SV20303 OBJECT ORIENTED PROGRAMMING

[1-2023/2024]

ASSIGNMENT 2 – CLASS AND ALL ITS FEATURES

LECTURER :

1. **PROF. ABDULLAH BADE**
2. **MADAM MIEZRA MIDIN**

NAME : SUREKADEVI SHANMUGANATHAN

MATRIC : BS22110175

TABLE OF CONTENT

NO	CONTENT	PAGE
1.	INTRODUCTION	3
2.	CLASS EXPLANATION	4
3.	APPLICATION EXPLANATION	5 -10
	a) FLOW DIAGARAM	5
	b) CLASS DIAGARAM	6
	c) SOURCE ARRANGEMENT	7 - 10
4.	STRENGTH AND UNIQUENESS	11
5.	SAMPLE OUTPUT	12 - 15

INTRODUCTION

The COVID19 Vaccine Management System is a software application designed to facilitate the efficient and organized management of COVID-19 vaccination-related data. Developed using C++, this system aims to streamline the process of administering and tracking COVID-19 vaccinations, catering to both administrative personnel and users seeking vaccination services.

The system comprises two main user roles: Admin and User. Each role is equipped with specific functionalities, offering a user-friendly interface to interact with the system. The system provides features for adding vaccine stock, managing patient data, and overseeing the overall vaccination process.

Upon launching the system, users are presented with a clear and user-friendly main menu. This menu serves as the gateway to the system's functionalities, offering options to log in as an Admin, User, or exit the system. The Admin module incorporates a secure login mechanism requiring a username, password, and captcha verification. This ensures the integrity and confidentiality of administrative functionalities. After successful authentication, the Admin is presented with a dedicated dashboard offering functionalities such as adding vaccine stock, displaying vaccine center information, managing patient data, adding doctor details, and more.

Admins can efficiently manage vaccine stock by adding doses to various vaccination centers. The system supports multiple centers, and administrators can easily monitor and update vaccine availability. The Admin module provides tools for displaying patient data, searching for specific information based on various criteria, and managing overall vaccination statistics. Admins can add new doctor details to the system, search for doctor data, and display comprehensive information about healthcare professionals involved in the vaccination process.

Users are presented with options to create an account or log in if already registered. The registration process captures essential information, including username, name, and password. Once logged in, users can search for vaccination centers, apply for the first and second vaccine doses, and view their vaccination details. The system ensures a seamless and user-friendly experience for individuals seeking vaccination services. To enhance data integrity, the system incorporates validation mechanisms for usernames, passwords, and other critical information, ensuring the accuracy and reliability of the stored data.

The COVID19 Vaccine Management System serves as a vital tool in the ongoing global efforts to combat the COVID-19 pandemic. Its user-friendly interface, robust administrative features, and data management capabilities contribute to the efficient and organized administration of COVID-19 vaccinations, ultimately playing a crucial role in public health initiatives.

CLASS EXPLANATION

The COVID19 Vaccine Management System is encapsulated within the covid_management class, serving as the central entity orchestrating various functionalities. This class embodies a console-based interface and employs procedural programming practices.

The menu() method functions as the system's gateway, presenting users with a main menu to choose between the roles of an administrator or a regular user, or to exit the system. Based on the user's selection, relevant functions are invoked, steering the program flow.

For administrators, the admin() method orchestrates functionalities such as managing vaccine stock, displaying data, and handling doctor information. Authentication is handled by the admin_password() method, incorporating a verification process involving admin credentials and a captcha for security.

Regular users interact with the system through the user() method, which enables functionalities like searching vaccination centers, applying for vaccine doses, and viewing details. User authentication and registration are managed by the user_password() method, providing a mechanism for creating an account or logging in.

