Pimpri Chinchwad Education Trust's

# Pimpri Chinchwad College of Engineering

**Department of Computer Engineering**

**Project Report**

on

# SWASTHYA

**Your Clinic Management System**

under

**Project-Based Learning – I**

Academic year 2023-24

submitted by

GROUP 12

| Roll numbers | Name of the Student |
|---|---|
| SYCOB80 | VEDANT GAIKAR |
| SYCOB76 | ANUSHKA DONDAL |
| SYCOB88 | KARTIK GIRASE |
| SYCOB86 | PRITHVIRAJ GAVHANE |

submitted under the guidance of

Mrs. Swati Chandurkar

Asst. Prof., Computer Dept.

PCCoE, Pune

# Index

# Problem Statement

## Problem Statement: Design and Implement a Clinic Management System in C++

1. **Objective:**
   a. Develop a Clinic Management System in C++ to facilitate healthcare professionals in efficiently managing patient records. The system should enable the addition of new patients, updating information for existing patients, and displaying patient details. It should also ensure that data is stored and retrieved from files for long-term record-keeping.
   b.
2. **Scope :**
   a. The Clinic Management System aims to provide healthcare professionals with a user-friendly interface to manage patient information, reducing manual paperwork and streamlining record-keeping processes. The system should cover the following aspects:

   **b. Patient Management:**
   i. Adding new patients with essential details.
   c. ii. Updating and modifying existing patient information.
   d. iii. Searching for patients based on specific criteria.
   e. **b. Data Storage:**
   f. I. Ensuring patient data is stored in files for long-term record-keeping and easy retrieval.
   g. ii. Implement secure and organized data storage mechanisms to safeguard patient information.
   h. **c. User Roles:**
   i. I. Designating user roles with appropriate access levels, such as administrative access for healthcare professionals and limited access for support staff.
3. **Functional Requirements:** The Clinic Management System should encompass the following functionalities:

   **a. Patient Registration:**
   1. Capture patient details, including name, age, contact information, medical history, and any relevant notes.
   2. Assign a unique identifier to each patient.

   **b. Patient Information Update:**
   Allow authorized users to modify patient information when needed.

   **c. Patient Search and Retrieval:**
   Provide a search feature that enables users to find patient records based on various parameters, such as name, ID, or medical condition.

   **d. Data Storage and Retrieval:**
   Implement file-based data storage to ensure that patient records are securely saved and can be retrieved when required.

**e. User Authentication:**
Develop a user authentication system with different access levels to protect sensitive patient information.

**f. User Interface:**
Create an intuitive and user-friendly interface that enables healthcare professionals and support staff to navigate and use the system efficiently.

## 4. Non-Functional Requirements:

In addition to the functional requirements, the system should also meet the following non-functional requirements:

**a. Security:**
Ensure data security by encrypting sensitive information and controlling access to authorized users only.

**b. Reliability:**
The system should be robust, minimizing the risk of data loss and ensuring uninterrupted access to patient records.

**c. Performance:**
Optimize system performance for quick data retrieval and smooth operation even with a large patient database.

**d. Scalability:**
Design the system in a way that allows for future expansion and integration with other healthcare systems.

## 5. Constraints:

1. The system should be implemented in C++.
2. Considerations for hardware and software requirements should be made to ensure the system runs efficiently.

## 6. Deliverables:

1. The completed Clinic Management System in C++.
2. Documentation detailing system functionality, usage, and installation instructions.

## 7. Stakeholders:

1. Healthcare professionals (doctors, nurses, etc.).
2. Clinic administrative staff.
3. Patients (indirectly, as the system manages their information).
4. Software developers responsible for implementing and maintaining the system

# Details of object-oriented programming features used

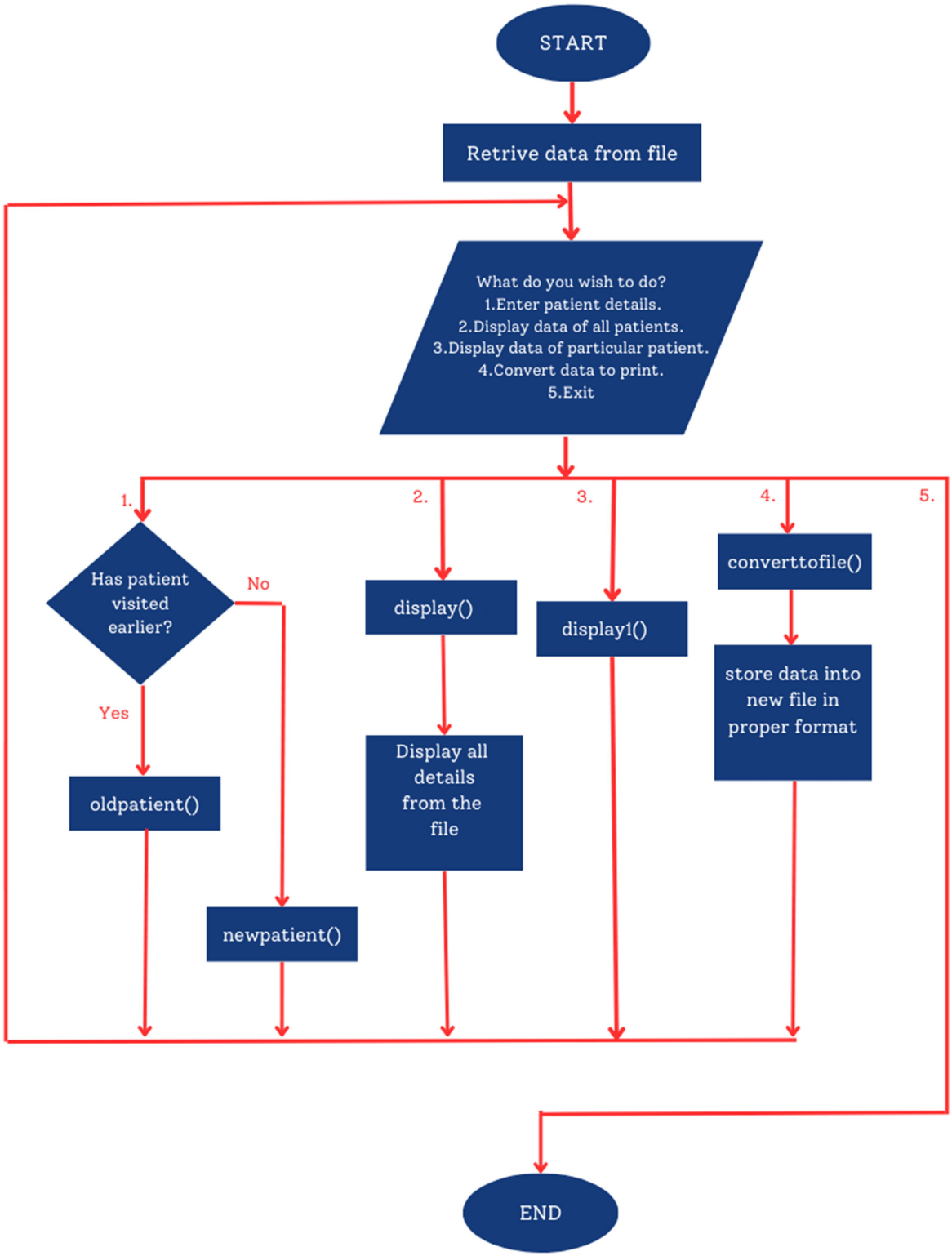
1. **Basic OOP Feature. :** Class as abstract data type.
   Constructor & Destructor.

