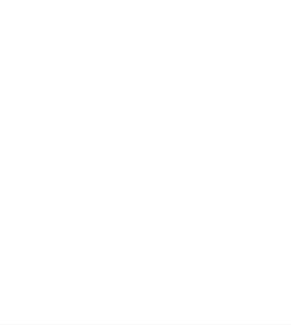
1



-

**Project Report**

Title

**“Telephone Directory System”**

|  |  |
| --- | --- |
| **i.**  **ii. iii.**  **iv.**  **v.**  **vi.**  **vii.**  **viii.**  **ix.**  **x.** | **Table of Contents**          **Abstract ----------------------------------------------- 04**  **Introduction ------------------------------------------ 04**  **Flow Chart -------------------------------------------- 05**  **Software Requirements --------------------------- 05**  **Hardware Requirements -------------------------- 05**  **Project Description --------------------------------- 06**  **Code Snippets ---------------------------------------- 06**  **Screen Shots+ Explanations ---------------------- 09**  **Conclusion -------------------------------------------- 14**  **References -------------------------------------------- 14**  3 |

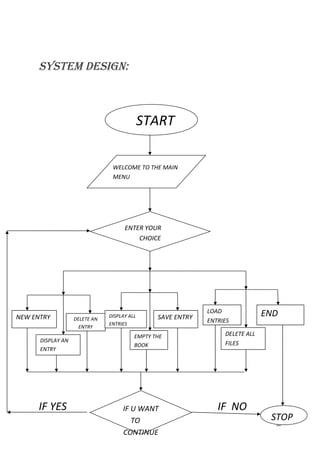
1. **Abstraction:**

To construct this whole directory system, we decided to use the best data structure which we have studied during our semester and by implementing the single link list we create this system. The main reason behind using this data structure is that firstly, it is a linear data structure and secondly its main advantage is it helps for us in a quick insertion and a deletion. A linked list, or one-way list is a linear collection of data elements called nodes, where linear order is given by means of pointer. A linked list is a series of connected nodes. Each node contains at least a piece of data (any type) and a pointer to the next node in the list. Head: pointer to the first node and the last node points to null.

1. **Introduction:**

In a telephone directory system, we build a telephone book program in C++. It stores the contacts of the people in the directory by their names and by providing their phone numbers. The basic use of the telephone directory system is to help people to stay connected with their friends, relatives and with their loved ones and instead of checking or looking the whole contacts in the directory they just search them by their added or saved information which saves their time and it helps to maintain their data properly.

1. **Flow Chart:**



1. **Software Requirements:**

 Visual Studio 2019

**5. Hardware Requirements:**

 Laptop/ PC

1. **Project Description:**

Our project is based on C++ programming language. So, for this project you must know about appropriate Data Structures and Algorithms and their proper use in C++ programming language. Some of the data structures are graph, stack, Binary Search tree, queue, linked list, hash table, arrays but for this project the main data structure which we must know about is linked list and in our project, we implement single linked list by using the C++ programming language.

1. **Code Snippets:**

#include <iostream>

#include <string>

using namespace std;

// Define the node struct for the single linked list struct Node

{ string name; string phone;

Node\* next;

};

// Define the linked list class class LinkedList

{ private: Node\* head;

public:

// Constructor to initialize an empty linked list

LinkedList()

{ head = NULL;

}

// Function to insert a new entry at the end of the list void addRecord(string name, string phone)

{

Node\* newNode = new Node; newNode->name = name; newNode->phone = phone; newNode->next = NULL;

if (head == NULL)

{ head = newNode;

} else {

Node\* current = head; while (current->next != NULL)

{

current = current->next;

}

current->next = newNode;

}

}

// Function to update an entry by name void updateRecord(string name, string phone)

{

Node\* current = head;

while (current != NULL && current->name != name)

{ current = current->next;

}

if (current != NULL)

{

// Update the phone number of the current entry current->phone = phone;

}

}

// Function to search for an entry by name string searchRecord(string name)

{

Node\* current = head;

while (current != NULL && current->name != name)

{ current = current->next;

}

if (current != NULL)

{ return "\t Phone Number : " + current->phone;

} else

{ return "\t Record Not Found ";