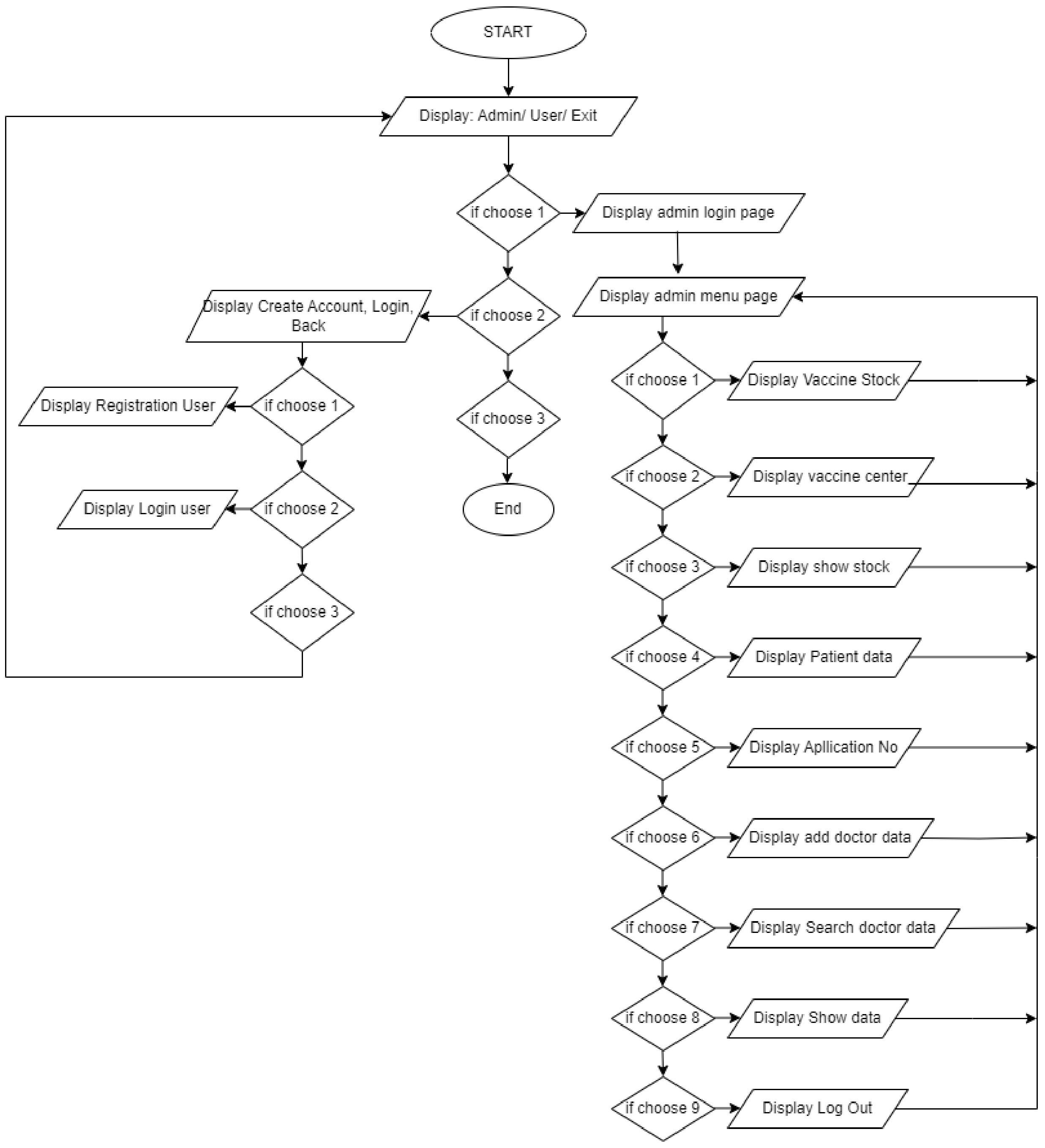
A crucial element is the valid(string str) method, ensuring the uniqueness of usernames during the registration process. This safeguards data integrity by preventing the creation of accounts with duplicate usernames.

One significant administrative function is the add_vaccine_stock() method. This functionality allows administrators to augment vaccine stock in different vaccination centers, specifying the center and quantity of vaccines to be added. The system maintains data related to vaccine stock and patient information, indicating a comprehensive approach to vaccine management.

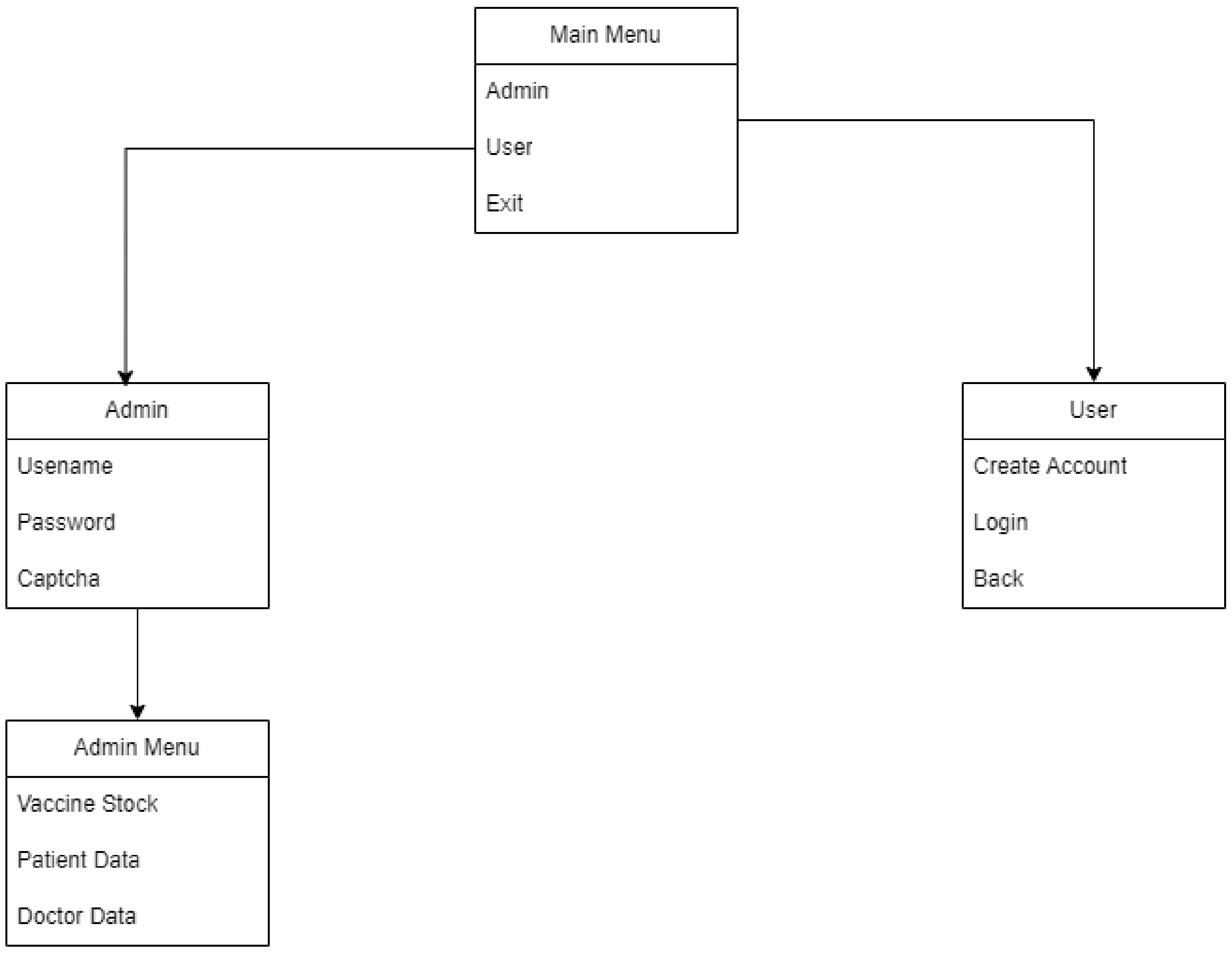
While the current implementation follows procedural programming principles, there is an opportunity for improvement through a more object-oriented design. This would involve creating distinct classes for entities such as administrators, users, and vaccination centers. Such an approach enhances code modularity and readability, fostering a more maintainable and extensible system. Despite its procedural nature, the COVID19 Vaccine Management System demonstrates effective management of vaccine-related data and user interactions within a console-based environment.

