcpy\_s(user[size].classification, classification);

}

else

{

cout << "\n---------------------------------------------------------------------------------------------------------------------";

cout << "\nYou are in category B -- SUSPECTED CASE" << endl;

cout << "\nReminder: Please update your COVID-19 status weekly." << endl;

cout << "\n\n!!It is important to take certain precautions to protect yourself from COVID-19." << endl;

cout << "\nHere are some advices to follow:" << endl;

cout << "# Wash your hands frequently with soap and water for at least 20 seconds, or use hand sanitizer if you are unable to wash your hands." << endl;

cout << "# When coughing or sneezing, keep your hands away from your face and cover your mouth and nose with a tissue or your elbow." << endl;

cout << "# Wear a well-fitting mask in public or among others to help prevent the spread of the virus." << endl;

cout << "# Follow all standard operating procedure (SOP) established by your local government or health officials." << endl;

const char\* classification = classLst[1].c\_str();

strcpy\_s(user[size].classification, classification);

}

}

}

}

/\*

Categorise users to category C - Low Risk

If users have COVID-19 symptoms, users do not need to quarantine and are allowed to work and study as usual if COVID-19 self-test result is negative

If users do not have COVID-19 symptoms, users do not need to quarantine and are allowed to work and study as usual

\*/

else

{

int respd\_catC;

const char\* classification = classLst[3].c\_str();

strcpy\_s(user[size].classification, classification);

cout << "\nHave you had any COVID-19 symptoms in the last 14 days, such as fever, cough, sore throat, bodyache, or shortness of breath? : \n";

cout << "YES[1] NO[2]\n";

cout << "Ans: ";

cin >> respd\_catC;

if (respd\_catC == 1)

{

cout << "\n---------------------------------------------------------------------------------------------------------------------";

cout << "\nYou are in category C -- LOW RISK" << endl;

cout << "\nReminder: Please update your COVID-19 self-test result for the next 3 days." << endl;

cout << "\n\n!!It is important to take certain precautions to protect yourself from COVID-19." << endl;

cout << "\nHere are some advices to follow:" << endl;

cout << "# Wash your hands frequently with soap and water for at least 20 seconds, or use hand sanitizer if you are unable to wash your hands." << endl;

cout << "# When coughing or sneezing, keep your hands away from your face and cover your mouth and nose with a tissue or your elbow." << endl;

cout << "# Wear a well-fitting mask in public or among others to help prevent the spread of the virus." << endl;

cout << "# Follow all standard operating procedure (SOP) established by your local government or health officials." << endl;

}

else

{

cout << "\n---------------------------------------------------------------------------------------------------------------------";

cout << "\nYou are in category C -- LOW RISK" << endl;

cout << "\n\n!!It is important to take certain precautions to protect yourself from COVID-19." << endl;

cout << "\nHere are some advices to follow:" << endl;

cout << "# Wash your hands frequently with soap and water for at least 20 seconds, or use hand sanitizer if you are unable to wash your hands." << endl;

cout << "# When coughing or sneezing, keep your hands away from your face and cover your mouth and nose with a tissue or your elbow." << endl;

cout << "# Wear a well-fitting mask in public or among others to help prevent the spread of the virus." << endl;

cout << "# Follow all standard operating procedure (SOP) established by your local government or health officials." << endl;

}

}

}

/\*

Ask users to answer Covid-19 questions and receive responses

Re-categorise the users to correct Covid-19 case categories

Pre: users need to login as user before updating their information

\*/

void update(int size)

{

int respd1, respd2, respd3, respd4;

get\_respd(&respd1, &respd2, &respd3, &respd4);

categories(respd1, respd2, respd3, respd4, size);

const char\* classification = user[size].classification;

strcpy\_s(user[size].classification, classification);

}

# 3 SAMPLE OUTPUT

## 3.1 Module 1: Registration

|  |  |
| --- | --- |
| Enter ‘2’ to enter register menu | Shape  Description automatically generated with medium confidence |
| Enter ‘Y’ to proceed to register | Text  Description automatically generated with medium confidence |
| Validate the input before proceed to register. | A screenshot of a computer  Description automatically generated with medium confidence |

### 3.1.1 Personal Information Page

|  |  |
| --- | --- |
| Validation for phone number | Graphical user interface, text  Description automatically generated |
| Validation for full name | Text  Description automatically generated with medium confidence |
| Validation for username | Graphical user interface, text  Description automatically generated |
| Validation for NRIC | Text  Description automatically generated |
| Validation for age | Shape  Description automatically generated with low confidence |
| Validation for gender | Graphical user interface, text  Description automatically generated |
| Validation for race | Text  Description automatically generated |
| Validation for home address | Graphical user interface, text  Description automatically generated |
| Validation for password | A screenshot of a computer  Description automatically generated with medium confidence |
| Sample output when all inputs are correct |  |

