A Mini Project Report

On

"PHONEBOOK MANAGEMENT SYSTEM"

submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

Submitted by

S.SAMANTH	1608-17-733-018
T.SAIDEEP REDDY	1608-17-733-007
G.YASHWANTH REDDY	1608-17-733-024

Under the Guidance of

Dr. G. SHYAMA CHANDRA PRASAD

Associate professor

Department of CSE

Department of Computer Science andEngineering

Matrusri Engineering College

(Affiliated to Osmania University, Approved by AICTE)

Saidabad, Hyderabad – 500059 (2018-2019)

Department of Computer Science and Engineering

Matrusri Engineering College

(Affiliated to Osmania University, Approved by AICTE)
Saidabad, Hyderabad - 500059

CERTIFICATE

This is to Certify that A Mini Project report entitled **"PHONEBOOK MANAGEMENT SYSTEM"**is being submitted by Samanth **(**1608-17-733-018**),** Saideep Reddy **(**1608-17-733-007**),** Yashwanth Reddy (1608-17-733-024) in partial fulfillment of the requirement of the award for the degree of Bachelor of Engineering in "Computer Science and Engineering" O.U., Hyderabad during the year 2018-2019 is a record of bonafide work carried out by him/her under my guidance. The results presented in this mini project report have been verified and are found to be satisfactory.

Project Coordinator Project Guide HOD

Mrs.M.Priyanka	**Dr.G.Shyama Chandra**	**Dr.P.Vijayapal**
Rao	**Prasad**	**Reddy**
Assistant Professor	**Associate Professor**	**Professor & Head**
Dept. of CSE	**Dept. of CSE**	**Dept. of CSE**

External Examiner(s)

CONTENTS

	Acknowledgement	V
	Abstract	VI
	List of Figures	VII
S.No	**Chapter**	**Page No**
1.	**INTRODUCTION**	**1**
	1.1 Project aim & Objective	2
	1.2 Existing System	2
	1.2.1 Disadvantages of Existing System	2
	1.3 Proposed System	3
	1.3.1 Advantages of Proposed System	3
	1.4 Scope and Importance	4
	1.5 System Requirements	5
	1.5.1 Software Requirements	5
	1.5.2 Hardware Requirements	5
2.	**ARCHITECTURE & IMPLEMENTATION**	**6**
	2.1 Modules	

2.1.1 Admin login module 6

2.1.2 User login module 6

2.2 Program Analysis 7

2.2.1 Header Files

2.3 File/Maps/Pointers/Classes

2.3.1 File Handling

2.3.2 Maps

2.3.3 Pointers

2.4 Implementation 8

2.5 Code 9

3. SCREENSHOTS 16

4. CONCLUSION 19

5. FUTURE ENHANCEMENT 20

6. REFERENCES 20

ACKNOWLEDGEMENT

It is our privilege and pleasure to express our profound sense of respect, gratitude and indebtedness to our guide Dr. G. Shyama Chandra Prasad, Associate professor, Department of Computer Science and Engineering, Matrusri Engineering College, for his/her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere thanks to mini project coordinator **Mrs. M.Priyanka Rao** Assistant professor, Department of Computer Science and Engineering, Matrusri Engineering College, for her valuable suggestions and constant help in completing the work.

We express our sincere gratitude to **Dr. P. Vijayapal Reddy**, Professor & Head, Department of Computer Science and Engineering, Matrusri Engineering College, for his precious suggestions, motivation and co-operation.

We extend our sincere thanks to **Dr. D. Hanumantha Rao**, Principal, Matrusri Engineering College, Saidabad, Hyderabad, for his encouragement and constant help.

We extend our sincere thanks to all the teaching and non-teaching staff of CSE Department for their support and encouragement.

Last but not least, we wish to acknowledge our friends and family members for giving moral strength and helping us to complete this dissertation.

ABSTRACT

Phonebook management system is a simple console application without graphics ,developed using c++ programming language and a project which aims in developing a computerized system to maintain all the contacts of a organization or a personal phonebook. The Project is developed in c++, which mainly focus on basic operations like Adding a Contact, Editing a contact, Deleting a contact, Searching a Contact, Super searching a Contact, Deleting Multiple Contacts, Listing of all Contacts. This Mini project utilizes Various aspects of c++ functions, pointers, File handling, Maps etc. This project is developed to reduce the paper work and to increase the Efficiency in searching the Contacts. It reduces the human effort.

LIST OF FIGURES

S.No	Fig No.	Name of the Figure	Page No
1	3.1	Login	16
2	3.2	Adding a contact	16
3	3.3	Adding a contact	17
4	3.4	Deleting a contact	17
5	3.5	Searching	18
6	3.6	Saving into file	19

CHAPTER 1

INTRODUCTION

It's a well organized software solution for a phonebook management system. It helps to provide information of any contact present in the phonebook to the user who uses it. It keeps a track of contacts added and deleted. Allows users to virtually browse the contacts without doing so manually. In this project, users can add a new phone record, display existing phone records, search a particular phone record and delete phone records. This simple project will teach you file handling operations such as how to add, search, modify, list and delete records using file. It allows both admin and student to search for the desired contact. This task if carried out manually will be tedious and includes chances of mistakes. It reduces the paper work and increases efficiency in searching. Advanced search helps in finding the contact by giving a single phrase of name or phone number. Errors can be minimized by computerizing the phonebook. It is a user friendly system. With multiple delete option we can delete multiple contacts at one point of time.

1.1 PROJECT AIMS AND OBJECTIVES

The aims objectives are as follows : To increase the secure features of the phonebook by creating User login and Admin login. User can login using user login to search phone contacts. Admin can login with Admin login by providing the security password.

There is more accuracy and better error handling system. Only Admin can edit the contacts and delete the contacts.

1.2 EXISTING SYSTEM

In Our existing ,we use linked list to store the information it takes more time to search the contacts because we need to traverse whole linked list to check the contact.we can delete only one contact at a time. In searching we need to give full name or phone number to search a particular contact.