APPLICATION EXPLANATION

1. FLOW DIAGRAM



2. CLASS DIAGRAM



3. SOURCE ARRAGEMENTS

1. Header Inclusions:**

```
#include <iostream>

#include <cstring>

#include <windows.h>

#include <fstream>

#include <conio.h>

#include <iomanip>

#include <cstdlib>

#include <string>

#include <unistd.h>

    #define

    TOTAL_VACCINE

    400
```

2. Class :

```
string username;
   string password;
   string usn;
   int tm;
   // FOR VACCINE CENTER
   string center1 = "Pusat Rawatan Warga Universiti Malaysia Sabah";
   string center2 = "Queen Elizabeth Hospital";
   string center3 = "KPJ Sabah Specialist";
   int sum_vaccine_c1 = 0; // Center1 vaccine Dose
   int sum_vaccine_c2 = 0; // Center2 vaccine Dose
   int sum_vaccine_c3 = 0; // Center3 vaccine Dose
   int add, center_no;
```

```
    // For Doctor Details
    string identification_id;
    char specialization[100];
    string center;
    // For User and some Doctor Details
    char name[100];
    char gender[100];
    int age;
    string myKadNumber;
    int c;
    string phone_no, profession, address, vaccine_name;
    int dose;

public:
    void menu();
    void admin();
    void admin_password();
    void user();
    void user_password();
    void valid(string str); // For Valid Username Or not

    // For ADMIN
    void add_vaccine_stock();     // 1
    void display_vaccine_stock(); // 2
    void show_patient_data();     // 4
    void show_data();             // 4-a
    void applied_vaccine();       // 5
    void add_doctor();            // 6
    void search_doctor_data();    // 7
    void display_doctor_data();   // 8
    void doctor_show_data();      // 8-a
    void search_by_myKadNumber();      // 4-a(1)
    void search_by_age();         // 4-a(2)
```

```
            void search_by_profession();  // 4-a(3)
            void search_by_gender();      // 4-a(4)


            // FOR USER
            void search_center();       // 1
            void add_patient_data();    // 2
            void patient_show_data();   // 3
            void update_patient_data(); // 4
             };
```

3. Class : covid_management::menu

```
system("cls");
   int choice;
   cout << "\n\t\t\t\t*****************************************";
   cout << "\n\t\t\t\t   *    COVID19 VACCINE MANAGEMENT SYSTEM    *";
   cout << "\n\t\t\t\t*****************************************";
   // MAIN MENU
   cout << "\n\n\t\t -->> MAIN MENU <<--";
   cout << "\n\n\t\t -->>1. ADMIN";
   cout << "\n\t\t -->>2. USER";
   cout << "\n\t\t -->>3. EXIT";
   cout << "\n\n\t\tEnter Choice: ";
   cin >> choice;
   // CALLING RELEVANT FUNCTION AS PER CHOICE
   switch (choice)
   {
   case 1:
      admin();
      break;
   case 2:
      user();
      break;
   case 3:
      system("cls");
      cout << "\n\n\t\t\t COVID19 VACCINE MANAGEMENT SYSTEM.";
      Sleep(10);
      exit(0);
   default:
      cout << "\n\n\t\t\t Invalid Key... Please Try Again....";
      cout << "\n\n Press Any Key To Continue: ";
      getch();
      menu();
   }
```

```
}
```