### 3.1.2 Covid-19 Status Page

|  |  |
| --- | --- |
| Test case for Category A – Confirmed Case | Graphical user interface, text  Description automatically generated with medium confidence |
| Test case for Category B – Close Contact |  |
| Test case for Category B – Suspected Case (Asymptomatic) |  |
| Test case for Category B – Suspected Case (Symptomatic) |  |
| Test case for Category C – Low Risk  (Asymptomatic) | Text  Description automatically generated |
| Test case for Category C – Low Risk  (Symptomatic) |  |

### 3.1.3 Vaccination Status Page

|  |  |
| --- | --- |
| Test case for taking 3 doses of vaccine |  |
| Test case for taking 2 doses of vaccine and register for third dose | Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated |
| Test case for taking 2 doses of vaccine and do not register for third dose |  |
| Test case for taking 1 dose of vaccine and register for second dose |  |
| Test case for taking 1 dose of vaccine and do not register for second dose |  |
| Test case for have not taken any vaccination before and register for vaccination |  |
| Test case for have not taken any vaccination before and do not register for vaccination |  |
| Validation for dose date format  Day entered should between 1 to 31, month entered should between 1 to 12, year entered must between 2020 to 2023 |  |
| Validation for vaccine type |  |

|  |  |
| --- | --- |
| User registered successfully and the data is saved to “Biodata.txt” text file |  |

## 3.2 Module 2: Login

|  |  |
| --- | --- |
| Enter 1 to enter login menu | Graphical user interface, text  Description automatically generated |
| In login menu, sample output when user enters 0 to exit login menu | Rectangle  Description automatically generated  Text  Description automatically generated with low confidence |
| In main menu, sample output when user enters 3 to quit the entire Covid-19 management system. |  |

### 3.2.1 Login as Admin Page

|  |  |
| --- | --- |
| In login menu, enter 1 to login as admin.  Sample output when the admin’s username entered is not correct |  |
| Sample output when admin’s username is correct, but admin’s password is incorrect |  |
| Sample output when admin’s username and password entered are correct.  Admin options are displayed. |  |
| Sample output when admin chooses option 1,  table and horizontal bar graph are displayed for number of cases in each category. |  |
| Sample output when admin chooses option 2, all users’ profiles are displayed. |  |
| Sample output when the admin enters invalid option. |  |

### 3.2.2 Login as User Page

|  |  |
| --- | --- |
| In login menu, enter 2 to login as user. |  |
| Sample output when the username entered is not registered. |  |
| Sample output when the username entered is correct, but password is wrong and the user resets the password by entering correct reCAPTCHA. |  |
| Sample output when the username entered is correct, but password is wrong.  The user wants to reset the password but user enters wrong reCAPTCHA at the first time and the system generate a new reCAPTCHA for user to enter again. |  |
| Sample output when user enters correct username but wrong password.  User can re-login again with correct username and password. |  |
| Sample output when the user enters correct username and password.  User options are displayed. |  |
| Sample output when the user chooses option 1, user’s covid-19 status is updated. |  |
| Sample output when user chooses option 2, user’s profile is displayed. |  |
| Sample output when user enters invalid option. |  |