1.2.1 Disadvantages of the Existing System

Some of the problems being faced in the existing system are as follows:

1 Searching a phone contact is not efficient.

2 Multiple deletion is not possible in this existing system.

3 There is no super searching in the existing system.

4 everyone can access the phonebook details.

1.3 PROPOSED SYSTEM

The proposed system is computerized phonebook management is designed to help the organization to keep phonebook data secure or for the personal use.It contains super search to search the contacts efficiently and it is user friendly.It makes phonebook management easy and efficient.

1.3.1 Advantages of the proposed System.

- Helps to save contact details in phonebook virtually.
- It removes manual process of storing the contacts details.
- It reduces the paper work and time.
- Admin can only access the operations like Adding, Editing, Deleting the Contact details.
- Users can only access Searching process.
- It is a User friendly Environment.
- Security features are very much provided in the System.

1.4 SCOPE & IMPORTANCE

This system supports adding a contact, Editing a contact, Searching a contact, Listing a contact. This system has applications in the various sectors like Educational Institutions, Industries. This system has Advanced search process which can even search by a single phrase also. This is more secure only Admin can access the phonebook management part in the system. The system would provide basic set of features to Add and Update contacts. It provides "better and efficient" service to the users. Provide facility for proper monitoring reduce paper work and provide data security.All details available on a click.

1.5 SYSTEM REQUIREMENTS

1.5.1 Software Requirements

Operating system : Windows Xp or Later versions of windows

IDE : Dev c++

1.5.2 Hardware Requirements

Processor: Intel Pentium core or later versions

Hard Disc: 500GB

RAM : 1GB/2GB/4GB/8GB

CHAPTER 2

ARCHITECTURE

This application is brought in C++ programming language The C++ programming language is used for developing system applications that forms a major portion of operating system such as Windows,Unix and Linux. Many of today's operating systems, system drivers, browsers and games use C++ as their core language. This makes C++ one of the most popular languages today.

2.1 MODULES

2.1.2 Admin Login Module

After entering to the homepage of the website , Admin can enter the user name as admin and then can enter security password. If the given credentials are correct then admin logged in.Then admin can add, delete, edit, search a contact in the phone book.

2.1.2 User Login Module

After entering to the homepage of the website , by entering username as user we get access to the user login module. User can only have access to search the contacts. The user can search by name or by phone number.

2.2 Program Analysis

2.2.1 Header Files

The header files used in the program can be listed as:

1) #include<iostream>

It is used in C++ in order to include the header file "iostream" in the program. Iostream is used to invoke the commonly used functions like Cout, Cin in a C++ program. Iostream stands for input output stream.

2) #include<sstream>

In the C++ programming language, <sstream> is a part of the C++ Standard Library. It is a header file that provides templates and types that enable interoperation between stream buffers and string objects.

3) #include<map>

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values. these is used for storing the information temporarily.

4) #include<algorithm>

The header <algorithm> defines a collection of functions especially designed to be used on ranges of elements. A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or an instance of some of the STL containers.

5) #include<string>

#include<string> is used for the string operations.

6) #include<fstream>

It is used for file input and file output operations.

7) #include<windows.h>

It contains the functions gotoxy() and console color functions

8) #include<conio.h>

It contains the functions like getch(), getline() etc.

2.3 FILES/MAPS/POINTERS

2.3.1 FILE HANDLING

A file represents a sequence of bytes on the disk where a group related data is stored. File is created for permanent storage of data. It is a readymade structure. In C++ language, we use a class pointer of file type to declare a file.Here at the end we copy the contact details and when program is opened it recovers the data which is there already.

2.3.2 MAPS

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. These is used for storing the information temporarily.Here we use phone number, first-name, last-name as keys and a pointer which holds data of person as value.

2.3.3 POINTERS

A pointer is avariable which contains the address in memory of another variable .we can have a pointer to any variable type. The unary or monadic operator & gives the “address of variable”.The indirection or dereference operator* gives the “contents of an object pointed to by a pointer”.

2.4 IMPLEMENTATION

- Add a new contact: Admin only can add a contact by logging in through the admin login
- Editing a contact:
  Admin only can edit a contact by logging in through the admin login
- Listing all contacts:
  All contacts can be listed at one point of time by clicking list option
- Deleting contacts:

Multiple contacts can be deleted by clicking multiple delete option

- Searching contacts:

  Searching can be done through a special feature called advanced search .contact can be searched by giving single phrase also.

2.5 CODE

Main block

```
int main()
{
    HWND console = GetConsoleWindow();RECT r;GetWindowRect(console, &r);
    MoveWindow(console, r.left, r.top, 1275, 580, TRUE);string Username=login();
    Person obj;char ch;obj.recoverformfile();
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
    if(Username=="admin")obj.menu();
    if(map1.size()==0)
    {
       system("cls");cout<<"zero contacts \n";
       SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );system("pause");return 0;
    }
    if(Username=="user")
    {
        do{
            system("cls");SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
            cout<<"[1].search [2].exit \nenter your choise   :  ";ch=getch();
             switch(ch)
             {
             case '1':{
                        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
                        multimap<string,Person*>::iterator itr=obj.search1();
                        if(itr!=map1.end()) obj.disp(itr->second);
                        else
                        {
                          SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                          cout<<"NOT FOUND"<<endl; getch();
                        }
                        break;
                        }
              case '2':exit(0);
              default : cout<<"Invalid";cout<<"\n \n try again in 2 seconds";Sleep(2000);
             }
        }while(1);
    }
}
```