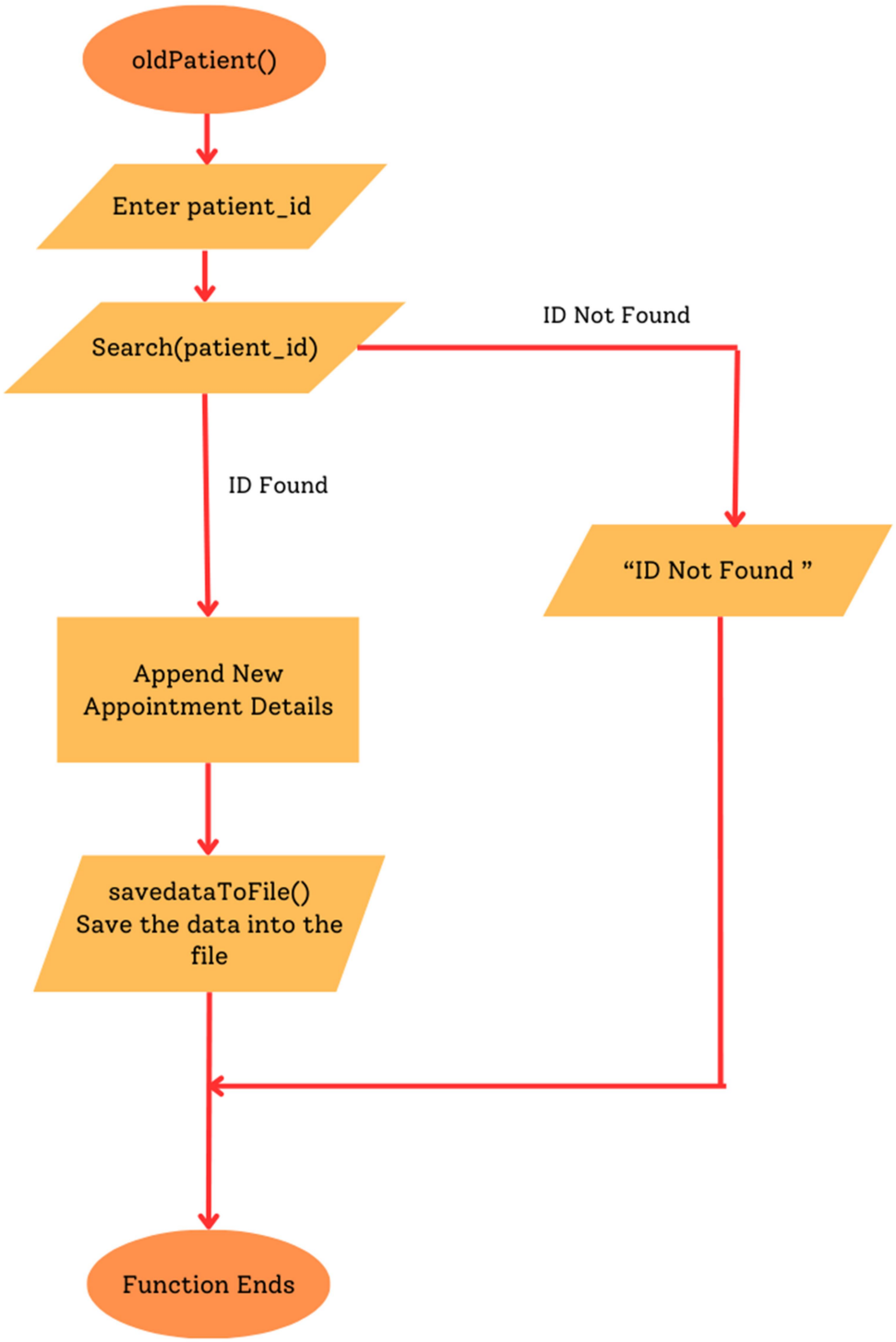
2. **Exception handling:** to handle exceptions met during the program execution

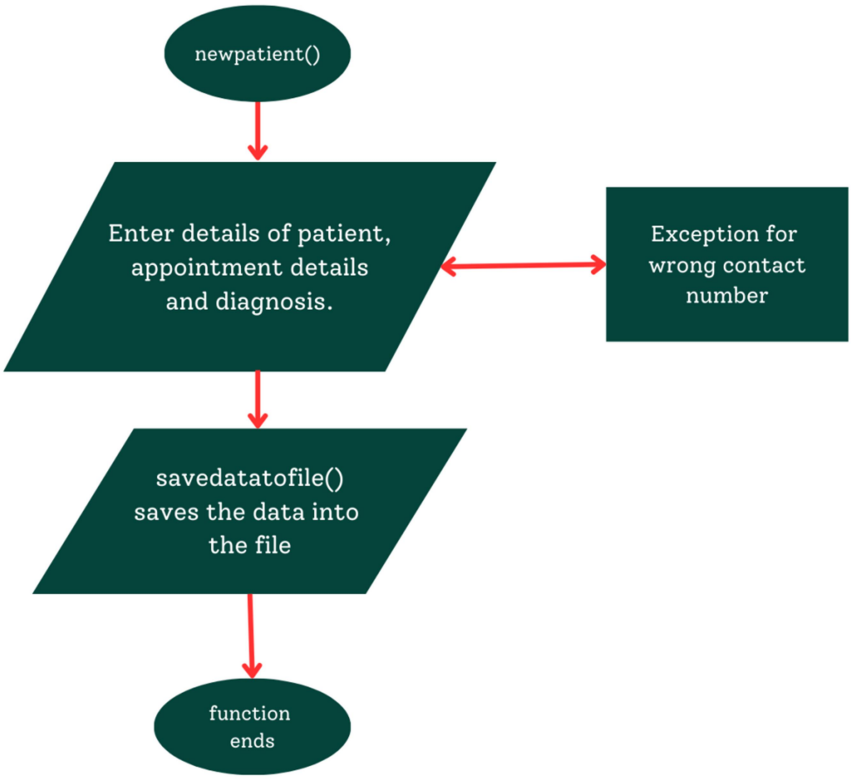
3. **File Handling:** to store the data obtained from the program and retrieve it again for further use.

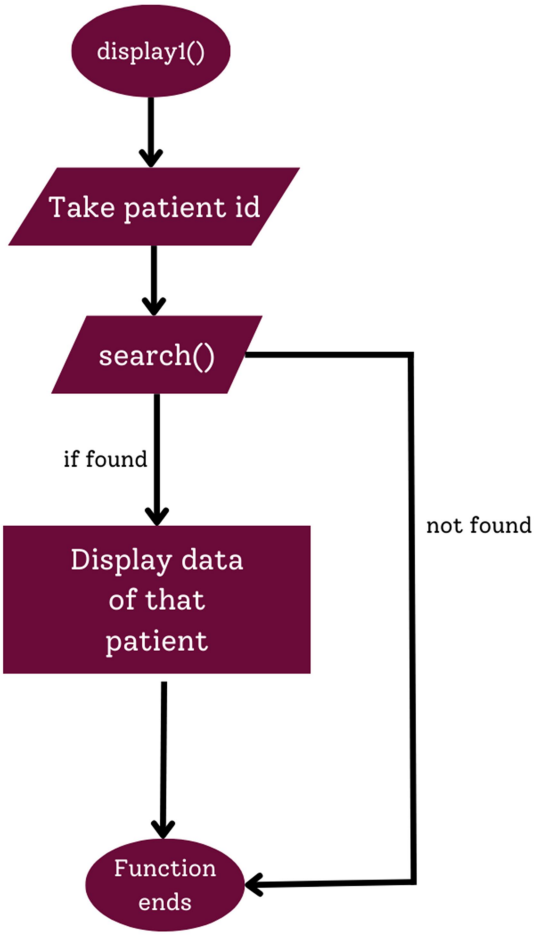
4. **Standard Template Library:** use of string class and vector class.

