

**TASK:02**

**TITLE: "CONTACT MANAGEMENT**

**SYSTEM IN C++"**

**AUTHOR: SIDRA YOUNAS**

**DATE: 16-08-2024**

**COMPANY: DIGITAL EMPOWERMENT**

**PAKISTAN**

## Table of contents

Introduction.....	03
Objective.....	03
Requirements.....	03
Implementation.....	03
Testing and Results.....	06
Conclusion.....	08

## **Introduction**

The Contact Management System is a simple application developed in C++ as part of an internship project. Its primary function is to manage contacts by allowing users to add, view, and delete entries, each consisting of a name and phone number. The system features a menu-driven interface, making it user-friendly and efficient for basic contact management tasks. This project provided an opportunity to apply fundamental programming concepts, including object-oriented design and dynamic data structures, to solve a practical problem.

## **Objective**

*"The objective of this project is to develop a simple Contact Management System using C++ that allows users to add, view, and delete contacts. The system should be easy to use, with a menu-driven interface."*

## **Requirements**

- Add a contact with a name and phone number
- View all stored contacts
- Delete a contact by name
- Provide a menu-driven interface for the user

## **Implementation**

### ***Code Explanation:***

#### **Contact Class:**

The Contact class is designed to hold information about each contact, specifically the name and phone number.

The class provides methods to set and retrieve the contact's name and phone number.

It also includes a display method to print the contact's details to the console.

#### **Storage of Contacts:**

A vector is used to store multiple Contact objects. The vector dynamically adjusts its size as contacts are added or removed.

Functions for Managing Contacts:

#### **Adding a Contact:**

The addContact function prompts the user to enter a name and phone number. A new Contact object is created and added to the contacts vector.

## Viewing Contacts:

The viewContacts function iterates over the contacts vector and displays each contact's details using the display method.

## Deleting a Contact:

The deleteContact function allows the user to delete a contact by name. It searches for the contact and removes it from the vector if found.

## Menu-Driven Interface:

The main function provides a menu-driven interface that allows users to choose between adding a contact, viewing all contacts, deleting a contact, or exiting the program. The user inputs a choice, and the corresponding function is called.

### Code:

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

class Contact {
private:
    string name;
    string phoneNumber;

public:
    Contact(string n, string p) : name(n), phoneNumber(p) {}

    string getName() const {
        return name;
    }

    string getPhoneNumber() const {
        return phoneNumber;
    }

    void setName(const string& n) {
        name = n;
    }

    void setPhoneNumber(const string& p) {
        phoneNumber = p;
    }

    void display() const {
        cout << "Name: " << name << ", Phone Number: " << phoneNumber << endl;
    }
};

vector<Contact> contacts;
void addContact(vector<Contact>& contacts) {
    string name, phoneNumber;
    cout << "Enter contact name: ";
    cin.ignore(); // to ignore leftover newline character from previous input
```

```

getline(cin, name);
cout << "Enter phone number: ";
getline(cin, phoneNumber);
contacts.push_back(Contact(name, phoneNumber));
cout << "Contact added successfully." << endl;
}

void viewContacts(const vector<Contact>& contacts) {
    if (contacts.empty()) {
        cout << "No contacts available." << endl;
        return;
    }
    for (const auto& contact : contacts) {
        contact.display();
    }
}

void deleteContact(vector<Contact>& contacts) {
    string name;
    cout << "Enter contact name to delete: ";
    cin.ignore();
    getline(cin, name);

    auto it = remove_if(contacts.begin(), contacts.end(), [&](const Contact& contact) {
        return contact.getName() == name;
    });

    if (it != contacts.end()) {
        contacts.erase(it, contacts.end());
        cout << "Contact deleted successfully." << endl;
    }
    else {
        cout << "Contact not found." << endl;
    }
}

int main() {
    vector<Contact> contacts;
    int choice;

    do {
        cout << "\n--- Contact Management System ---\n";
        cout << "1. Add Contact\n";
        cout << "2. View Contacts\n";
        cout << "3. Delete Contact\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