Class Person And Maps

```
class Person                                        //class
{   string phoneno,fname,lname,email;
public:
    void recoverformfile();                         //recover data from file and add to maps
    void disp(Person *);                            //for dispaly
    void add();                                     //for adding a new contact (name,phoneno,email)
    bool check(multimap<string,Person*>::iterator ,string );    //tokenizing given string,contact name returns whether given string tokens are
    void supersearch();                             //display all names with given name or phone number even by phrase
    multimap<string,Person*> ::iterator search1();  //for searching and returing iterator pointing to it,if not returns map1.end()
    multimap<string,Person*> ::iterator searchbyname();     //searching by name
    multimap<string,Person*> ::iterator searchbyphoneno();  //searcing by phone number
    void draw_fill(multimap<string,Person*> &);     //prints border and fill them with given map
    void changedetails();                           //changing details
    void get_name(Person *);                        //get's name
    bool get_phonenumber(Person * );                //get's phone number returns 0 if you exit
    void get_email(Person * );                      //get's email
    void deletephoneno();                           //delete phone number (calls delete in maps)
    void deleteinmaps(Person *);                    //deleting in maps
    string nameset( string );                       //changing the printed string by set of opreations
    void writeinfile();                             //writing in file
    void multidelete();                             //for deleting more numbers at a time
    void menu();                                    //for menu
};
multimap<string,Person*> map1;multimap<string,Person*> map4;    //map1 for storing by first name.map4 is for supersearch
multimap<string,Person*> map2;                                  //map2 for storing by last name
multimap<string,Person*> map3;multimap<string,Person*> map6;    //map3 for storing by phone number.map6 is for super search
multimap<string,Person*> map5;                                  //for super search
```

Menu block

```
void Person::menu()
{
    system("cls");
    char choise;
    while(1)
    {
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0x9 );
        cout<<"                                                  phonebook   management ";
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
        cout<<"\n\n[1].Add a phone number\n[2].Edit a phone number\n[3].search\n[4].Delete a phone number\n[5].multi delete\n[6].list\n[7].super search";
        cout<<"\n[8].exit  for writing into file\nEnter you choise    :";
        choise=getch();       cout<<choise<<endl;
        switch(choise)
        {
            case '1':{ border();add(); gotoxy(7,29);            break;}
            case '2':{ changedetails();gotoxy(7,29);           break;}
            case '3':{
                     map<string,Person*>::iterator itr=search1();  system("cls");
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
                     if(itr!=map1.end()) disp(itr->second);
                     else
                     {
                       SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                       cout<<"NOT FOUND"<<endl;
                     }
                     cin.ignore();
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );break;
                    }
            case '4':{
                     deletephoneno();
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                     cout<<"    THE NUMBER IS DELETED";  gotoxy(7,29);
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
                     cin.ignore();
                     break;
                    }
```

```
            case '5':{int co=map1.size()*2+10;
                     if(map1.size()!=0)
                     {
                       multidelete();
                       gotoxy(7,co);
                       cout<<" SELECTED NUUMBERS ARE DELETED ";  gotoxy(7,29);
                      }
                     else
                     {
                       SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                       cout<<"no contact "<<endl;
                     }
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
                     gotoxy(7,co+2);                                    break;
                    }
            case '6':{
                     draw_fill(map1);
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
                     gotoxy(4,map1.size()*2+11);                        break;
                    }
            case '7':{  map4=map1;map6=map3;supersearch();              break;}
            case '8':{
                     writeinfile();
                     map1.clear();map2.clear();map3.clear();map4.clear();map5.clear();map6.clear();
                     exit(0);                                           break;
                    }
            default:{
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );gotoxy(0,30);
                     cout<<"invalid ";gotoxy(0,31);                     break;
                    }
        }
        system("pause");
        system("cls");
    }
}
```

Adding Contact

```
void Person::add()                                          //for adding new contact
{
    Person *person=new Person;
    border();
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(68,5);
    cout<<"NEW RECORD";
    bool t=get_phonenumber(person);if(t==0) return ;            //if want to exit add returns 0 then t==0 go's back to menu
    get_name(person);
    get_email(person);

    gotoxy(59,28);
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
    cout<<"ENTRED THE NUMBER SUCCESSFULLY";
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
}
```

Get Number

```
bool Person::get_phonenumber(Person *person)                    // for phone number
{
    string phonenumber;
    int check=1;
    while(check==1)
    {
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(7,8);
    cout<<"enter the phone number   :                                  ";
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(35,8);
    getline(cin,phonenumber);
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
    for (int i = 0; i < phonenumber.length(); i++)              //checking the number or not , if any alphabet ask again
        if (isdigit(phonenumber[i]) == 0) check=0;

    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );gotoxy(85,7);
    if(phonenumber.length()==0) {cout<<"please enter name                          ";check=1;}
    else if(check==0){cout<<"number contains  letter or space try again\n";check=1;}
    else if(check==1) {cout<<"                                                    ";check=0;}
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
    }
    if(map3.find(phonenumber)==map3.end())                      //if number not present in list it adds the number
    {
       person->phoneno=phonenumber;
       map3.insert(pair<string,Person*> (phonenumber,person) );
    }

    else                                                        //present ask's whether to exit or try again
    {
        gotoxy(7,10);   char y;
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
        cout<<"THIS NUMBER ALREADY EXIXTING IN THE LIST\n";
        do{
            SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(7,12);
            cout<<"do you want to [1].try another number  [2].want to exit    :                  ";
            gotoxy(70,12);y=getch();
            switch(y)
            {
              case '1' : {
                gotoxy(7,10);cout<<"                                                              ";
                gotoxy(7,12);cout<<"                                                                    ";
                return(get_phonenumber(person));
                break;
              }
              case '2' :  return 0;
              default  :  { gotoxy(85,7);SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                            cout<<"INVALID PRESS 1 OR 2";
                            SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );break;}
            }
        }while(y!=1 && y!=2);
    }
}
```

Get Name

```
void Person::get_name(Person *person)                          // for name
{
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(7,10);
    cout<<"enter the person's name  :                                    ";
    gotoxy(35,10);  char ch[35],ch2[23];  memset(ch, 0, sizeof(ch));  memset(ch2, 0, sizeof(ch2));  string line,line1,line2;
    do{
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );getline(cin,line);
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
       if(line.length()==0)                                        //if number not entered alerts user
       {
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
        gotoxy(92,7);cout<<" name was not entered ";
        gotoxy(35,10);cout<<"                                  ";gotoxy(35,10);
       }
    }while(line.length()==0);
    gotoxy(92,7);cout<<"                        ";
    int k=line.find(' ',0);                                        //find is there any gap
    if(k==-1)                                                      //if no gap store only first name
    { line1=line;              person->fname=line1;           person->lname="-";
      map1.insert(pair<string,Person*> (line1,person));}
    else                                                           //if gap store in both maps
    { line.copy(ch,k,0);                                    line.copy(ch2,line.length()-k,k+1);
      person->fname=ch;                                     person->lname=ch2;
      line1=ch;                                             line2=ch2;
      map1.insert(pair<string,Person*> (line1,person));     map2.insert(pair<string,Person*> (line2,person));}
}
```