4. Class : covid_management::admin

```
admin_password();
A:
   system("cls");
   int admin_choice;
   cout << "\n\t\t\t\t******************************************";
   cout << "\n\t\t\t\t   *    COVID19 VACCINE MANAGEMENT SYSTEM    *";
   cout << "\n\t\t\t\t******************************************";
   // ADMIN MENU OPTIONS
   cout << "\n\n\t\t -->> ADMIN MENU <<--";
   cout << "\n\n\t\t 1. Add Vaccine Stock";
   cout << "\n\t\t 2. Show Vaccine Center";
   cout << "\n\t\t 3. Show Vaccine Stock";
   cout << "\n\t\t 4. Show Patient Data";
   cout << "\n\t\t 5. Show Total Number Of Vaccines Applied";
   cout << "\n\t\t 6. Add New Doctor Data";
   cout << "\n\t\t 7. Search Doctor Data";
   cout << "\n\t\t 8. Show Doctor Data";
   cout << "\n\t\t 9. LOG OUT";
   cout << "\n\n\t\tEnter Choice: ";
   cin >> admin_choice;
   switch (admin_choice)
   {
   case 1:
      add_vaccine_stock();
      goto A;
      break;
   case 2:
      search_center();
      goto A;
      break;
   case 3:
      display_vaccine_stock();
      goto A;
      break;
   case 4:
      show_patient_data();
      goto A;
      break;
   case 5:
      applied_vaccine();
      goto A;
      break;
   case 6:
      add_doctor();
      goto A;
      break;
   case 7:
        search_doctor_data();
        goto A;
        break;
    case 8:
        display_doctor_data();
        goto A;
        break;
    case 9:
        menu();
    default:
        cout << "\n\n\t\t\t Invalid Choice... Please Try Again....";
        cout << "\n\n Press Any Key To Continue: ";
        getch();
        goto A;
        break;
    }
}

// ADMIN LOGIN
void covid_management::admin_password()
{
    system("cls");
    char a_name[20];
    char a_password[20];
    int ch, i = 0, capt = 0, capta = 0;
    cout << "\n\t\t\t\t*****************************************";
    cout << "\n\t\t\t\t   *    COVID19 VACCINE MANAGEMENT SYSTEM    *";
    cout << "\n\t\t\t\t*****************************************";
    cout << "\n\n\t\t -->> LOGIN ADMIN <<--";
    cout << "\n\n\t\tEnter Your Name: ";
    cin >> a_name;
    cout << "\n\t\tEnter Your Password: ";
    while ((ch = getch()) != 13)
    {
        cout << "*";
        a_password[i] = ch;
        i++;
    }
    a_password[i] = '\0';
    srand(time(0));
    capt = rand();
    cout << "\n\n\t\tCaptcha: " << capt;
    cout << "\n\n\t\tEnter Valid Captcha: ";
    cin >> capta;
    if ((strcmp(a_name, "sureka") == 0) && (strcmp(a_password, "12345") == 0) && (capt ==
capta))
    {
        cout << "\n\n\n\t\t\t\t\t| Verfiying ADMIN |\n\t\t\t\t\t";
        for (int a = 1; a < 8; a++)
        {
            Sleep(500);
            cout << "...";
        }
        cout << "\n\nAccess Granted..\n\n";
        system("PAUSE");
        system("cls");
    }
    else
    {
        cout << "\n\n\n\t\t\t\t| Verfiying ADMIN |\n\t\t\t\t";
        for (int a = 1; a < 8; a++)
        {
            Sleep(500);
            cout << "...";
        }
        cout << "\n\nAccess Aborted...\n\n";
        system("PAUSE");
        system("cls");
        menu();
    }
```

5. Main Function :

```
int main()
{
    system("color B");
    covid_management system;
    system.menu();
}
```

6. Program Flow and Logic :

- Admin log in to access data such as vaccine stock, Vaccine center, Patient data, Doctor Data.

- User log in to register new user, create a new account, update vaccine status, choose vaccine and check vaccination data

7. Strength and Uniqueness :

- Colored Text and Background Output in Output Terminal:

  The program employs colored text and background in the output terminal, enhancing the visual appeal and providing a more user-friendly interface.

- Captcha and password during Input (Changed to *****):

  Security is enhanced by concealing the user's password during input. This not only protects sensitive information but also ensures a more secure user experience.

STRENGTH AND UNIQUENESS

This coding has several features that enhance its uniqueness and effectiveness in a COVID19 vaccine management system. Firstly, it includes data structures such as classes and objects, which allows the program to organize and store patient data efficiently. Additionally, the coding uses file handling and binary search algorithms, enabling the program to read, write, and update patient records efficiently.

Another notable aspect of this coding is its exception handling capabilities. The use of try and catch blocks ensures that the program remains operational even in the event of unexpected errors or exceptions. For example, if a user tries to input an invalid phone number, the program can detect and handle this exception without terminating the program.

Furthermore, the coding is implemented in an object-oriented programming language, such as C++, which supports encapsulation and inheritance principles. Encapsulation helps in organizing and managing code by hiding data implementation details. For instance, in the coding, the data members of the patient class are made private, and only the required data members are made public using getter and setter functions.

Moreover, the coding adheres to the single responsibility principle, which is another important principle in object-oriented programming. This principle suggests that a class should have only one responsibility or function, making the code more organized, maintainable, and scalable.