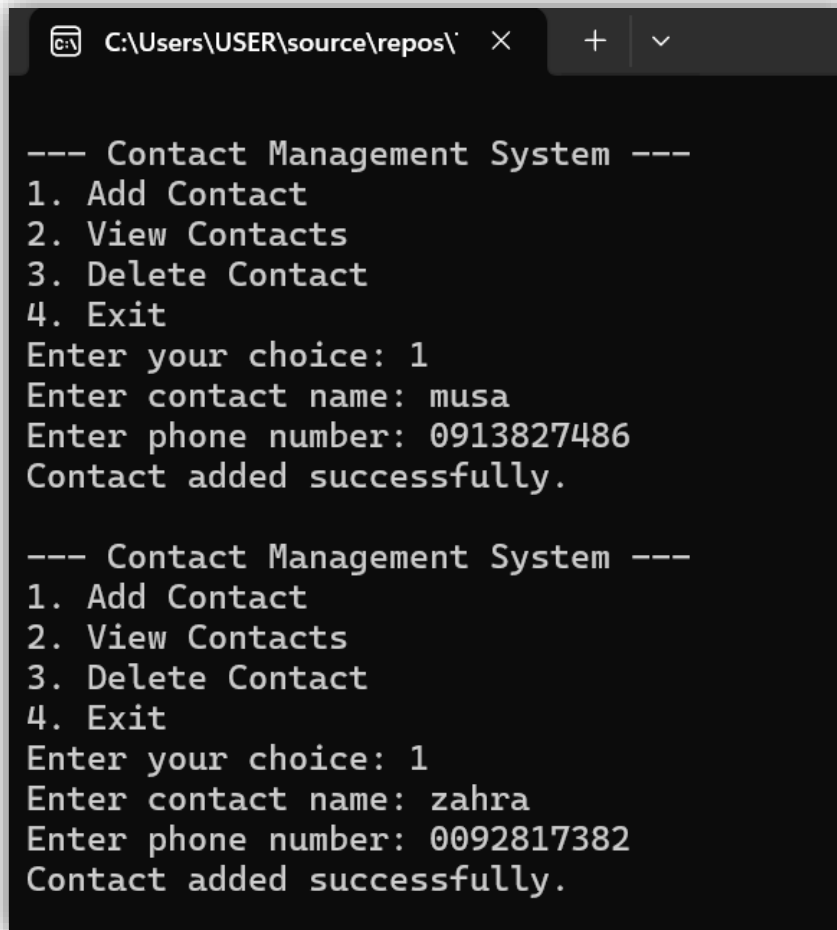
        switch (choice) {
            case 1:
                addContact(contacts);
                break;
            case 2:
                viewContacts(contacts);
                break;
            case 3:
                deleteContact(contacts);
                break;
            case 4:

```

```
        cout << "Exiting..." << endl;
        break;
    default:
        cout << "Invalid choice. Please try again." << endl;
    }
} while (choice != 4);
system("pause");
return 0;
}
```

## **Testing and Results**

- Added multiple contacts and verified that they are stored and displayed correctly



```
C:\Users\USER\source\repos\ >
--- Contact Management System ---
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 1
Enter contact name: musa
Enter phone number: 0913827486
Contact added successfully.

--- Contact Management System ---
1. Add Contact
2. View Contacts
3. Delete Contact
4. Exit
Enter your choice: 1
Enter contact name: zahra
Enter phone number: 0092817382
Contact added successfully.
```

- Attempt to view added contacts

```
--- Contact Management System ---  
1. Add Contact  
2. View Contacts  
3. Delete Contact  
4. Exit  
Enter your choice: 2  
Name: musa, Phone Number: 0189278653  
Name: zahra, Phone Number: 0918286235  
Name: zavi, Phone Number: 0917382163
```

- Attempt to delete an existing contact and verified that it's removed

```
Enter your choice: 3  
Enter contact name to delete: musa  
Contact deleted successfully.  
  
--- Contact Management System ---  
1. Add Contact  
2. View Contacts  
3. Delete Contact  
4. Exit  
Enter your choice: 2  
Name: zahra, Phone Number: 0918286235  
Name: zavi, Phone Number: 0917382163
```

- Attempt to delete a non-existent contact and verified the error message

```
--- Contact Management System ---  
1. Add Contact  
2. View Contacts  
3. Delete Contact  
4. Exit  
Enter your choice: 3  
Enter contact name to delete: sidra  
Contact not found.
```

## **Conclusion**

The Contact Management System successfully achieved its goal of providing a simple and effective way to manage contacts using C++. Through this project, I applied key programming concepts such as object-oriented design and dynamic data structures. The system is user-friendly and fulfills the basic requirements of adding, viewing, and deleting contacts. This experience enhanced my understanding of software development and prepared me for more complex programming challenges in the future.