Get Email

```
void Person::get_email(Person *person)                         //for email
{
    string line;  int flage;     char c;
    do{
        flage=1;
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(7,12);
       cout<<"enter email              :                                  ";
       SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(35,12);
       getline(cin,line);gotoxy(92,7);
       if(line.length()==0)                                        //if length is 0 ask again
       {
           SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
           cout<<"continue with out email Y/N";c=getch();if(c!='y') flage=0;
           SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
       }
       else
       {   size_t f= line.find(' ');
           if( f<line.length() )
           { SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
             cout<<"mail ID don't contain space try again"; flage=0;
           }
           else{ cout<<"                                          ";  }
       }
    }while(flage==0);
    person->email=line;
}
```

Search

```
multimap<string,Person*>::iterator Person::search1()           // for searching
{
    char ch,r;
    do
    {
        system("cls");
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(0,2);
        cout<<"SEARCH by   [1].name    [2].phone number   [3].exit  :           ";
        gotoxy(56,2);
        ch=getch();cout<<ch;
        switch(ch)
        {
            case '1':return(searchbyname());break;
            case '2':return(searchbyphoneno());break;
            case '3':return(map1.end());break;
            default :SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );gotoxy(82,0);
                     cout<<"invalid \n";
                     gotoxy(75,1);
                     cout<<"want to exit press 1  : ";
                     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0x7 );
                     r=getch();
                     if(r=='1') return map1.end();
        }

    }while(ch!='1' && ch!='2');
}
```

Search By Name

```
multimap<string,Person*>::iterator Person::searchbyname()       // for search by name
{
    int f=1,i,j;
    string line,line1,line2;
    do{
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(0,5);
    cout<<"enter name     :";
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );
    getline(cin,line);
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
    }while(line.length()==0);
    if(line.length()==0) return map1.end();
    int k=line.find(" ",0);                                    //dividing into first name,last name
    if(k==-1)
    {
        line1=line;line2="  ";
    }
    else
    {
      char ch[35],ch2[35];
      line.copy(ch,k,0);
      line.copy(ch2,line.length()-k,k+1);
      line1=ch;line2=ch2;
        f=0;
    }
    //example:yashwanth reddy line1=yashwanth line2=reddy
    //example:yashwanth       line1=yashwanth line2=" "
    //example:reddy           line1=reddy line2=" "
```

```
multimap<string,Person*>::iterator  itr=map1.find(line1); multimap<string,Person*>::iterator  itr2=map2.find(line2);
if(f)                                                  // if only one word is entered
{
    itr2=map2.find(line1);
    if(itr==map1.end() && itr2==map2.end() )  return map1.end(); //not present in both maps return map.end (null)
    if(itr!=map1.end() && itr2==map2.end())                      //if present in map1 and not in map2
    { int count=0; map4.clear();                                          //counting
      for(multimap<string,Person*>::iterator  itrt=itr ; itrt!=map1.end() &&itrt->second->fname==line1 ;itrt++ )
      { count++; map4.insert(pair<string,Person*>( itrt->first,itrt->second)); }
      if(count==1) return itr;                                 //if only one return itr=map1.find
      else                                                     //more than one draw and ask
      {   draw_fill(map4);
          SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(4,count*2+11);
          cout<<"press the number which one do you want   :        "; int n;
          SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(47,count*2+11);cin>>n;
          while(n>count)
          {
            gotoxy(70,count*2+8);  SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
            cout<<"enter between range  1 to  "<<count;          gotoxy(67,count*2+9);
            cout<<" if you want to exit enter press 0         "; gotoxy(47,count*2+11);
            cout<<"                ";                          gotoxy(47,count*2+11);
            SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );cin>>n;
            if(n==0) return map1.end();
          }
          multimap<string,Person*>::iterator  itrn=itr;for(i=1;i<n;itrn++) i++; map4.clear();   return itrn;
      }
    }

    if(itr==map1.end() && itr2!=map2.end())                  //if present in map2 and not in map1
    {
          int count=0;
          for(multimap<string,Person*>::iterator  itrt=itr2 ; itrt!=map2.end() &&itrt->second->lname==line1 ;itrt++ )
          { count++;map4.insert(pair<string,Person*>( itrt->first,itrt->second));  }
          if(count==1) return itr;                           //if only one return itr=map1.find
          else                                               //if more draw and ask
          {
              draw_fill(map4);
              SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(4,count*2+11);
              cout<<"press the number which one do you want   :        ";int n;
              SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(47,count*2+11);cin>>n;
              while(n>count)
              {
                gotoxy(70,count*2+8);SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                cout<<"enter between range  1 to  "<<count;          gotoxy(67,count*2+9);
                cout<<"  if you want to exit enter press 0         ";gotoxy(47,count*2+11);
                cout<<"                ";                          gotoxy(47,count*2+11);
                SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA ); cin>>n;
                if(n==0) return map1.end();
              }
              multimap<string,Person*>::iterator  itrn=itr;for(i=1;i<n;itrn++) i++;map4.clear();return itrn;
           }

    }


    if(itr!=map1.end() && itr2!=map2.end())   //if present i n both maps   ex:a b,b a; search for 'a' should search in both maps
    {
          int count=0;map4.clear();                                  //counting
          for(multimap<string,Person*>::iterator  itrt=itr ; itrt!=map1.end() &&itrt->second->fname==line1 ;itrt++ )
          { count++; map4.insert(pair<string,Person*>( itrt->first,itrt->second)); }
          for(multimap<string,Person*>::iterator  itrt=itr2 ; itrt!=map2.end() &&itrt->second->lname==line1 ;itrt++ )
          { count++;map4.insert(pair<string,Person*>( itrt->first,itrt->second));  }
          if(count==1) return itr;                          //if only one return itr=map1.find
          else                                              //if more draw and ask
          {
              draw_fill(map4);
              SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(4,count*2+11);
              cout<<"press the number which one do you want   :        ";int n;
              SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(47,count*2+11);cin>>n;
              while(n>count)
              {
                gotoxy(70,count*2+8);SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                cout<<"enter between range  1 to  "<<count;        gotoxy(67,count*2+9);
                cout<<"if you want to exit enter press 0         ";gotoxy(47,count*2+11);
                cout<<"                ";                          gotoxy(47,count*2+11);
                SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );cin>>n;
                if(n==0) return map1.end();
              }
           multimap<string,Person*>::iterator  itrn=map4.begin();for(i=1;i<n;itrn++) i++;map4.clear();return itrn;
           }
    }
}
```

```
    else                                    // if two or more word are entered line1=first string lin2=next all strings
    {
       if(itr==map1.end() && itr2==map2.end()) return map1.end();  //ex:a b search for a b then both first name and second name should b
       int count=0;
       for(multimap<string,Person*>::iterator  itrt=itr; itrt!=map1.end() && itrt->first==line1 ;itrt++)
         if(itrt->second->lname==line2) { count++; map4.insert(pair<string,Person*>( itrt->first,itrt->second));}
       if(count==0) return map1.end();                          //if count=0 retun map1.end()
       if(count==1)
          for(multimap<string,Person*>::iterator  itrt=itr;itrt->second->fname==line1 && itrt->second->lname==line2;itrt++)
                          return itrt;
       else
        {      draw_fill(map4);
               SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );gotoxy(4,count*2+11);
               cout<<"press the number which one do you want   :        ";int n;
               SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(47,count*2+11);cin>>n;
               while(n>count)
               {
                 gotoxy(70,count*2+8);SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
                 cout<<"enter between range  1 to  "<<count;        gotoxy(67,count*2+9);
                 cout<<"if you want to exit enter press 0         ";gotoxy(47,count*2+11);
                 cout<<"                 ";                        gotoxy(47,count*2+11);
                 SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );cin>>n;
                 if(n==0) return map1.end();
               }
               multimap<string,Person*>::iterator  itrn=map4.begin();for(i=1;i<n;itrn++) i++;map4.clear();return itrn;
        }
      }
}
```

