

INTRODUCTION

1.1 ABOUT THE PROJECT

This **mini project in C Phonebook** allows you to perform simple Phonebook operations like in your mobile. You can add, list, modify, search and delete phonebook related records. File handling and data structure concepts has been extensively used for almost all functions in this mini project.

Phonebook in C is a console application with graphics. The source code is complete and totally error-free. It is compiled in Code::Blocks with GCC compiler. The source code for this project is just over 300 lines, and it is very simple to understand.

1.2 OBJECTIVE & SCOPE OF THE PROJECT

The objective and goals of the proposed system are :

- ◆ Provide simple features of phonebook application.
- ◆ User can add contact record ,edit the saved contact record, delete the saved contact record and search the saved record
- ◆ Personal detail such as name, e-mail, phone number and address are asked while adding a record into the Phonebook.
- ◆ The software will maintain the person details as contact records.

1.3 DEFINITION OF PROBLEM

The main aim of the project entitled “PHONE_BOOK” is to create the contact.

This application will teach you how to add, list, modify or edit, search and delete data to/from the file. I have used many functions in this mini project.

- main() - This function is used to display the main menu.
- void create() - It adds a new Phonebook record.
- void erase() – It deletes a saved Phonebook record.
- void search() – It searches a saved Phonebook record.
- void edit() - It modifies a saved Phonebook record.

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

We have a normal phone book in every phone .This is a phone book application that can be used only in laptop.

2.2 PROPOSED SYSTEM

This is offline phonebook application that can save the several number of contact with name and address. To search any person can be easily done without searching the whole list of contact ,we can easily search by name and number.

The prime advantage is that many people can be classified according to their name or number.

1. ADMINISTRATOR

Administrator can enhance the facilities provided as user interface.

2. USER:

User can act with user interface only ,which provide facilities for adding ,searching deleting and modifying contact.

2.3FEASIBILITY STUDY

The feasibility study is major factor, which contributes to the analysis and development of the system. The decision of the system analyst whether to design a particular system or not depends on its feasibility study.

1. Operational Feasibility:-

- The introduction to this system is not going to hamper any user of the system.
- The proposed system is very flexible and user friendly
- The proposed system produces best results and gives high performance. It can be implemented easily .So this project is operationally feasible.

2. Technical Feasibility:-

- This feasibility deals with technicality of the system. Instead of storing details related to banking in manual way ,this system is fully automated.
- No efficient manpower is required to handle the system.

3. Economic Feasibility:-

- Economic Feasibility deals about the economic impact faced by the organization to implement a new system.
- Economic Feasibility in this project:
 - The cost to conduct a full system investigation is possible.
 - There is no additional manpower requirement.
 - There is no additional cost involved in maintaining the proposed system.

SYSTEM REQUIREMENT AND SPECIFICATIONS

3.1 HARDWARE CONFIGURATIONS

Processor	: Intel Pentium 4 and above processor
RAM	: 512MB to 1GB
Hard Disk	: 1 GB and above
Keyboard	: 108 keys
Clock Rate	: 500 MHz and above
System bus	: 32 bits