# Flowchart


START
Retrive data from file
What do you wish to do?
1.Enter patient details.
2.Display data of all patients.
3.Display data of particular patient.
4.Convert data to print.
5.Exit
1.
2.
3.
4.
5.
Has patient visited earlier?
No
Yes
oldpatient()
newpatient()
display()
Display all details from the file
display1()
converttofile()
store data into new file in proper format
END


oldPatient()
Enter patient_id
Search(patient_id)
ID Not Found
ID Found
"ID Not Found "
Append New Appointment Details
savedataToFile() Save the data into the file
Function Ends


newpatient()
Enter details of patient, appointment details and diagnosis.
Exception for wrong contact number
savedatatofile() saves the data into the file
function ends


display1()
Take patient id
search()
if found
not found
Display data of that patient
Function ends

# Pseudo-code of the project

```
# Define the Patient class
class Patient:
    string name
    int patient_id
    int age
    string gender
    List<string> date_appointment
    List<string> time_appointment
    string contact_no
    string address
    List<string> diagnosis

    constructor Patient():
        Initialize all attributes to default values

# Define the Clinic class
class Clinic:
    List<Patient> patients
    string hyphen

    constructor Clinic():
        Initialize patients list and hyphen

    method newPatient():
        Create a new Patient object
        Input patient details (name, ID, age, gender, date of appointment, time of appointment, address, contact no, diagnosis)
        Append the patient to the patients list
        Call saveRecordsToFile() to save the records

    method oldPatient():
        Search for a patient by their ID
        Input updated appointment details (date of appointment, time of appointment, diagnosis)
        Update the patient's appointment details
        Call saveRecordsToFile() to save the records

    method display():
        Check if patients list is empty
        If not, loop through each patient and display their details, including appointment details

    method displayOne():
        Search for a patient by their ID
        Display the details of the selected patient, including appointment details

    method search():
        Input a patient's ID
```

```
        Loop through patients list to find the patient by ID
        Return the index of the found patient or -1 if not found

    method saveRecordsToFile():
        Open a file for writing (e.g., "patient_records.txt")
        Loop through each patient in the patients list
            Write patient details (ID, name, age, gender, address, contact no)
            Loop through each appointment of the patient
                Write appointment details (date of appointment, time of appointment, diagnosis)
        Close the file

    method loadRecordsFromFile():
        Open a file for reading (e.g., "patient_records.txt")
        Clear the patients list
        Initialize a Patient object
        Loop through each line in the file
            If the line indicates the start of a new patient record
                Append the previous patient to the patients list
                Initialize a new Patient object
            Read and set patient data (ID, name, age, gender, address, contact no)
            Loop through each appointment of the patient
                Read and set appointment details (date of appointment, time of appointment, diagnosis)
        Append the last patient to the patients list
        Close the file

# Main function
function main():
    Create an instance of the Clinic class (e.g., "records")
    Call records.loadRecordsFromFile() to load existing records from a file
    Display a welcome message

    Repeat:
        Display a menu of options
        Input the user's choice (ch)
        Switch based on the user's choice:
            Case 1:
                Prompt the user if the patient has visited earlier (ch1)
                If ch1 is 1:
                    Call records.oldPatient()
                Else if ch1 is 2:
                    Call records.newPatient()
                Else:
                    Display an error message
            Case 2:
                Call records.display()
            Case 3:
                Call records.displayOne()
            Case 4:
                Exit the program
```

```
        Default:
            Display an error message

# Start the program by calling the main function
main()
```

# Code of the project

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
using namespace std;

class Patient
{
public:
    string name;
    int patient_id;
    int age;
    string gender;
    vector<string> date_appointment;
    vector<string> time_appointment;
    string contact_no;
    string address;
    vector<string> diagnosis;
    Patient()
    {
        name = "-";
        patient_id = 0;
        age = 0;
        gender = "-";
        contact_no = "-";
        address = "-";
    }

    ~Patient() {}
};

class Clinic
{
    vector<Patient> patients;
    string hyphen = "------------------------------------------------------------
---\n";
    string path = "patientsRecord.txt";

public:
    void newPatient();       // for the patient visiting for the first time
    void oldPatient();       // for the patient visiting again
    void display();          // display data of all patients
    void displayOne();       // displays data of a particular patient
    int search();            // used to find theindex of the patient who has
already visited once and is visitng again.
    void convertToFile();    // to convert the file into printed form
    void saveDataToFile();   // to save the data into the file
    void readDataFromFile(); // to read the data stored into the file
```

```
};

void Clinic::newPatient()
{
    Patient patient;
    cout << hyphen << "Enter the Name of the Patient: \n--> ";
    cin.ignore();
    getline(cin, patient.name);
    cout << "\nEnter the Patient ID of the Patient: \n--> ";
    cin >> patient.patient_id;
    cout << "\nEnter the Age of the Patient: \n--> ";
    cin >> patient.age;
    cout << "\nEnter the Gender of the Patient(M/F): \n--> ";
    cin.ignore();
    getline(cin, patient.gender);
    cout << "\nEnter the Date of Appointment: \n--> ";
    string DoA;
    cin >> DoA;
    patient.date_appointment.push_back(DoA);
    cout << "\nEnter the Time of Appointment (24:00hrs): \n--> ";
    string ToA;
    cin >> ToA;
    patient.time_appointment.push_back(ToA);
    cout << "\nEnter the Address of the Patient: \n--> ";
    cin.ignore();
    getline(cin, patient.address);
    cout << "\nEnter the Contact No. of the Patient: \n--> ";
    getline(cin, patient.contact_no);

    cout << "\nEnter the Diagnosis of the Patient: \n--> ";
    string Diagnosis;
    getline(cin, Diagnosis);
    patient.diagnosis.push_back(Diagnosis);

    patients.push_back(patient);
    saveDataToFile(); // Automatically save records after adding a new patient
}
void Clinic::oldPatient()
{
    int index = search();
    string DoA, ToA, Diagnosis;
    cout << "\nEnter the Date of Appointment: \n--> ";
    cin.ignore();
    getline(cin, DoA);
    patients[index].date_appointment.push_back(DoA);
    cout << "\nEnter the Time of Appointment (24:00hrs): \n--> ";
    getline(cin, ToA);
    patients[index].time_appointment.push_back(ToA);
    cout << "\nEnter the Diagnosis of the Patient: \n--> ";
    cin.ignore();
    getline(cin, Diagnosis);
    patients[index].diagnosis.push_back(Diagnosis);
    saveDataToFile(); // Automatically save records after adding an old patient
    cout << hyphen << "Data successfully saved" << endl;
```