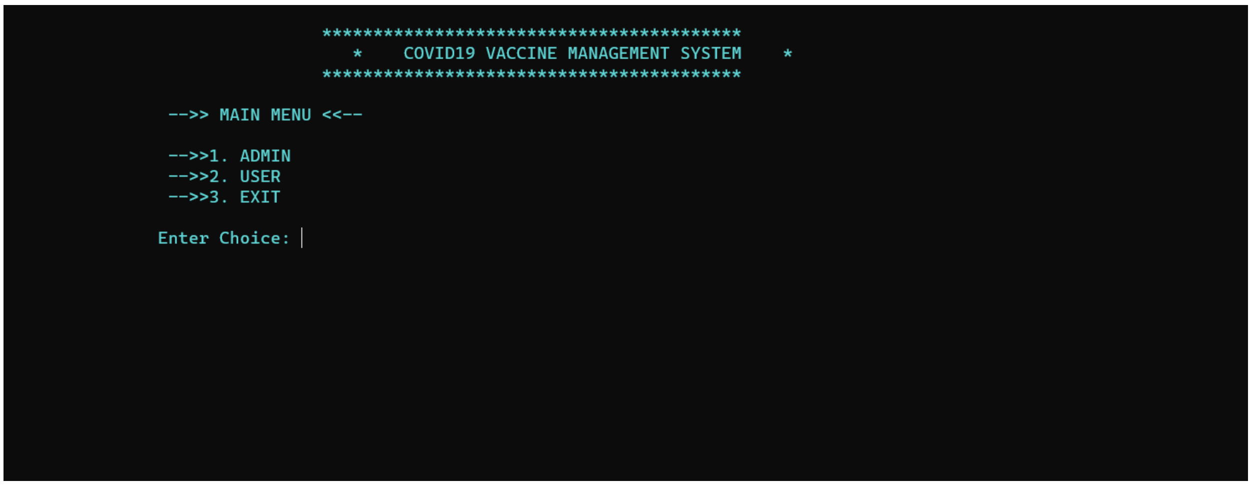
The unique feature of this coding lies in its approach to implementing a menu-driven interface for easy navigation and interaction with the system. This feature ensures that users can easily access different functionalities of the system, such as adding patient details, updating patient data, and searching for a patient record.

The use of search algorithms like binary search further enhances the performance and efficiency of the coding. Binary search algorithms can efficiently search through sorted arrays or lists, reducing the number of iterations required to find a specific element. This approach improves the overall speed and responsiveness of the system.

In conclusion, the uniqueness of this coding lies in its efficient use of data structures, exception handling capabilities, object-oriented programming principles, menu-driven interface, and search algorithms. These features combine to create a robust and scalable COVID19 vaccine management system that effectively handles the needs of healthcare professionals and the general public during this pandemic