}

}

// Function to delete an entry by name void delRecord(string name)

{ if (head == NULL)

{ return; }

if (head->name == name)

{

Node\* temp = head; head = head->next; delete temp; return;

}

Node\* current = head;

Node\* predecessor = NULL;

while (current != NULL && current->name != name)

{ predecessor = current; current = current->next;

}

if (current == NULL)

{ return;

}

predecessor->next = current->next; delete current;

}

void displayAllRecords()

{

Node\* current = head; while (current != NULL)

{ cout << "Name : " << current->name << " | "; cout << "Phone Number : " << current->phone << endl; current = current->next;

}

}

}; int main()

{

// Create an empty linked list

LinkedList directory;

int choice = 0; string name, n;

while (choice != 6)

{ cout << "\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl; cout << "\t \t Telephone Directory \n";

cout << "\t Press 1 to Add a New Record in Directory " << endl; cout << "\t Press 2 to Display all Records " << endl; cout << "\t Press 3 to Update a Record " << endl; cout << "\t Press 4 to Delete a Record " << endl; cout << "\t Press 5 to Search a Record " << endl; cout << "\t Press 6 to Exit the Directory " << endl;

cout << "\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl; cin >> choice;

switch (choice)

{ case 1:

system("cls");

cout << "\t----Adding a New Record----\n " << endl; cout << " Enter Name : ", cin >> name; cout << " Enter Phone Number : ", cin >> n; directory.addRecord(name, n);

cout << "\n\t----Record Added Successfully----" << endl; break;

case 2:

system("cls");

cout << "\t Displaying all Records :- \n\n"; directory.displayAllRecords(); break;

case 3:

system("cls");

cout << "Enter Name of Record that is to be updated "; cin >> name;

cout << "Enter New Phone Number "; cin >> n;

directory.updateRecord(name, n); break;

case 4:

system("cls");

cout << "Enter Name of Record that is to be Deleted "; cin >> name; directory.delRecord(name);

cout << "\t----Record Deleted Successfully----\n"; break;

case 5:

system("cls");

cout << "\t----Searching Records----\n";

cout << "\t Enter Name of Record that is to be Searched : "; cin >> name;

cout << directory.searchRecord(name); cout << " " << endl; break;

case 6:

cout << "\t----Exit----\n"; break;

default:

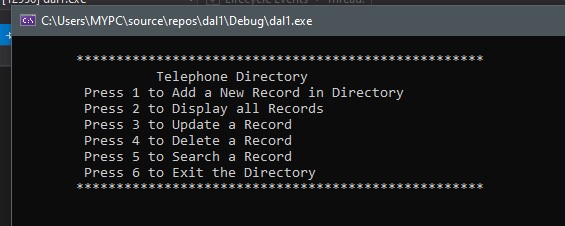
break;

} } return 0;

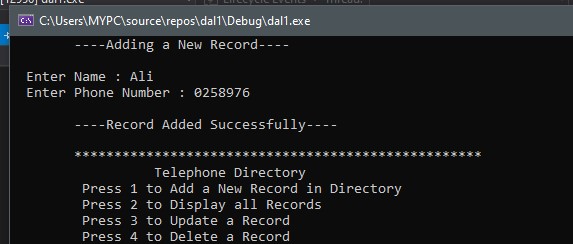
}

**8. Screen Shots + Explanations:**

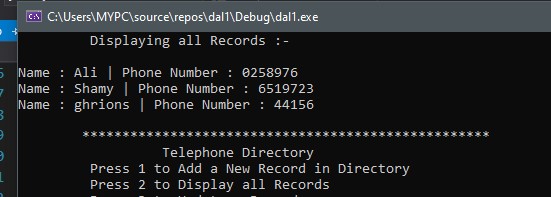
Now this is the first screen shot of the main screen of our telephone directory management system in which we propose a menu driven program in which you see a various different operations like to add, display, update, delete, search and to exit from the program.