```
}
void Clinic::display()
{
    if (patients.empty())
    {
        cout << "No data to display" << endl;
    }
    else
    {
        cout << hyphen;
        cout << "\t\tDisplaying the data of all patients" << endl;
        for (int i = 0; i < patients.size(); i++)
        {
            cout << hyphen;
            cout << "Sr. No." << i + 1 << endl;
            cout << "- Patient ID          :" << patients[i].patient_id <<
endl;
            cout << "- Name of the Patient :" << patients[i].name << endl;
            cout << "- Age                 :" << patients[i].age << endl;
            cout << "- Gender              :" << patients[i].gender << endl;
            cout << "- Address             :" << patients[i].address << endl;
            cout << "- Contact details     :" << patients[i].contact_no <<
endl;
            cout << "- Appointment details: " << endl;
            for (int j = 0; j < patients[i].diagnosis.size(); j++)
            {
                cout << "\tAppointment " << j + 1 << endl;
                cout << "\tDate of Appointment : " <<
patients[i].date_appointment[j] << endl;
                cout << "\tTime of Appointment : " <<
patients[i].time_appointment[j] << endl;
                cout << "\tDiagnosis : " << patients[i].diagnosis[j] << endl
                     << endl;
            }
        }
    }
}
void Clinic::displayOne()
{
    int index = search();
    cout << "- Patient ID          :" << patients[index].patient_id << endl;
    cout << "- Name of the Patient :" << patients[index].name << endl;
    cout << "- Age                 :" << patients[index].age << endl;
    cout << "- Gender              :" << patients[index].gender << endl;
    cout << "- Address             :" << patients[index].address << endl;
    cout << "- Contact details     :" << patients[index].contact_no << endl;
    cout << "- Appointment details: " << endl;
    for (int j = 0; j < patients[index].diagnosis.size(); j++)
    {
        cout << "\tAppointment " << j + 1 << endl;
        cout << "\tDate of Appointment : " <<
patients[index].date_appointment[j] << endl;
        cout << "\tTime of Appointment : " <<
patients[index].time_appointment[j] << endl;
```

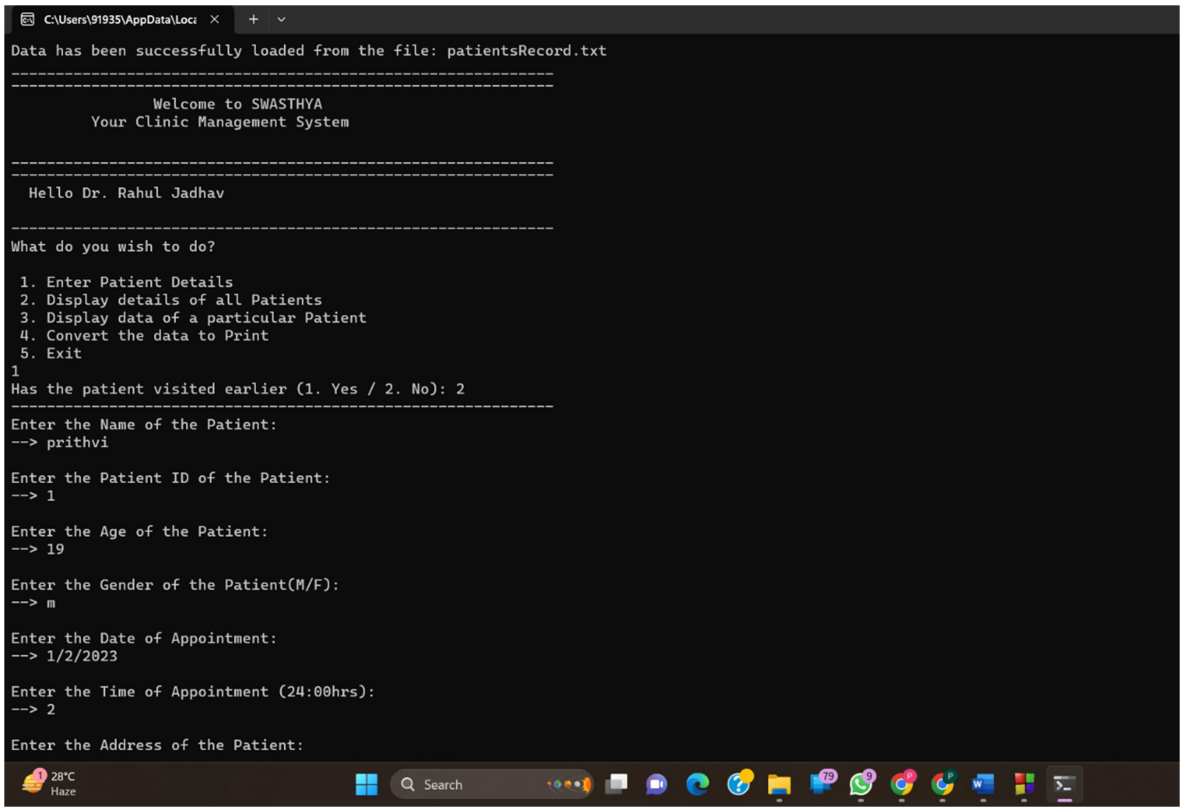
```
        cout << "\tDiagnosis : " << patients[index].diagnosis[j] << endl
             << endl;
    }
}
int Clinic::search()
{
    cout << "Enter the ID of an old patient: ";
    int id;
    cin >> id;
    for (int i = 0; i < patients.size(); i++)
    {
        if (id == patients[i].patient_id)
        {
            return i;
        }
    }
    cout << "Data not found\n";
    return -1;
}
void Clinic::saveDataToFile()
{
    string filename = path;
    ofstream outputFile(filename); // Create an output file stream

    if (outputFile.is_open())
    {
        for (const Patient &patient : patients)
        {
            outputFile << "\n";
            outputFile << patient.patient_id << "\n";
            outputFile << patient.name << "\n";
            outputFile << patient.age << "\n";
            outputFile << patient.gender << "\n";
            outputFile << patient.address << "\n";
            outputFile << patient.contact_no << "\n";

            for (int i = 0; i < patient.diagnosis.size(); i++)
            {
                outputFile << patient.date_appointment[i] << "\n";
                outputFile << patient.time_appointment[i] << "\n";
                outputFile << patient.diagnosis[i] << "\n";
            }

            outputFile << "----------------------------------------";
        }

        outputFile.close();
        cout << hyphen << "Data has been successfully saved to the file: " <<
filename << endl;
    }
    else
    {
        cout << "Error: Unable to open the file for writing." << endl;
    }
```

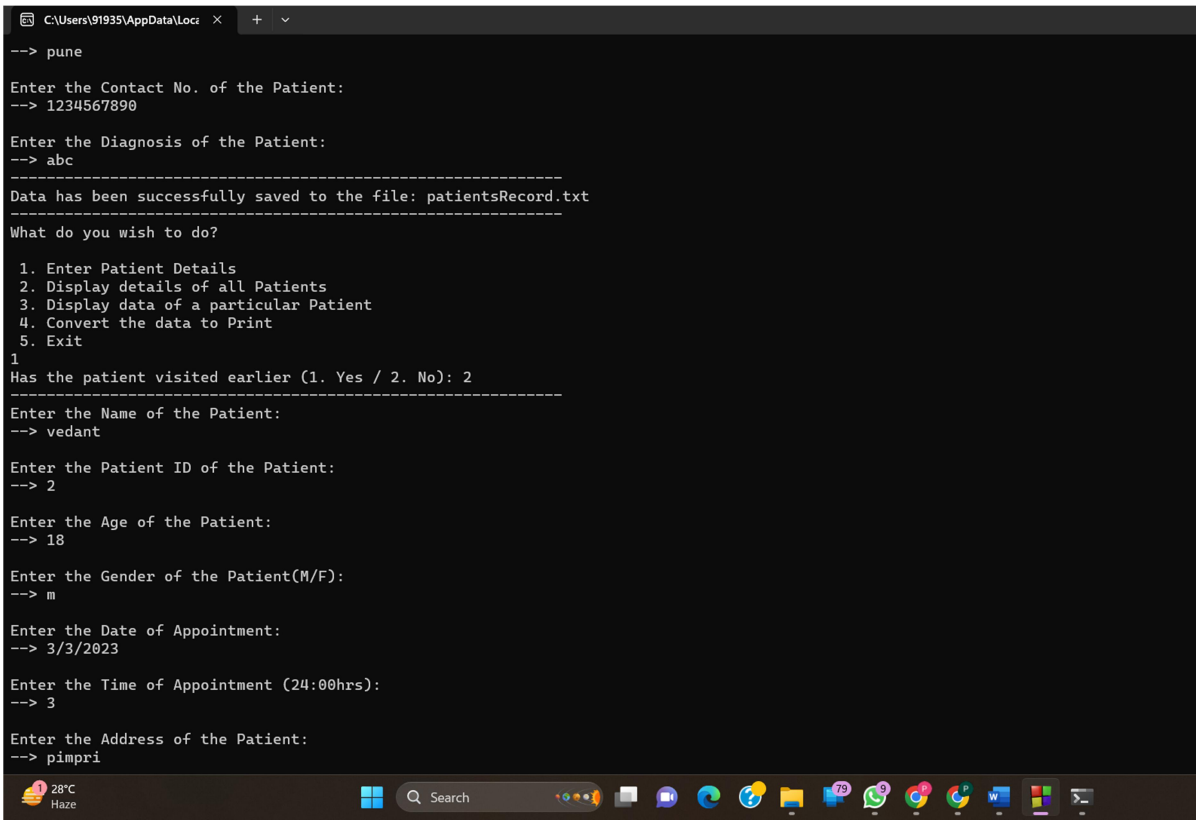
```
}
void Clinic::convertToFile()
{
    string printPath;
    cout << "Enter the path of file to print the data: ";
    cin >> printPath;
    ofstream file(printPath);
    if (file.is_open())
    {
        for (const Patient &patient : patients) // to run untill all members of
vector are traversed
        {
            file << "Patient ID: " << patient.patient_id << "\n";
            file << "Name: " << patient.name << "\n";
            file << "Age: " << patient.age << "\n";
            file << "Gender: " << patient.gender << "\n";
            file << "Address: " << patient.address << "\n";
            file << "Contact No: " << patient.contact_no << "\n";
            for (size_t i = 0; i < patient.date_appointment.size(); i++)
            {
                file << "Appointment " << i + 1 << ":\n";
                file << "Date of Appointment: " << patient.date_appointment[i]
<< "\n";
                file << "Time of Appointment: " << patient.time_appointment[i]
<< "\n";
                file << "Diagnosis: " << patient.diagnosis[i] << "\n";
            }
            file << hyphen;
        }
        file.close();
    }
}
void Clinic::readDataFromFile()
{