# 4 SAMPLE INPUT

## 4.1 Text File: Biodata.txt

|  |
| --- |
| Caitlyn Yong|0167456678|985677-14-7890|23|F|B|C-12-10, No.7 Jalan Citra 56100, Selangor|Caitlyn679|Caitlyn\_64|LOW RISK|01/11/2020|Pfizer|12/03/2021|Sinovac|31/12/2022|Sinovac  Johnson|0185267935|948676-06-6678|32|M|C|T-3-15, No.6 Jalan Lexus 59100, Kedah|John67son9|John11|CLOSE CONTACT|05/08/2021|Pfizer|26/09/2021|Pfizer|17/01/2022|Sinovac  Oliver|0176557890|780411-08-5567|47|F|C|H-15-1, No.4 Jalan Dataran 51000, Sabah|Oli9012\_23|Oli\_london|CONFIRMED CASE|15/07/2021|Pfizer|05/08/2021|Pfizer||  Noah Ling|0173653857|737781-04-1123|22|M|B|N-1-13, No.15 Jalan Desa Park City 52100, Kuala Lumpur|noAh119812|Noah\_77|LOW RISK|13/10/2020|AstraZenca|12/12/2020|Pfizer|19/03/2021|Pfizer  Le Yun|0113985157|030920-01-0123|20|F|C|B-33-10, No.9 Jalan Mandana 53100, Terengganu|minion1124|leyun\_single|LOW RISK|04/06/2021|Sinovac|08/08/2021|Sinovac|07/01/2022|Sinovac  Avania Rania|0154567732|076342-14-0276|19|F|I|N-2-9, No.5 Jalan Gasing 55100, Sarawak|jinglebell|AvaNia4567|SUSPECTED CASE|17/09/2022|Sinovac|26/10/2022|Pfizer||  Mohd Joe|0178905567|089034-05-6678|38|M|I|B-8-1, No.1 Jalan Budiman 57100, Melaka|basketball|Mohd990133|CONFIRMED CASE|15/10/2022|Pfizer|29/11/2022|Pfizer|02/02/2023|Pfizer  Aurora kaur|0175557824|087369-17-8362|78|F|S|G-19-3, No.3 Jalan Jamanan 58000, Johor|Gyhenhehe9|Aurora112|CLOSE CONTACT|03/11/2022|Pfizer|06/12/2022|Pfizer|03/02/2023|Pfizer  Nur Siti|0126777829|836772-93-7362|43|F|I|B-11-12, No.12 Jalan Taman Midah 55100, Kuala Lumpur|Siti003628|Siti\_Balqis|LOW RISK|17/08/2021|AstraZenca|19/09/2021|AstraZenca|18/03/2022|Pfizer  Kumaran|0137362588|973729-67-9432|25|M|H|A-2-26, No.18 Jalan Cochrane 59000, Perlis|kUKAN89008|Kumaran56|CONFIRMED CASE|30/11/2020|AstraZenca|30/12/2020|Pfizer|19/06/2021|Pfizer  Aiden|0138992034|889203-19-0398|69|M|B|M-17-10, No.1 Jalan Bukit Dukung 51100, Pulau Penang|aiden18b20|Aiden\_67|LOW RISK|01/04/2021|Pfizer|06/08/2021|Sinovac||  Priyanka|0179238305|738582-65-9667|36|F|H|C-15-12, No.4 Jalan Permaisuri 54100, Johor|ppopop8890|Pop\_pri45|SUSPECTED CASE|26/04/2021|Sinovac|02/10/2021|Sinovac|05/03/2022|Pfizer  Lucas|0136167030|113904-78-0484|26|M|B|B-3-11, No.8 Jalan Kajang 51000, Pahang|lucathebuc|Lucas1738|LOW RISK|10/12/2022|Pfizer|11/01/2023|Pfizer||  Nur Ciani|0167489204|839409-15-3874|53|F|I|G-10-18, No.19 Jalan Connaught 52100, Kelantan|sillica112|nUR\_ciani1|CONFIRMED CASE|03/09/2021|AstraZenca|02/11/2021|Sinovac|04/04/2022||Sinovac  Ethan|0182737620|928198-23-6643|74|M|C|B-56-4, No.2 Jalan Gemboja 53100, Negeri Sembilan|fastcars86|Eth\_ethan|CONFIRMED CASE|16/08/2021|Sinovac|08/11/2021|Pfizer|08/06/2022|Pfizer  Harish Singh|0128583662|802432-29-3348|34|M|S|C-22-8, No.1 Jalan Senaman 59200, Melaka|harish8819|Harish91|CLOSE CONTACT|01/12/2020|AstraZenca|01/02/2021|Pfizer|07/07/2021|Pfizer  Elijah|0168583920|382953-75-8305|24|M|C|H-1-32, No.19 Jalan Citra 51600, Sarawak|elmo163820|Elijah\_elmo|LOW RISK|30/7/2020|Pfizer|10/09/2020|Pfizer|11/09/2021|Pfizer  Mohd. Ikmal|0138503749|904638-89-4066|37|M|I|C-5-5, No.7 Jalan Kampung Gajah 58100, Selangor|ikamels901|Ikmal\_11|SUSPECTED CASE|03/05/2021|Sinovac|27/06/2021|Sinovac|03/01/2022|Pfizer  Matil Kaur|0173985844|927490-32-3738|83|F|S|D-1-12, No.2 Jalan Sri Petaling 53300, Perlis|MATilhda12|Matil1\_Kaur|LOW RISK|27/04/2021|Sinovac|30/06/2021|Pfizer||  