Search By Number

```
multimap<string,Person*>::iterator Person::searchbyphoneno()    // for searcing by number
{
    cout<<"\n \n enter phone number:  ";
    string number; SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );
    cin>>number;
    map<string,Person*>::iterator  itr=map3.find(number);
    if(itr!=map3.end()) return itr;
    else return map1.end();
}
```

Deleting

```
void Person::deletephoneno()                                    // deleting contact
{
    system("cls");int ch;
    multimap<string,Person*>::iterator  itr=search1();Person *itr2=itr->second;
    if(itr==map1.end()) cout<<"not found";
    else deleteinmaps(itr2);
}
void Person::deleteinmaps(Person *person)                       // deleting in maps
{
    string p,f,l;
    p=person->phoneno;f=person->fname;l=person->lname;
    multimap<string,Person*>::iterator  itr1=map3.find(p);      //for deleting in number map
    multimap<string,Person*>::iterator  itr2=map1.find(f);      //for deleting in first name map
    multimap<string,Person*>::iterator  itr3=map2.find(l);      //for deleting in second name map
     while(1)
        if(itr2->first==f && itr2->second->lname==l && itr2->second->phoneno==p)
             break;
        else itr2++;
     if(itr3!=map2.end())
       while(1)
           if(itr3->first==l && itr3->second->fname==f && itr3->second->phoneno==p )
              break;
           else itr3++;
    free(itr1->second);
    map3.erase(itr1);
    if(itr3!=map2.end())map2.erase(itr3);
    map1.erase(itr2);
}
```

Multi-Deleting

```
void Person::multidelete( )
{
    map <Person*,int> m; multimap<string,Person*>::iterator p=map1.begin();
    int y=6;     char c;        draw_fill(map1);

    for(int i=1;i<=map1.size();i++)     {gotoxy(5,y);cout<<"   ";y+=2;}

        gotoxy(0,0);SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
        cout<<"enter button for select  backs WINBASEAPI HANDLE WINAPI GetStdHandle (DWORD nStdHandle)e \necs to exit ";
        y=6;
    do{ gotoxy(5,y);c=getch();
        switch(c)
        {
            case 72:{y-=2;p--;gotoxy(5,y);break;}
            case 80:{y+=2;gotoxy(5,y);p++;break;}
            case 13:{SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );cout<<"del";m[p->second]=1;break;}
            case 8:{cout<<"   ";m.erase(p->second);break;}
            case 27:m.clear();return ;
        }
    }while(c!='@');

    for(multimap<Person*,int>::iterator l=m.begin();l!=m.end();)
    {
        multimap<Person*,int>::iterator o=l;
        l++;
        deleteinmaps(o->first);
    }
}
```

Change In Details

```
void Person::changedetails()                          //modifing data
{   string line;    char c;  map<string,Person*>::iterator  itr=search1();
    if(itr==map1.end())
    {   system("cls");SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );
        cout<<"NOT FOUND"<<endl;
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );  return ;    }
    int flage;   Person *p=new Person;  system("cls");
    if(itr!=map1.end())
    {
        do{
        p->phoneno=itr->second->phoneno;p->fname=itr->second->fname;p->lname=itr->second->lname;p->email=itr->second->email;
          SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF );
          cout<<"\n[1].change in name  [2].in phonenumber   [3].change in email  [4].exit   : ";c=getch();cout<<c;flage=0;
          switch(c)
          { case '1':{  cin.ignore();get_name(p);map3.insert(pair<string,Person*> (p->phoneno,p)); break;  }
            case '2':{  cin.ignore();get_phonenumber(p);map1.insert(pair<string,Person*> (p->fname,p));
                        if(p->lname.at(0)!='-' && p->lname.length()!=1) map2.insert(pair<string,Person*> (p->lname,p));
                        break;}
            case '3':{  cin.ignore(); get_email(p);
                        map3.insert(pair<string,Person*> (p->phoneno,p));map1.insert(pair<string,Person*> (p->fname,p));
                        if(line!="-")map2.insert(pair<string,Person*> (p->lname,p));break;}
            case '4': return ;
            default : flage=1;cout<<"invalid ";break;}
        if(c=='1'||c=='2'||c=='3') { deleteinmaps(itr->second);}
       }while(flage);
     SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xF ); gotoxy(0,20);  disp(p);
    }
}
```