    string filename = path;
    ifstream inputFile(filename); // Create an input file stream
    if (inputFile.is_open())
    {
        patients.clear(); // Clear existing patient data in the clinic

        while (inputFile) // here again patients object is created with default
values
        {
            Patient patient;
            inputFile >> patient.patient_id;
            inputFile.ignore(); // Consume the newline character

            getline(inputFile, patient.name);
            inputFile >> patient.age;
            inputFile.ignore(); // Consume the newline character

            getline(inputFile, patient.gender);
            getline(inputFile, patient.address);
```

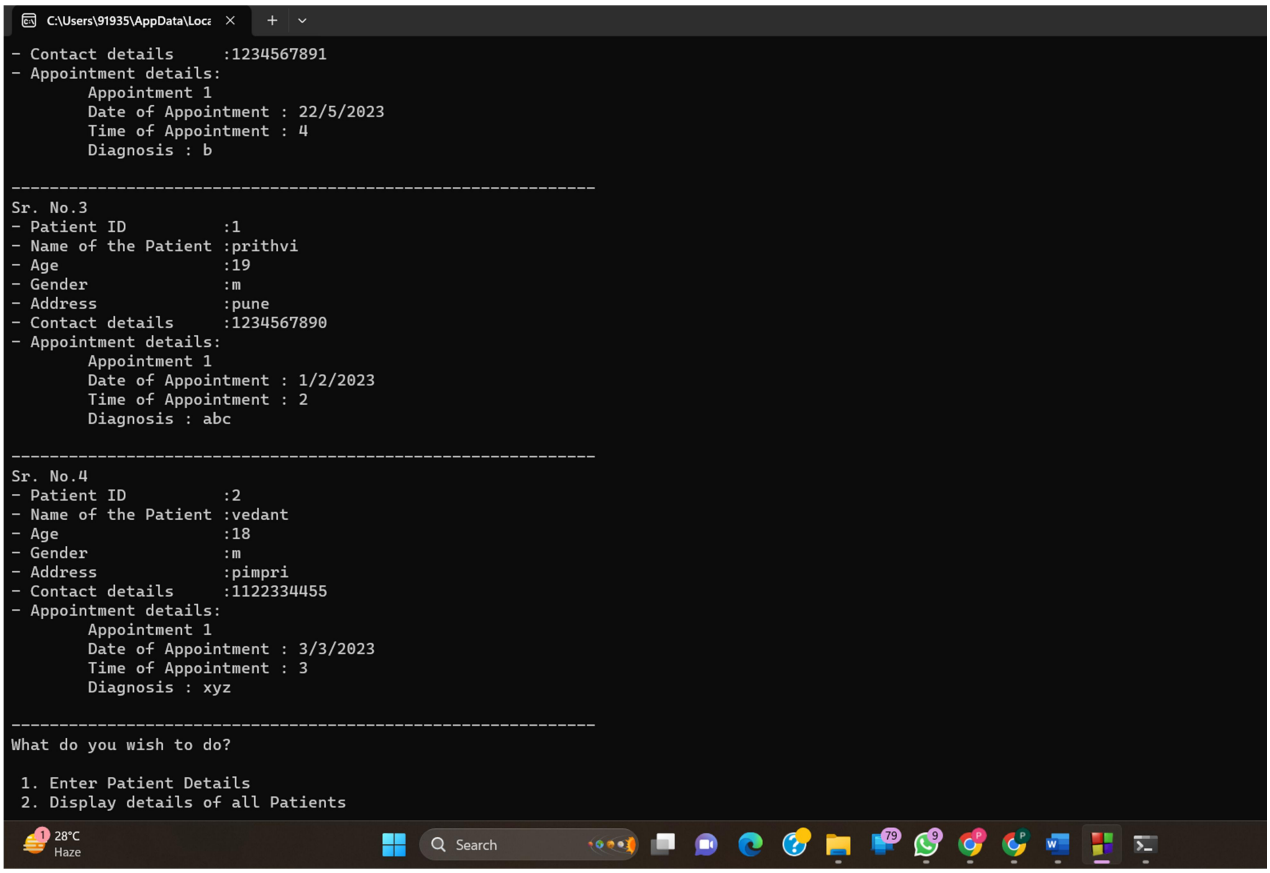
```
            getline(inputFile, patient.contact_no);

            string line;
            while (getline(inputFile, line) && line != "----------------------
----------------")
            {
                patient.date_appointment.push_back(line);
                getline(inputFile, line);
                patient.time_appointment.push_back(line);
                getline(inputFile, line);
                patient.diagnosis.push_back(line);
            }
            patients.push_back(patient); // Add the read patient to the clinic
        }
        inputFile.close();
        patients.pop_back(); // to remove extra patient created in while loop
at l:239
        cout << "Data has been successfully loaded from the file: " << filename
<< endl;
    }
    else
    {
        cout << "Error: Unable to open the file for reading." << endl;
    }
}
int main()
{
    Clinic records;
    records.readDataFromFile(); // Load existing records from a file

    string hyphen = "----------------------------------------------------------
---\n";
    string uc =
"_____________________________________________________________\n";
    cout << uc << hyphen;
    cout << "\t\tWelcome to SWASTHYA\n\t Your Clinic Management System\n " <<
endl;
    cout << uc << hyphen;
    cout << "  Hello Dr. Rahul Jadhav" << endl;
    cout << endl;