Abigail|0128473477|827848-29-0399|42|F|H|C-65-14, No.5 Jalan Alam Damai 57200, Kuala Lumpur|ABisnel123|Abigail2830|LOW RISK|08/02/2021|Pfizer|18/03/2021|Pfizer||  RUI LUAN|0177220232|030224-04-0384|20|F|C|C324, Cypress Condominium 84000 Muar, Johor|shirley24k|shirley24kkkk|LOW RISK|12/12/2020|Sinovac||||  SALLY SEE|0177220909|090923-09-9990|14|F|I|No. 5, Jalan Ali 13, Taman Abu 84000 Muar, Johor|sallysee|sallysee123|LOW RISK||||||  ZOE CHUA|0107807792|030406-01-0370|20|F|C|C342, Scotpine Condominium 43000 Kajang, Selangor|zoechua|zczy0406|LOW RISK|23/3/2021|Pfizer|23/5/2021|Pfizer|9/9/2022|Sinovac  ALI BIN MOHAMMAD|0177220232|990909-12-2331|24|M|M|No. 6, Jalan Sembilan, Taman Durian 85000 Batu Pahat, Johor.|alimhd|alimhd99|SUSPECTED CASE||||||  ZOEY LIM|0171234833|020302-04-0300|21|F|C|B23, Durian Condominium, 43000 Kajang, Selangor|zoeyyyy|zoeyyylim|LOW RISK|11/12/2021|Pfizer|2/12/2021|Pfizer|23/4/2022|Sinovac  ASHLEY LIM|0192348765|090219-02-4030|14|F|M|G168, Heng Condominium 85000 Batu Pahat, Johor|ashley23lim|ashleyyyyy|LOW RISK|24/3/2022|Pfizer|31/5/2022|Pfizer|18/1/2023|Sinovac  NURUL NATASYA|0192340939|590923-88-3424|64|F|M|No. 34, Jalan Lavender, Taman Bunga Raya, 91000 Tawau, Sabah|natasya|natasya939|CONFIRMED CASE|14/4/2021|Sinovac|26/6/2021|Pfizer||  PRIYA A/P ASOGAN|0143248573|690423-02-0842|54|F|I|D24, Silk Residence 05000 Alor Setar, Kedah|priyaasg|iampriya|CONFIRMED CASE|30/9/2021|AstraZenca|31/12/2021|AstraZenca||  ERIC CHUA|0177330454|780319-23-2231|45|M|BSH|S45, Gawai Residence 88000 Kota Kinabalu, Sabah|ericchua|eric2231|SUSPECTED CASE|18/8/2022|Sinovac||||  KIM BEI ER|0123920393|090203-43-2940|14|F|C|No. 23, Jalan Utara, Taman Utara 40000 Shah Alam, Selangor|beierrrrr|beier393|LOW RISK|9/9/2022|Pfizer||||  NIVIKA|0182189032|660606-66-6666|57|F|I|No. 66, Jalan Enam, Taman Cempedak 46000 Petaling Jaya, Selangor|nivikaaaa|nivika666|SUSPECTED CASE||||||  MUHAMMAD HAFIF|0166333231|430208-64-9989|80|M|BSK|S95, Jasmani Condominium 98000 Miri, Sarawak|mohdhafif|hafif9989|SUSPECTED CASE|||||| |

# Task Distribution

|  |  |
| --- | --- |
| **Name** | **Task** |
| KUIK RUI LUAN 2103790 SE | * Coding part for functions vacStatus, dose\_question, and register\_vac, register\_main, login, chk\_user * Combining all functions together and debugging the system * Documentation for entire program * Flowchart for vacStatus, dose\_question, register\_vac, register\_main, main functions * Edit flowcharts and combine all flowcharts * Structure chart * Complete report |
| HUI LE YUN 2105611 SE | * Coding part for login function * Flowchart for login function * Solution design for login as user. |
| KIM BEI ER 2103893 SE | * Coding part for registration function * Documentation for registration function * Flowchart for registration * Solution design for register of user which include getting personal information, vaccination status and COVID-19 question from user |
| NIVIKA PRASAD A/P KASHI NATH 2104910 SE | * Coding part for main function, combine register\_main function * Flowchart for main function * Structure chart * “Biodata.txt” text file * Solution design for login as admin. |
| ZOE CHUA ZI YI 2103816 SE | * Coding part for function get\_respd, categories and update * Flowchart for chk\_user, get\_respd, categories and update * Edit “Biodata.txt” text file * Solution design for introduction. |