Listing

```
void Person::draw_fill(multimap <string,Person*> &map7)         // for drawing and filling in table
{
    system("cls");
    if(map7.size()!=0)
    {
    int count=map7.size();
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xA );gotoxy(66,6);
    cout<<"LIST WITH GIVEN NAME \n"; border(3,3,150,count*2+6);hline(4,5,149);
    gotoxy(4,4);cout<<"S.NO ";gotoxy(10,4);cout<<"NAME ";gotoxy(65,4);cout<<"PHONE NUMBER";
    gotoxy(120,4);cout<<"EMAIL";cout<<endl;
    int y=6,i=1;
    SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0x9 );
    for(multimap<string,Person*>::iterator it=map7.begin();it!=map7.end();it++)
        {
            gotoxy(5,y);cout<<i;
            gotoxy(10,y);cout<<it->second->fname<<" ";
            if(it->second->lname!="-")cout<<it->second->lname;
            gotoxy(65,y);cout<<it->second->phoneno;
            gotoxy(120,y);cout<<it->second->email;
            y+=2;i++;
         }
    }
    else
    {
      SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), 0xC );gotoxy(0,4);
      cout<<endl<<"     zero records ";
    }
}
```

Writing in file

```
void Person::writeinfile()                                    // writing in fire
{
    ofstream myfile("phone book.txt");
    if (myfile.is_open())
    {
       for(multimap<string,Person*>::iterator  itr=map1.begin();itr!=map1.end();itr++)
        {
            myfile<<itr->first<<"\n";
            myfile<<itr->second->lname<<"\n";
            myfile<<itr->second->phoneno<<"\n";
            myfile<<itr->second->email<<"\n";
        }
        myfile.close();
    }
}
```

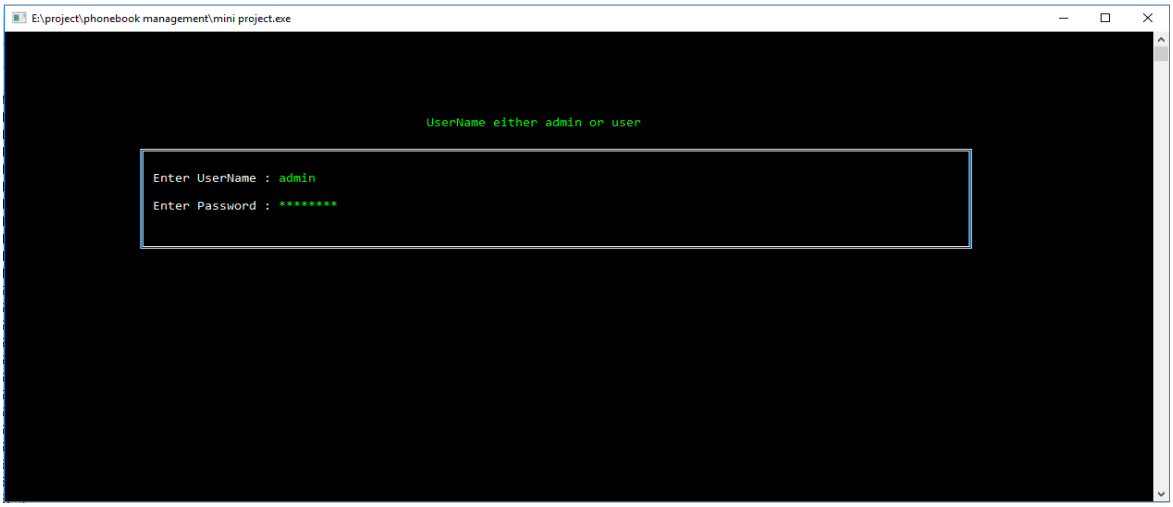
Reading file

```
void Person::recoverformfile()                              // reading from file and updating maps for operations
{
  string line,line2;
  ifstream myfile ("phone book.txt");
  int count=0;
  cout<<endl;
  Person *p=new Person;
  if (myfile.is_open())
  {
    while ( getline (myfile,line) )
    {
        if(count%4==0)p->fname=line;
        if(count%4==1){p->lname=line;line2=line;}
        if(count%4==2)p->phoneno=line;
        if(count%4==3)p->email=line;
        if(count%4==3)
          {
            map1.insert(pair<string,Person*> (p->fname,p));

            if(line2.at(0)!='-' && line.length()!=1) map2.insert(pair<string,Person*> (p->lname,p));

            map3.insert(pair<string,Person*> (p->phoneno,p));
            Person *temp=new Person;
            p=temp;
          }
        count++;
    }
    myfile.close();
  }
}
```