    int ch;
    while (1)
    {
        cout << hyphen;
        cout << "What do you wish to do?\n"
             << endl;
        cout << " 1. Enter Patient Details\n 2. Display details of all
Patients\n 3. Display data of a particular Patient\n 4. Convert the data to
Print\n 5. Exit" << endl;
        cin >> ch;
        switch (ch)
        {
        case 1:
```

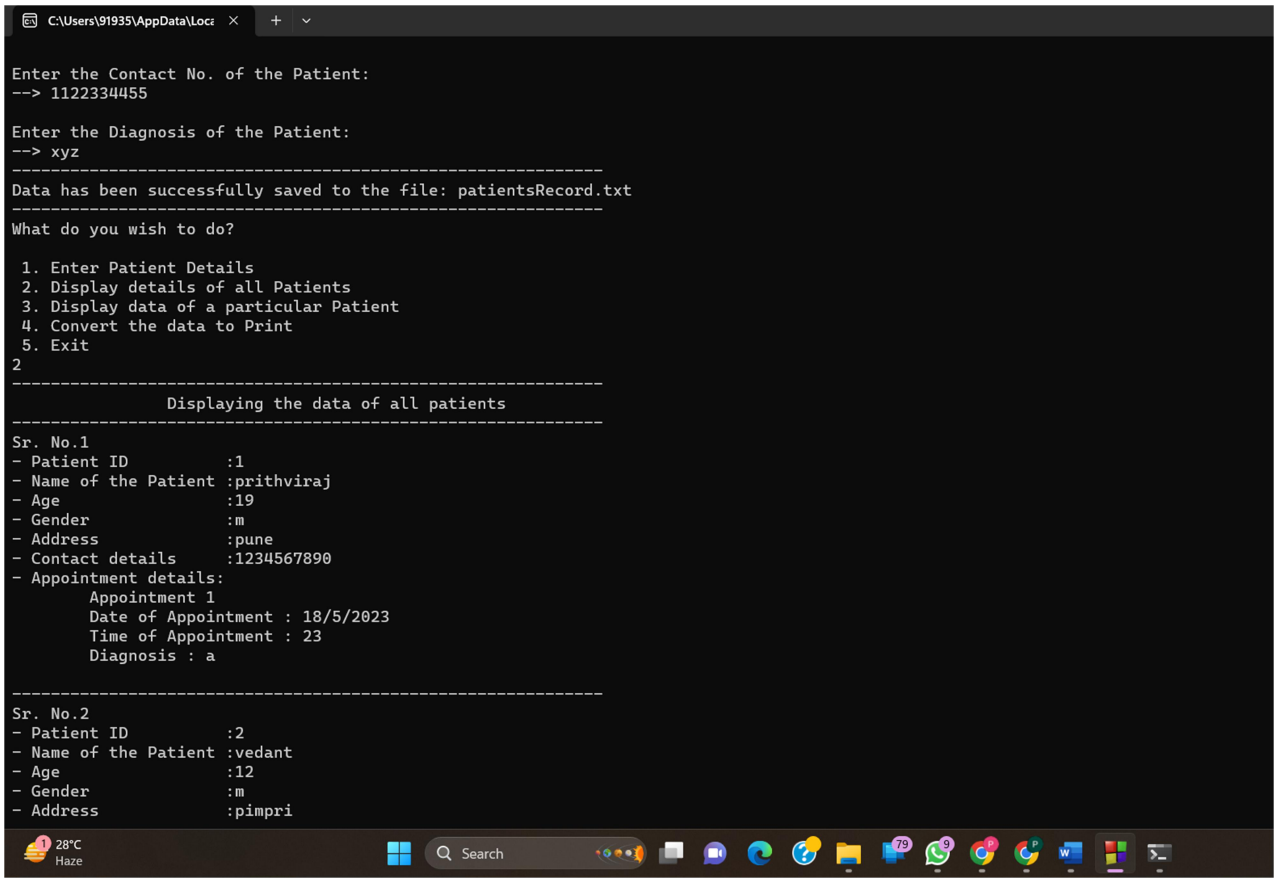
```
        {
            int ch1;
            cout << "Has the patient visited earlier (1. Yes / 2. No): ";
            cin >> ch1;
            if (ch1 == 1)
                records.oldPatient();
            else if (ch1 == 2)
                records.newPatient();
            else
                cout << "Invalid input" << endl;
        }
        break;

        case 2:
            records.display();
            break;

        case 3:
        {
            records.displayOne();
        }
        break;

        case 4:
        {
            records.convertToFile();
            cout << "File is ready to print" << endl;
        }
        break;

        case 5:
            exit(0);