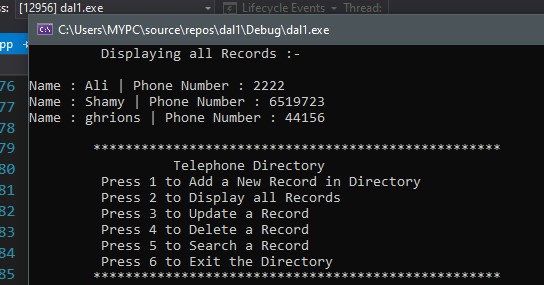
Now this is the second screen shot of the management system in which we have added a record and you have to provide the user name with their number and after giving the information the users record will be saved as you see in the screen shot below.



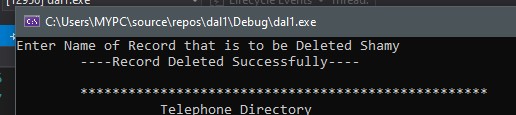
Now this is the third screen shot of the management system, this screen appears by pressing the second option in which you asked to display all the records in the directory system and it will show you all records as given below.



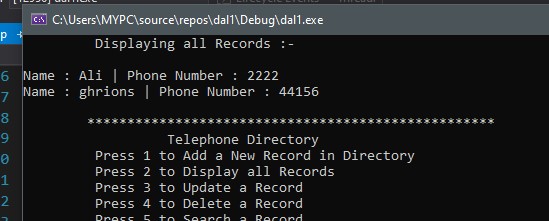
Now this is the fourth screen shot of the management system, you will again see the display all records message but with a bit difference, and difference is that it will appear after the updatation of the record like in the below screen shot you will see that the Ali named user phone number has been updated and saved automatically.



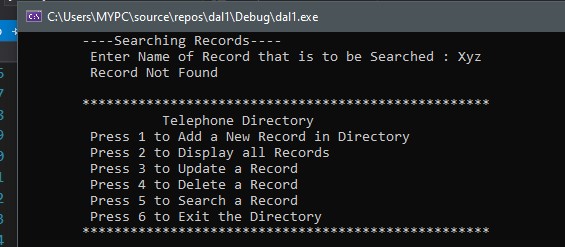
Now this is the fifth screen shot of the management system you will see the next operation of deletion in which the user has been deleted and it will display a message the record deleted successfully.



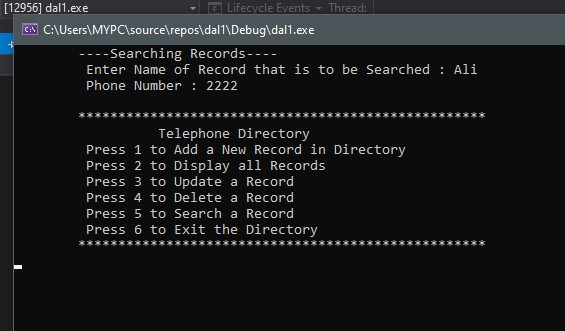
Now in this sixth screen shot of the management system you see the total records of the directory system after the deletion of one user.



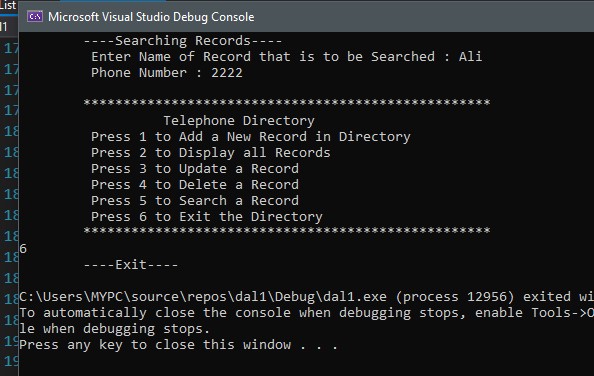
In this seventh screen shot of the management system this different message will appears when you want to search the user in the directory system and the search input is invalid and not present in the directory system, then it will display a message like record not found.



And in this screen shot of the management system when your input is valid and the record is present in the system then it will display you the correct record of that user.



And this is the last screen shot of our management system and it will appear only when you have a desire to exit from this beautifully maintained and smart telephone directory management system and at the end it will display you a message exit.



**9. Conclusion:**

Hence it is concluded that our project follows the single link list data structure and by using link list we make telephone directory system which manages all the records and contacts safely because it is far much better than the manual and work more efficiently. So, by this project we can apply the appropriate data structure and understands the behavior of single link list this project might be helpful from future perspective.