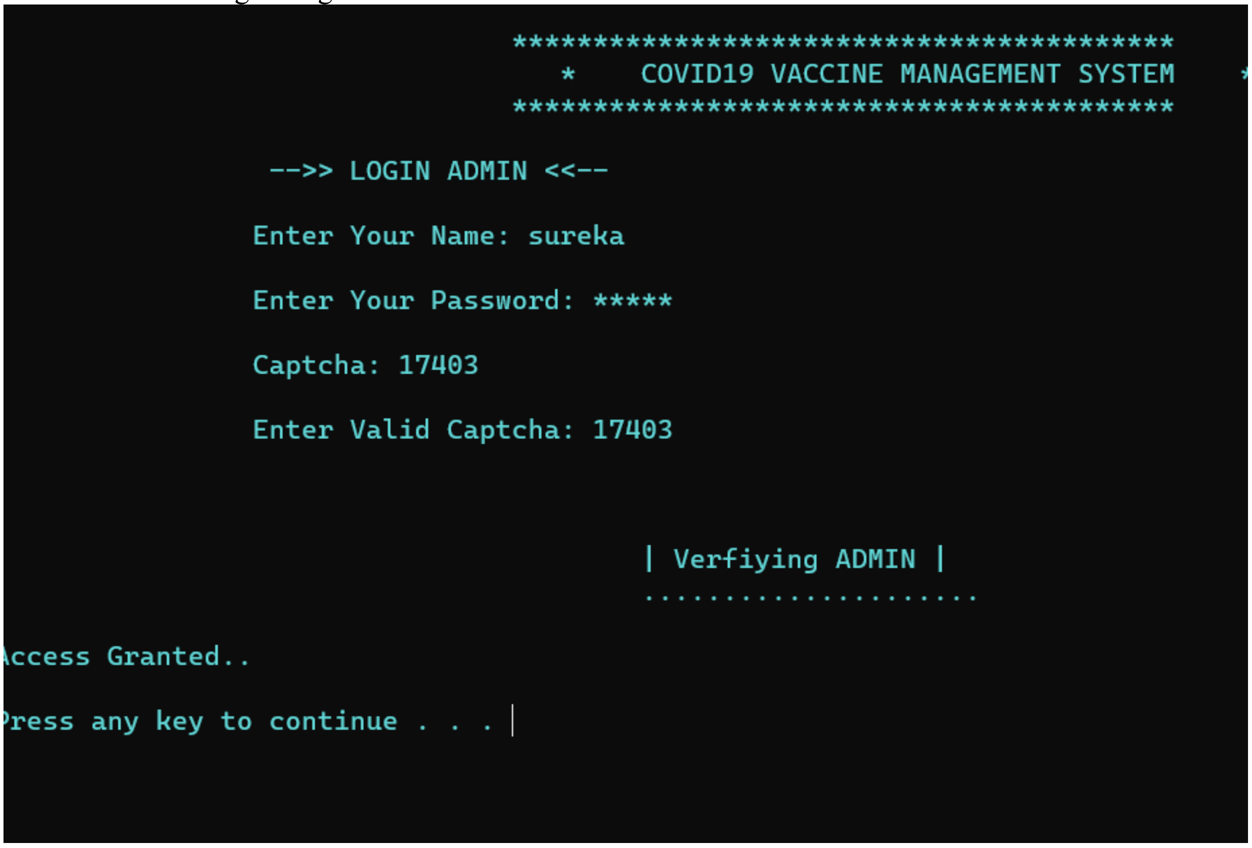
SAMPLE OUTPUT

1. Login/Registration page



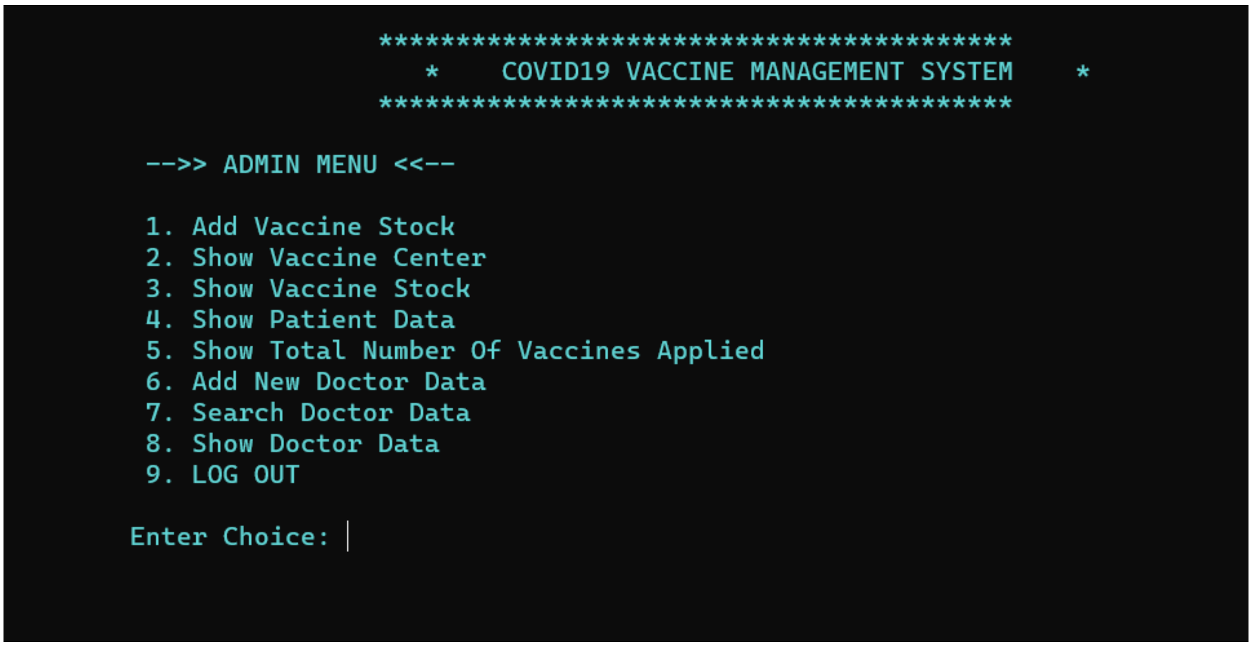
```
                              ******************************************
                                 *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                              ******************************************

               -->> MAIN MENU <<--

               -->>1. ADMIN
               -->>2. USER
               -->>3. EXIT

              Enter Choice: 
```

2. Admin Log In Page



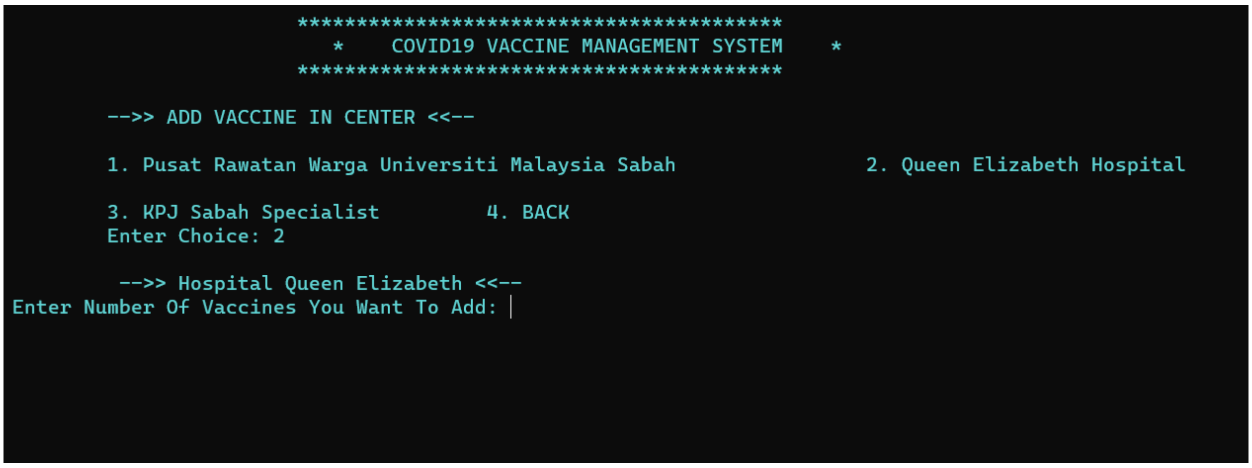
```
                              ****************************************
                                 *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                              ****************************************

                -->> LOGIN ADMIN <<--

               Enter Your Name: sureka

               Enter Your Password: *****

               Captcha: 17403

               Enter Valid Captcha: 17403


                                          | Verfiying ADMIN |
                                          ....................

ccess Granted..

ress any key to continue . . . 
```