        default:
            cout << "Invalid choice.\n Please try again\n"
                 << endl;
        }
    }
    return 0;
}
```

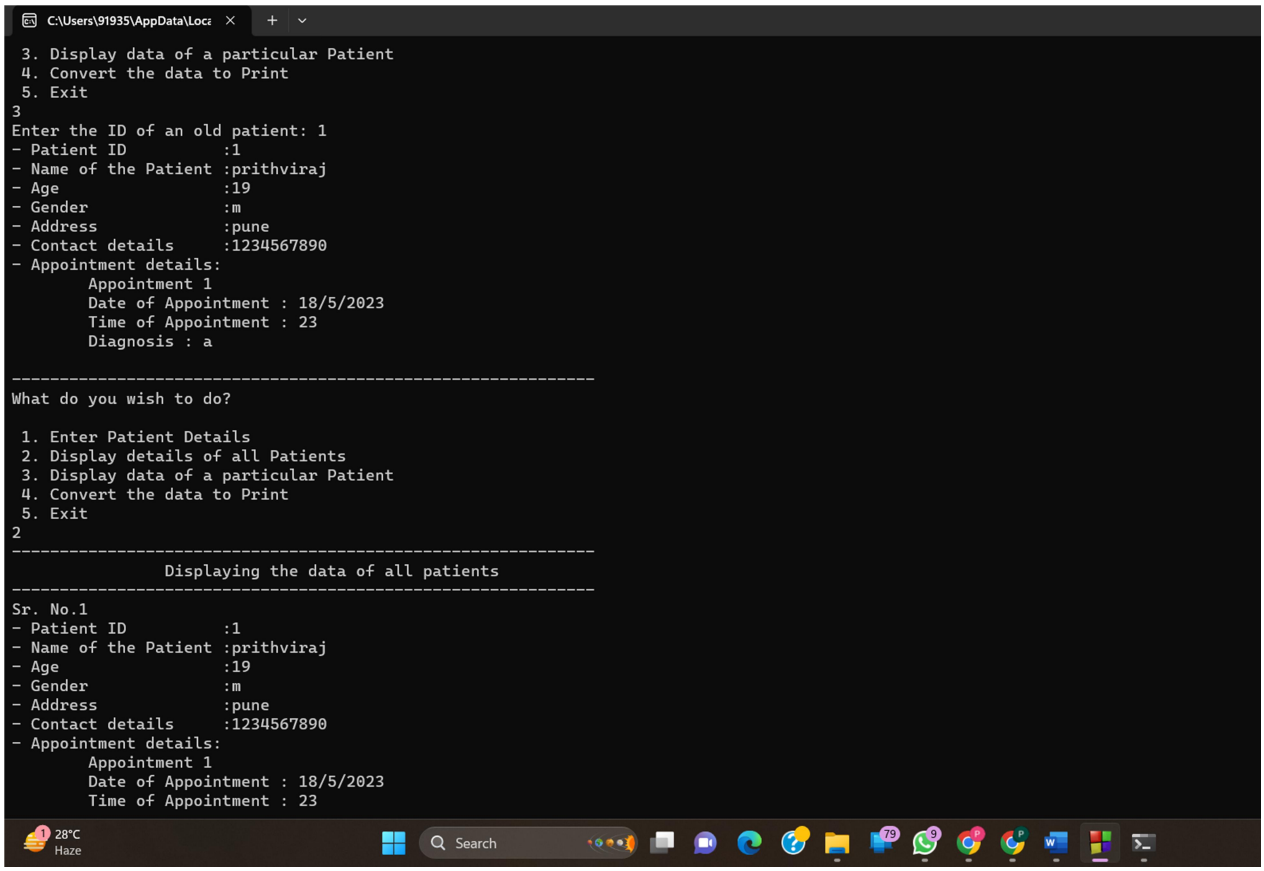
# Output screenshots

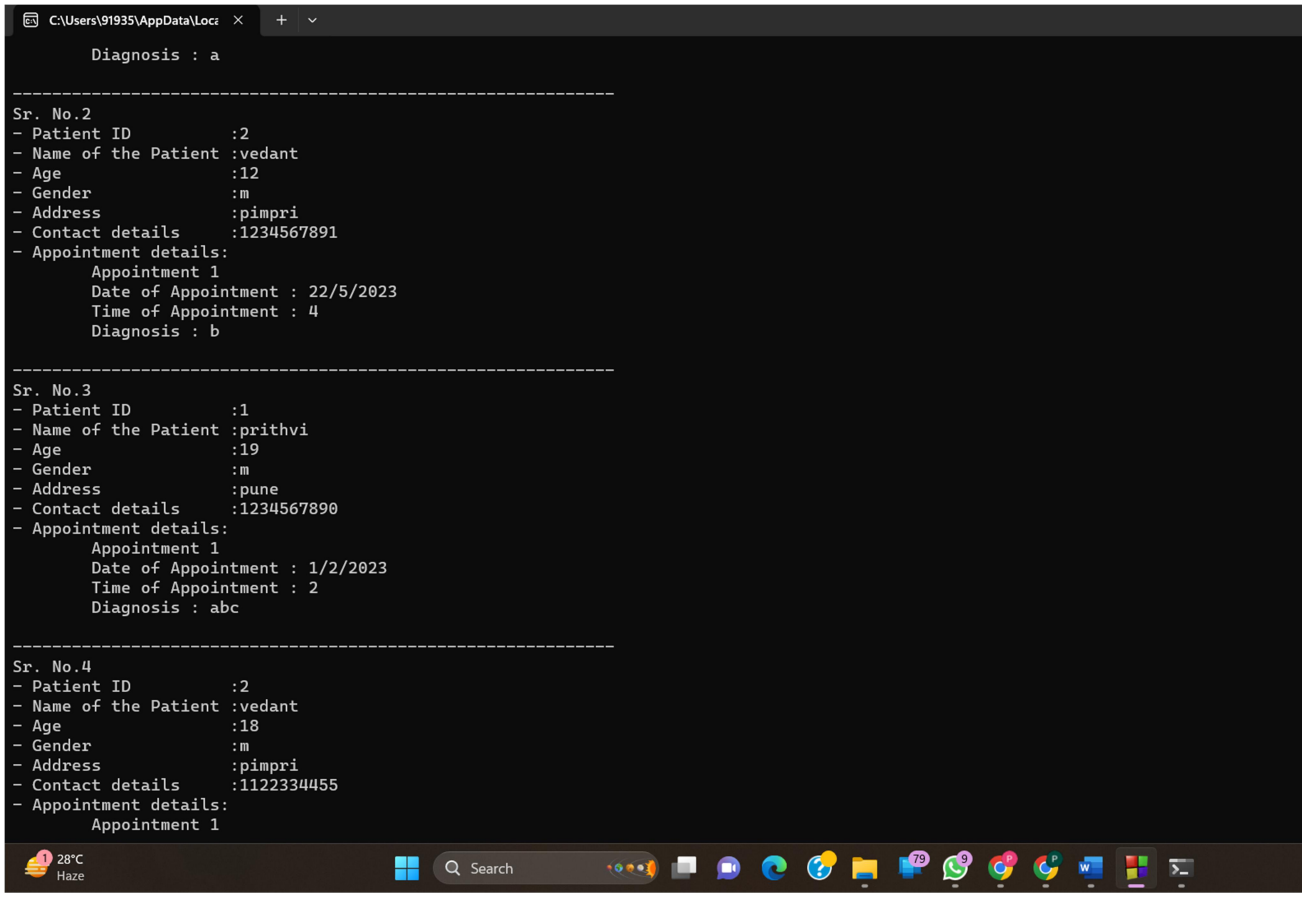


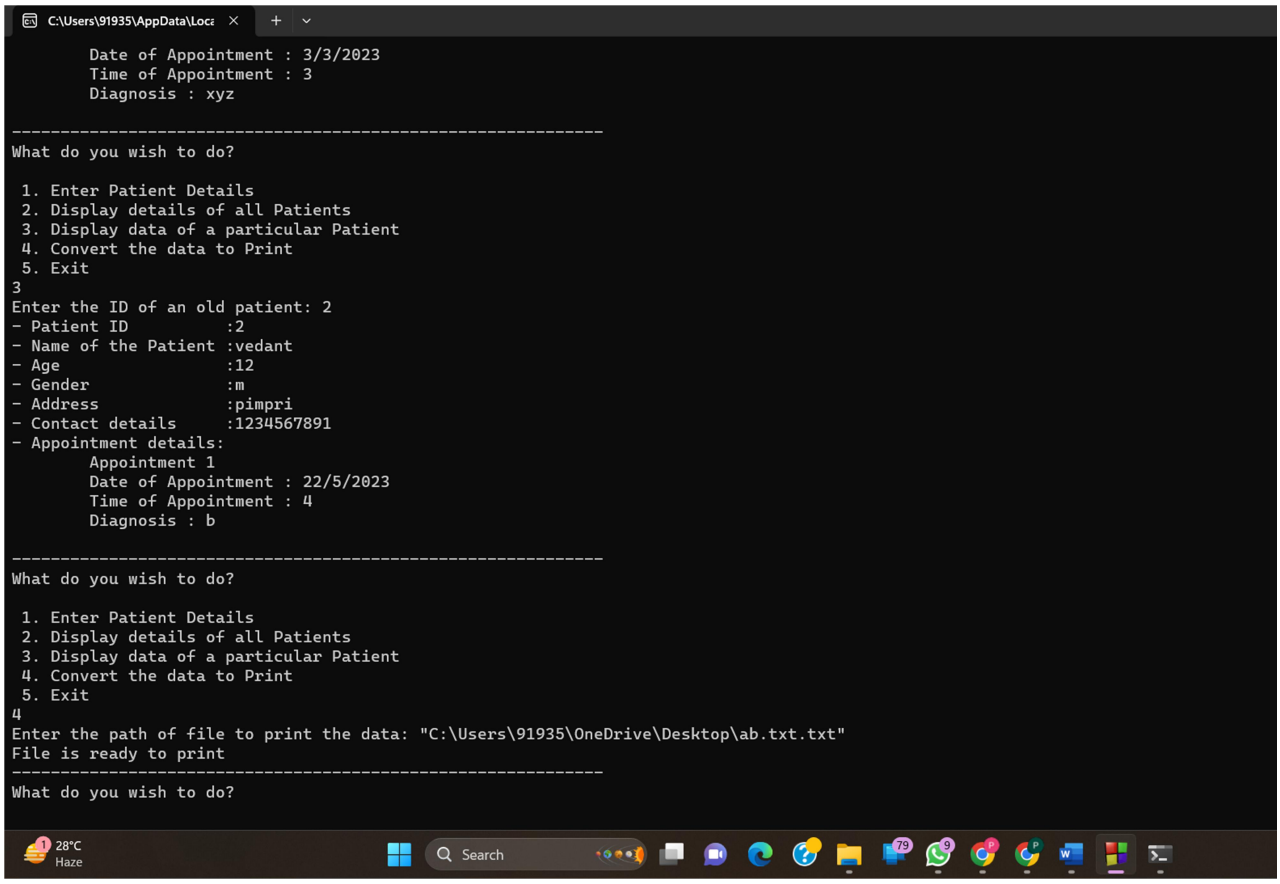



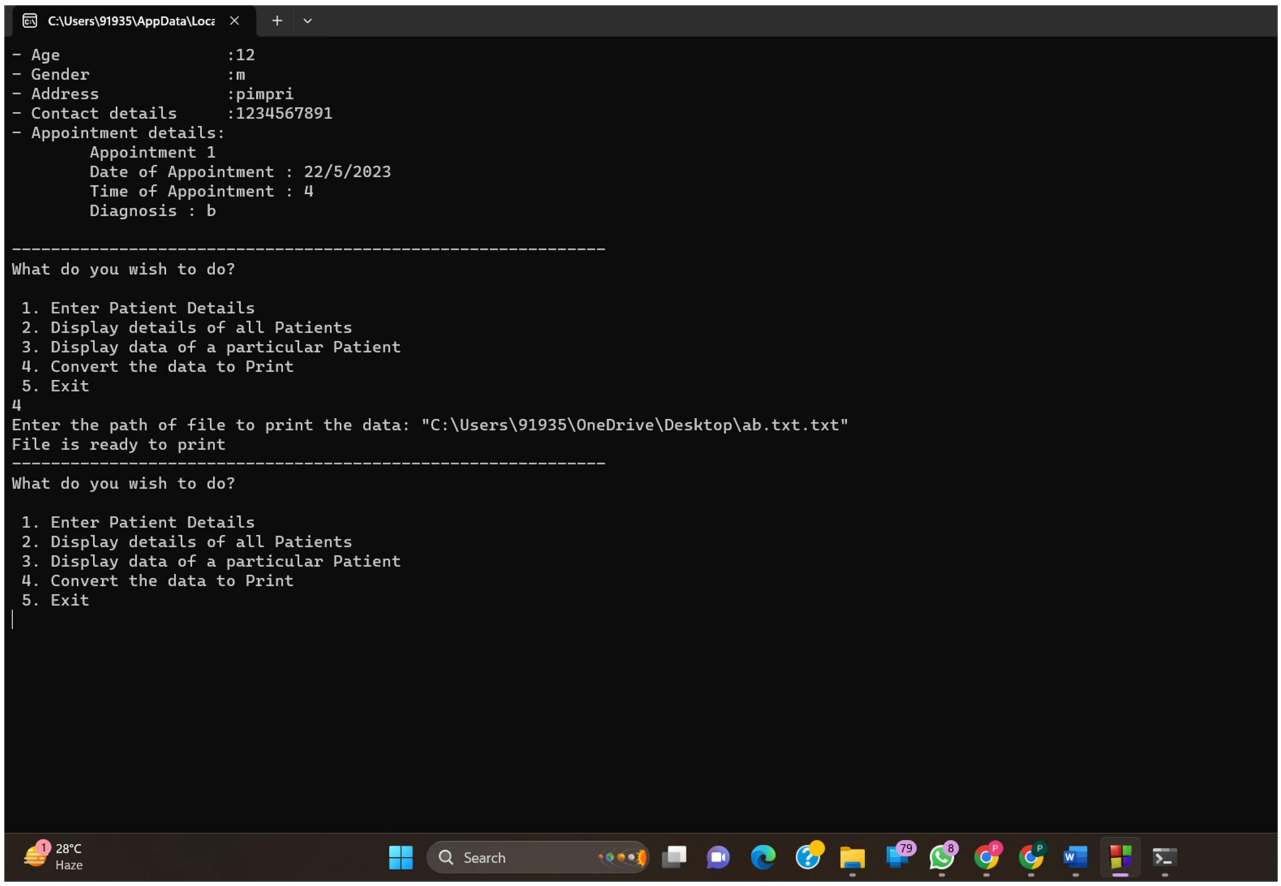

C:\Users\91935\AppData\Loca
- Contact details :1234567891
- Appointment details:
Appointment 1
Date of Appointment : 22/5/2023
Time of Appointment : 4
Diagnosis : b
Sr. No.3
- Patient ID :1
- Name of the Patient :prithvi
- Age :19
- Gender :m
- Address :pune
- Contact details :1234567890
- Appointment details:
Appointment 1
Date of Appointment : 1/2/2023
Time of Appointment : 2
Diagnosis : abc
Sr. No.4
- Patient ID :2
- Name of the Patient :vedant
- Age :18
- Gender :m
- Address :pimpri
- Contact details :1122334455
- Appointment details:
Appointment 1
Date of Appointment : 3/3/2023
Time of Appointment : 3
Diagnosis : xyz
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
28°C
Haze
Search


C:\Users\91935\AppData\Loca
Enter the Contact No. of the Patient:
--> 1122334455
Enter the Diagnosis of the Patient:
--> xyz
Data has been successfully saved to the file: patientsRecord.txt
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
2
Displaying the data of all patients
Sr. No.1
- Patient ID :1
- Name of the Patient :prithviraj
- Age :19
- Gender :m
- Address :pune
- Contact details :1234567890
- Appointment details:
Appointment 1
Date of Appointment : 18/5/2023
Time of Appointment : 23
Diagnosis : a
Sr. No.2
- Patient ID :2
- Name of the Patient :vedant
- Age :12
- Gender :m
- Address :pimpri
28°C
Haze
Search


C:\Users\91935\AppData\Loca
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
3
Enter the ID of an old patient: 1
- Patient ID :1
- Name of the Patient :prithviraj
- Age :19
- Gender :m
- Address :pune
- Contact details :1234567890
- Appointment details:
Appointment 1
Date of Appointment : 18/5/2023
Time of Appointment : 23
Diagnosis : a
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
2
Displaying the data of all patients
Sr. No.1
- Patient ID :1
- Name of the Patient :prithviraj
- Age :19
- Gender :m
- Address :pune