CHAPTER 3

SCREENSHOTS

LOGIN



- Fig 3.1

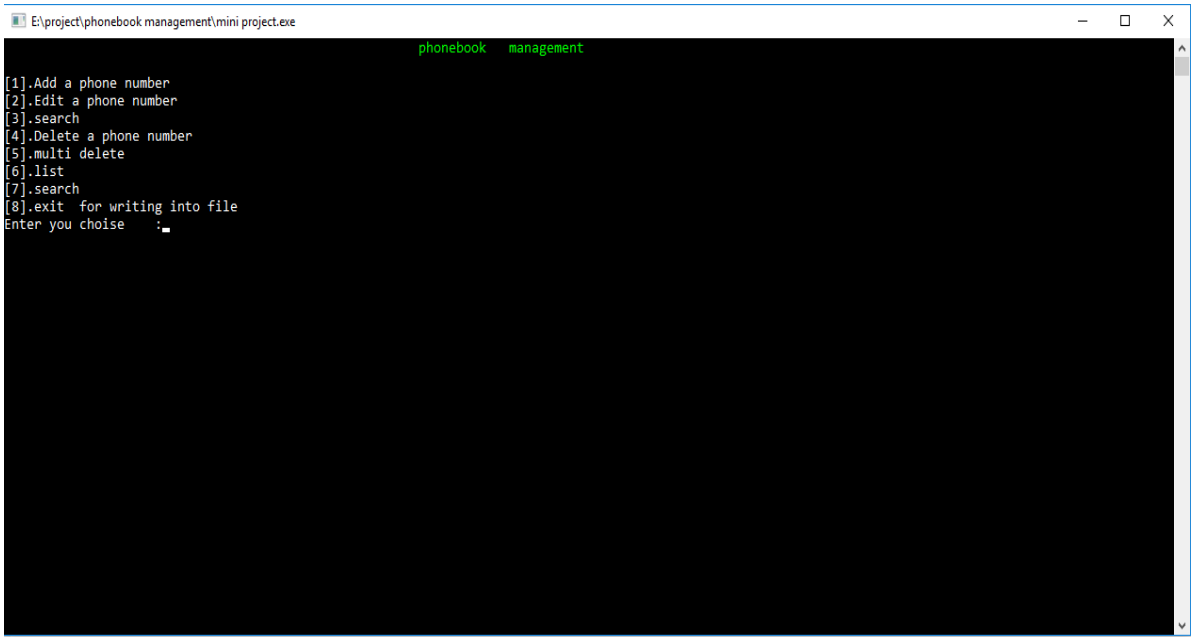
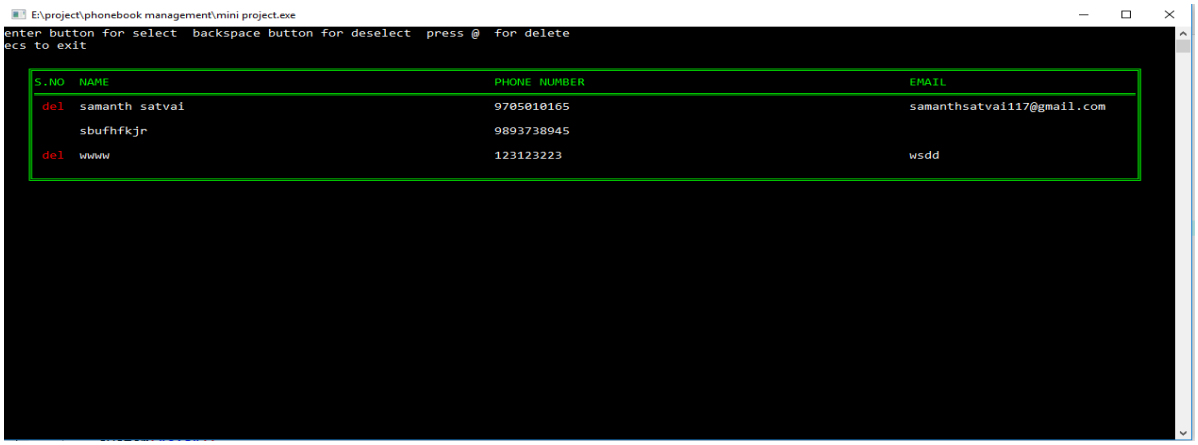


Fig 3.2

ADDING A CONTACT

- Fig 3.3

DELETING CONTACTS



- Fig 3.4

SEARCHING

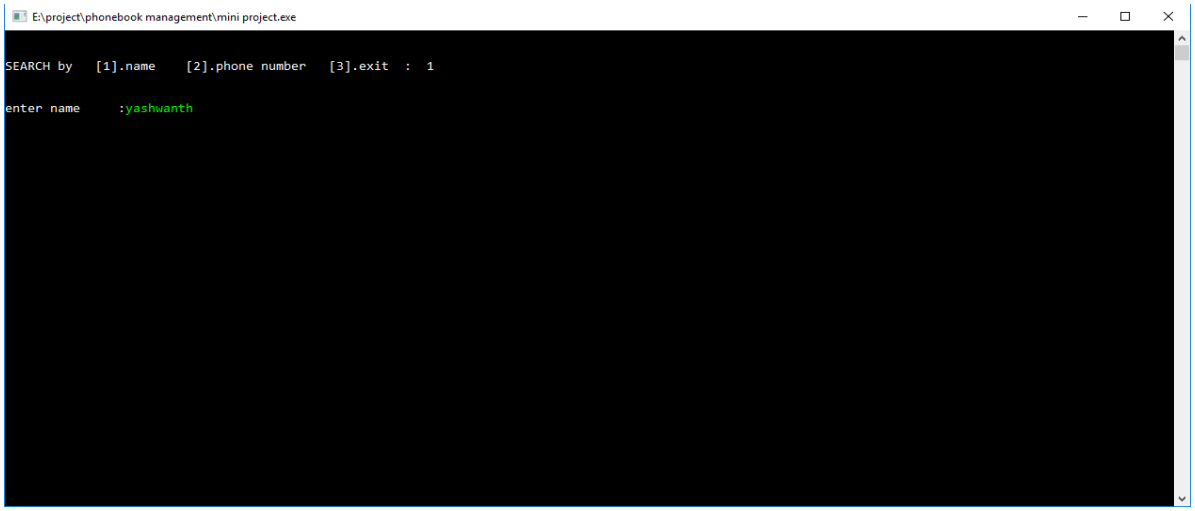
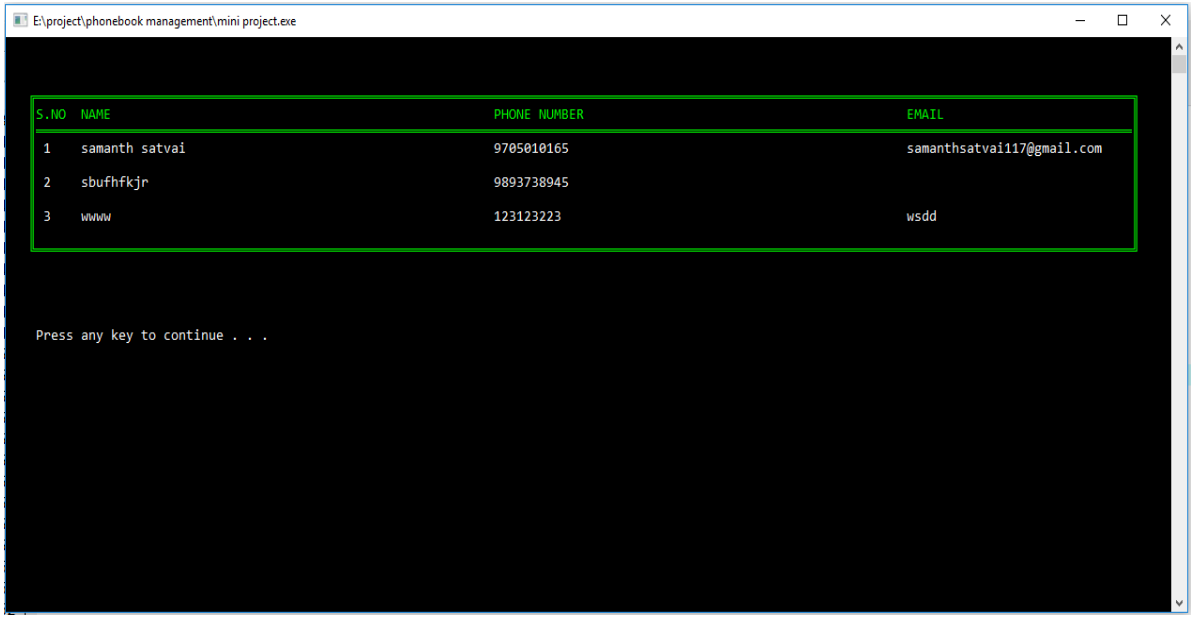