3. Admin Menu



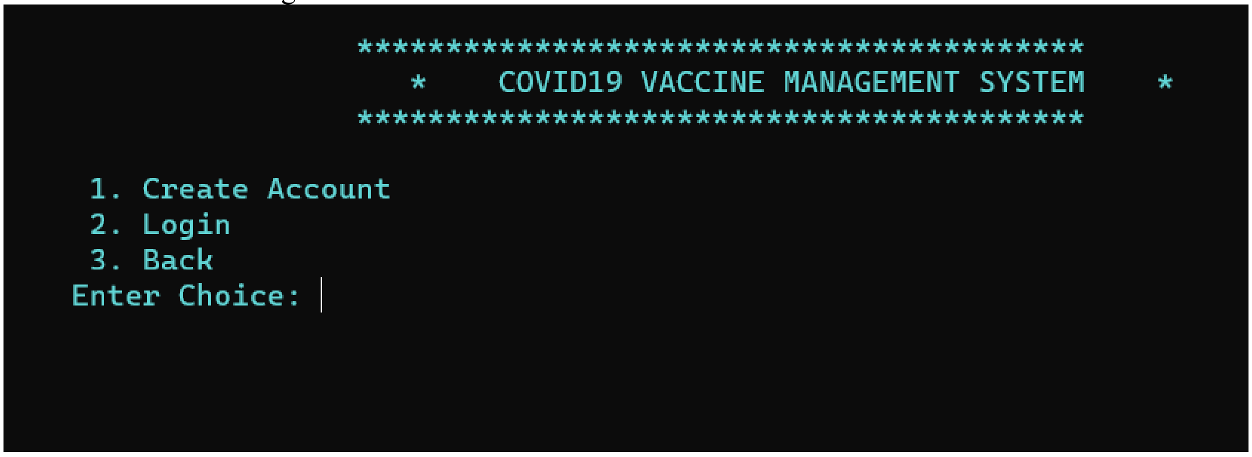
```
                        *****************************************
                           *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                        *****************************************

         -->> ADMIN MENU <<--

         1. Add Vaccine Stock
         2. Show Vaccine Center
         3. Show Vaccine Stock
         4. Show Patient Data
         5. Show Total Number Of Vaccines Applied
         6. Add New Doctor Data
         7. Search Doctor Data
         8. Show Doctor Data
         9. LOG OUT

        Enter Choice: 
```

4. Vaccine Center



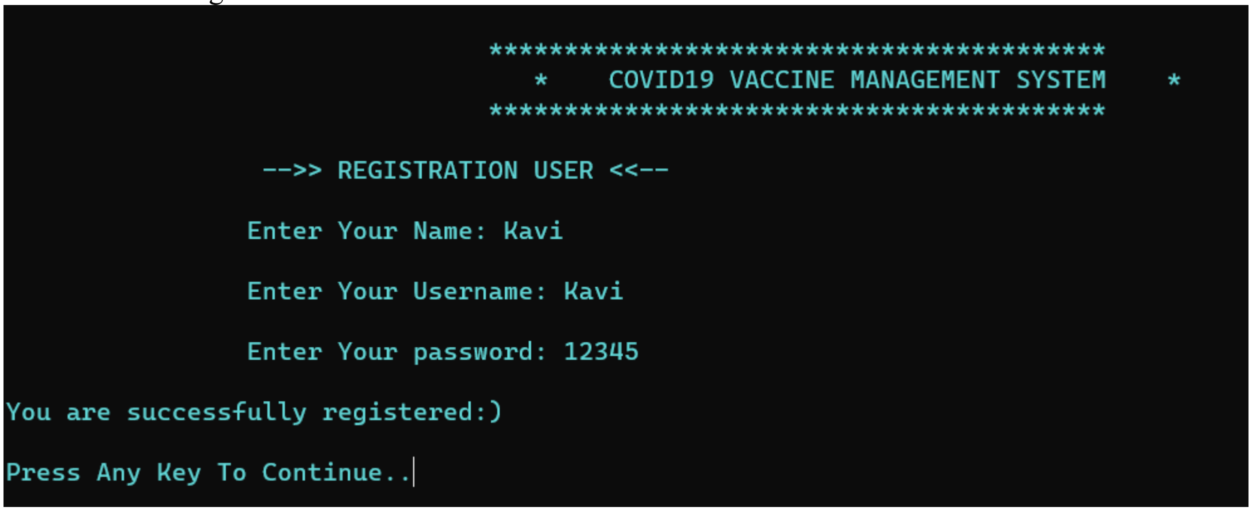
```
                        *****************************************
                           *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                        *****************************************

        -->> ADD VACCINE IN CENTER <<--

        1. Pusat Rawatan Warga Universiti Malaysia Sabah                2. Queen Elizabeth Hospital

        3. KPJ Sabah Specialist         4. BACK
        Enter Choice: 2

         -->> Hospital Queen Elizabeth <<--
Enter Number Of Vaccines You Want To Add: 
```