- Contact details :1234567890
- Appointment details:
Appointment 1
Date of Appointment : 18/5/2023
Time of Appointment : 23
28°C
Haze
Search


C:\Users\91935\AppData\Loca
Diagnosis : a
Sr. No.2
- Patient ID :2
- Name of the Patient :vedant
- Age :12
- Gender :m
- Address :pimpri
- Contact details :1234567891
- Appointment details:
Appointment 1
Date of Appointment : 22/5/2023
Time of Appointment : 4
Diagnosis : b
Sr. No.3
- Patient ID :1
- Name of the Patient :prithvi
- Age :19
- Gender :m
- Address :pune
- Contact details :1234567890
- Appointment details:
Appointment 1
Date of Appointment : 1/2/2023
Time of Appointment : 2
Diagnosis : abc
Sr. No.4
- Patient ID :2
- Name of the Patient :vedant
- Age :18
- Gender :m
- Address :pimpri
- Contact details :1122334455
- Appointment details:
Appointment 1
28°C
Haze
Search


C:\Users\91935\AppData\Loca
Date of Appointment : 3/3/2023
Time of Appointment : 3
Diagnosis : xyz
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
3
Enter the ID of an old patient: 2
- Patient ID :2
- Name of the Patient :vedant
- Age :12
- Gender :m
- Address :pimpri
- Contact details :1234567891
- Appointment details:
Appointment 1
Date of Appointment : 22/5/2023
Time of Appointment : 4
Diagnosis : b
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
4
Enter the path of file to print the data: "C:\Users\91935\OneDrive\Desktop\ab.txt.txt"
File is ready to print
What do you wish to do?
28°C
Haze
Search


C:\Users\91935\AppData\Loca
- Age :12
- Gender :m
- Address :pimpri
- Contact details :1234567891
- Appointment details:
Appointment 1
Date of Appointment : 22/5/2023
Time of Appointment : 4
Diagnosis : b
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
4
Enter the path of file to print the data: "C:\Users\91935\OneDrive\Desktop\ab.txt.txt"
File is ready to print
What do you wish to do?
1. Enter Patient Details
2. Display details of all Patients
3. Display data of a particular Patient
4. Convert the data to Print
5. Exit
28°C
Haze
Search

# References (Books/ Notes/ Web Links )

**Books:**

1. E Balaguruswamy

**Notes:**

1. Class notes.

**Web links:**

1. https://www.javatpoint.com/cpp-oops-concepts
2. https://cplusplus.com/reference/vector/vector/