Fig 3.5

LISTING ALL NUMBERS



- Fig 3.6

SAVING INTO A FILE

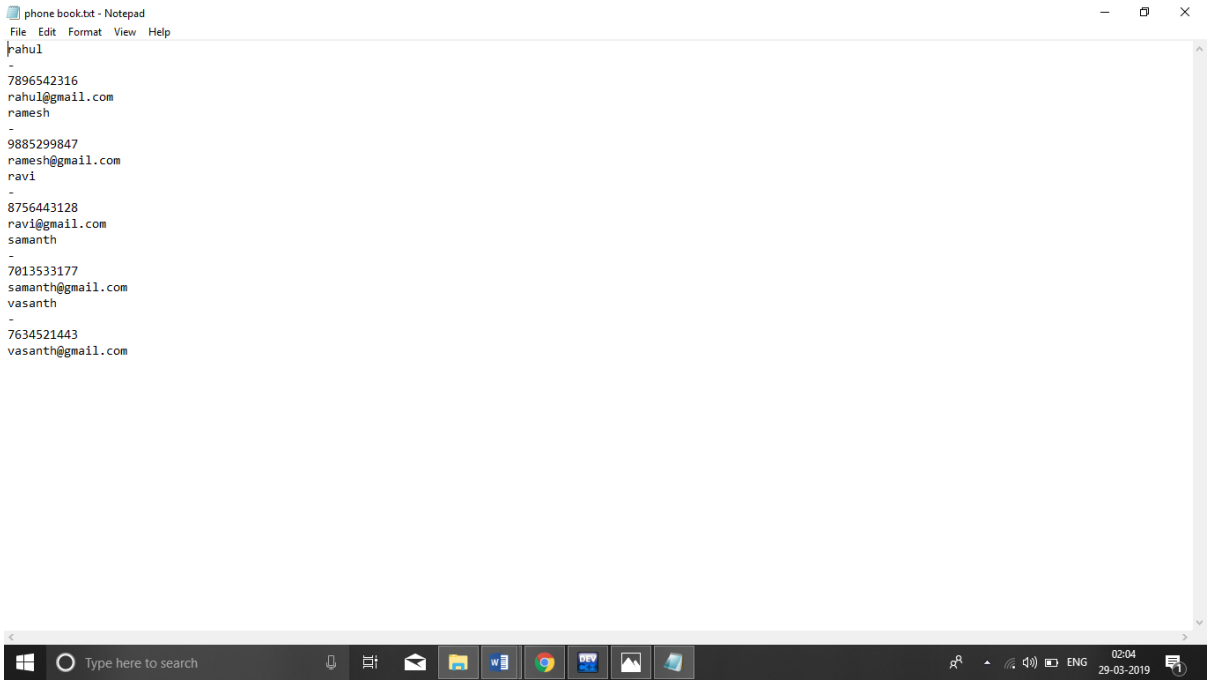


Fig 3.6

CONCLUSION

From a proper analysis of positive points and constraints on the components, it can be safely concluded that the product is a highly efficient based component. This application is working properly and and meeting all the user requirements. The components can be easily plugged into many others systems.

This website provides a computerized version of phonebook management system which will benfit the admin as well as the users of the phonebook.

It makes the entire process online where user can search for contacts, Admin can edit and delete the contacts. It has the for the User login where User can login and can search for the contacts. It has the facility of admin login. This project is user friendly and much more efficient than the existing one.This project which fulfills each users need in the best way possible.

FUTURE ENHANCEMENT

Phonebook management system is itself a complete system, through it has few limitations but it has a lot of future scope and features that could be added to make it more widely acceptable.More number of categiories can be added as part of development. The phonebook management can be designed as an application and be made available smartphones. In the contemporary day, high accessibility can be achieved by creating application on Android And apple os. The programs can be improved further. The increases the accessibility to many more students.

REFERENCES

Phonebook Management System C++ Project

https://www.codewithc.com › Projects › C++ Projects

Jul 17, 2014 - Similar to **Phonebook** and Contact **Management System**, this **phonebook management system** project in **C++** is a simple console application built without graphics.

Phonebook Management System in C/C++ project .

https://www.kashipara.com/project/c-c-/746/phonebook-management-system

May 18, 2016 - **Phonebook Management System** project is a desktop application which is implemented in C/**C++** platform