5. User Menu Page



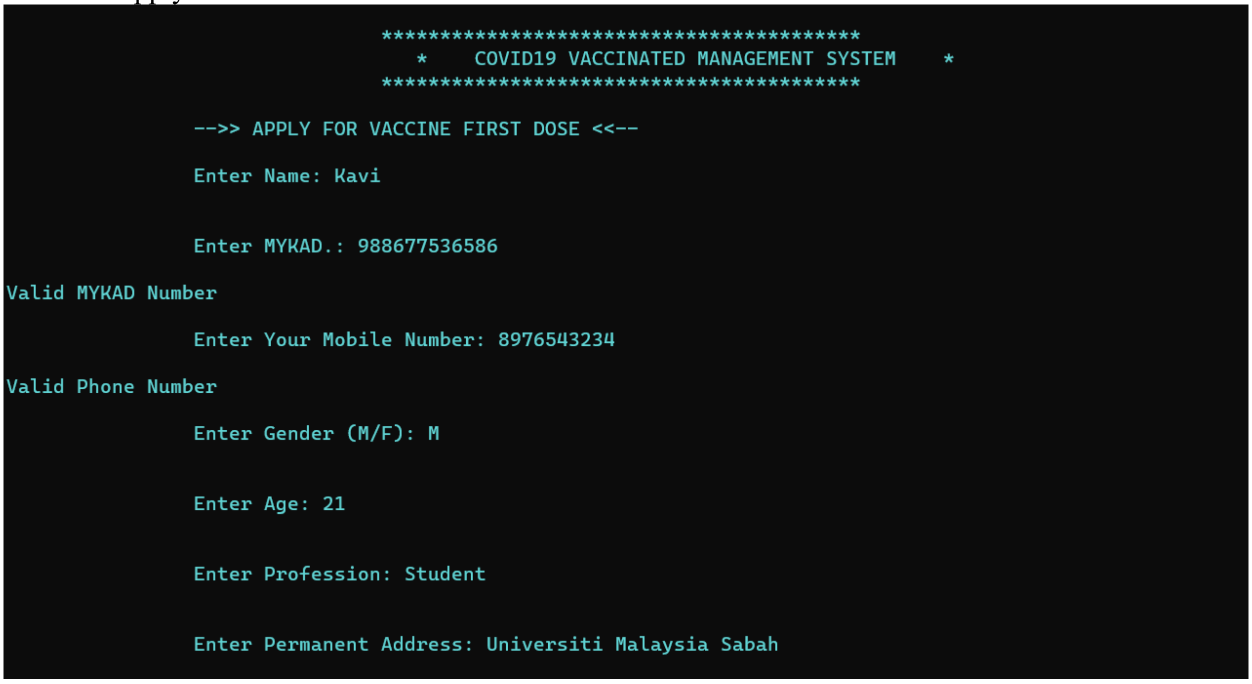
```
                        *****************************************
                           *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                        *****************************************

     1. Create Account
     2. Login
     3. Back
    Enter Choice:
```

6. User Registration



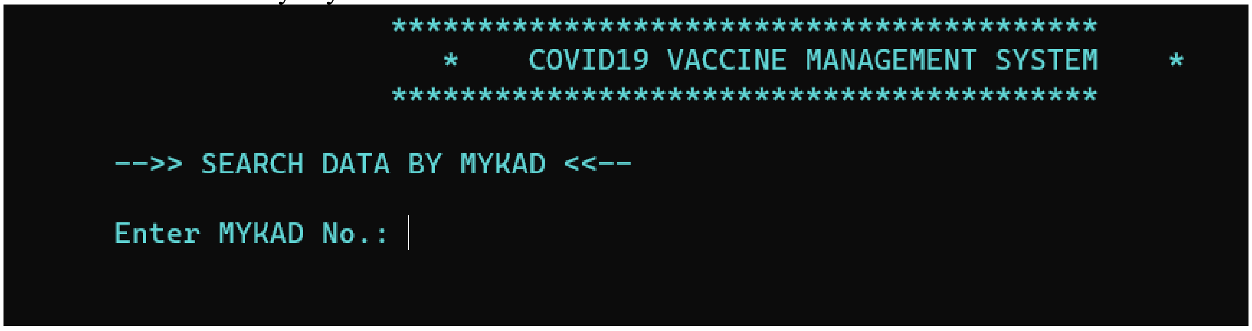
```
                                *****************************************
                                   *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                                *****************************************

                 -->> REGISTRATION USER <<--

                Enter Your Name: Kavi

                Enter Your Username: Kavi

                Enter Your password: 12345

You are successfully registered:)

Press Any Key To Continue..
```

7. Apply for first dose



```
                                ******************************************
                                   *    COVID19 VACCINATED MANAGEMENT SYSTEM    *
                                ******************************************

                --->> APPLY FOR VACCINE FIRST DOSE <<---

                Enter Name: Kavi


                Enter MYKAD.: 988677536586

Valid MYKAD Number

                Enter Your Mobile Number: 8976543234

Valid Phone Number

                Enter Gender (M/F): M


                Enter Age: 21


                Enter Profession: Student


                Enter Permanent Address: Universiti Malaysia Sabah
```

8. Search Data by MyKad



```
                                ******************************************
                                   *    COVID19 VACCINE MANAGEMENT SYSTEM    *
                                ******************************************

        --->> SEARCH DATA BY MYKAD <<---

        Enter MYKAD No.:
```