3.2 SOFTWARE CONFIGURATIONS

Operating System	: Windows 7/8/8.1/10
Language	: C

3.3 TECHNOLOGY USED

TECHNOLOGY

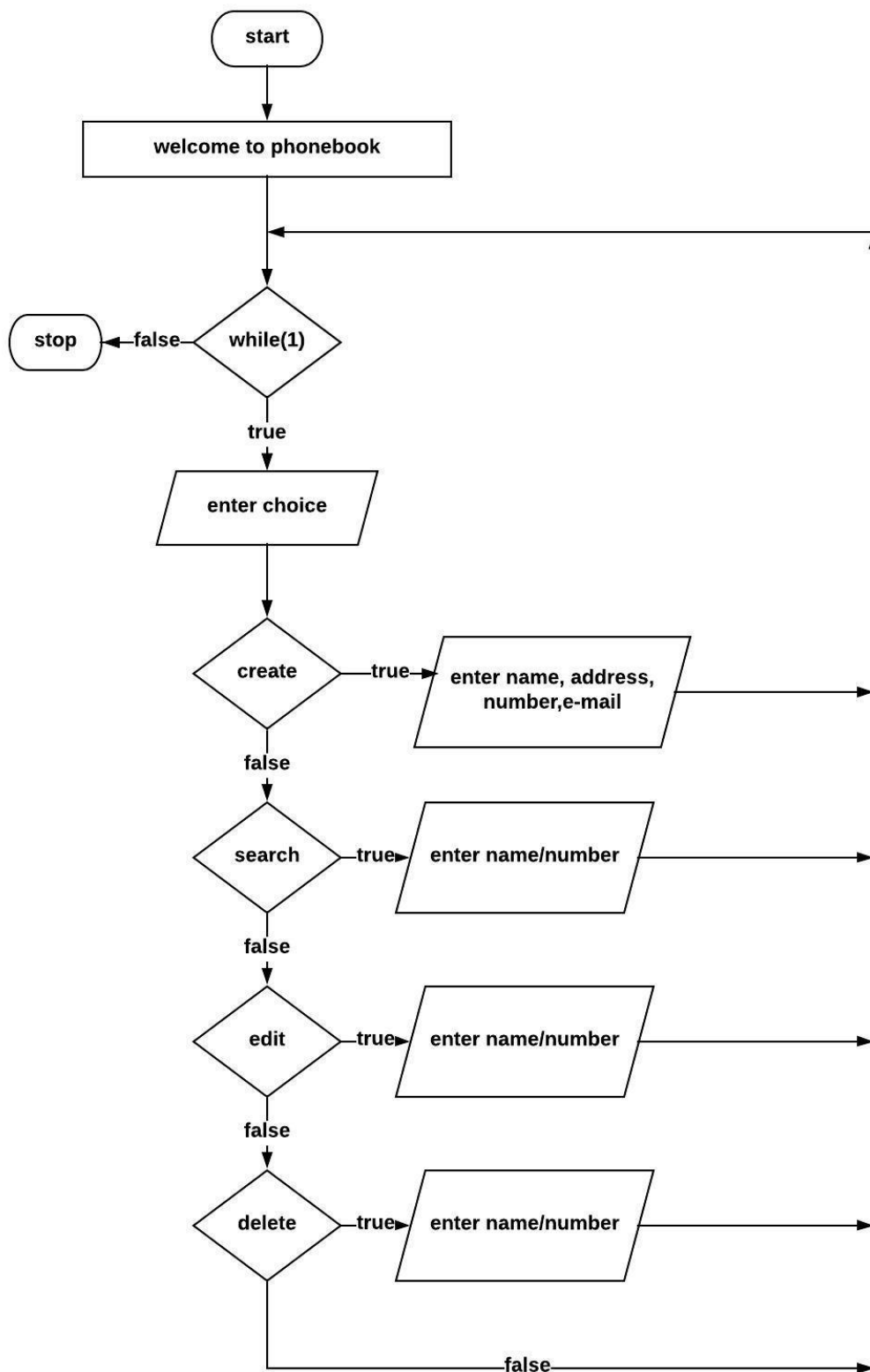
C

3.4 PLATFORM USED

CODE-BLOCK

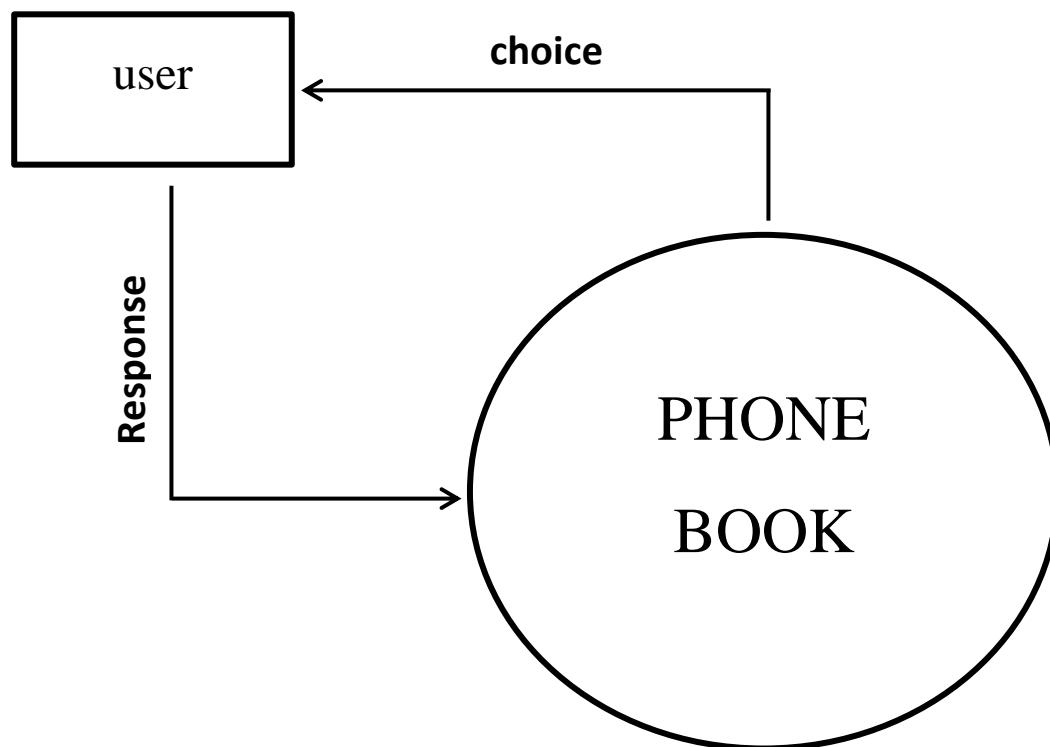
DEV C

SYSTEM DESIGN

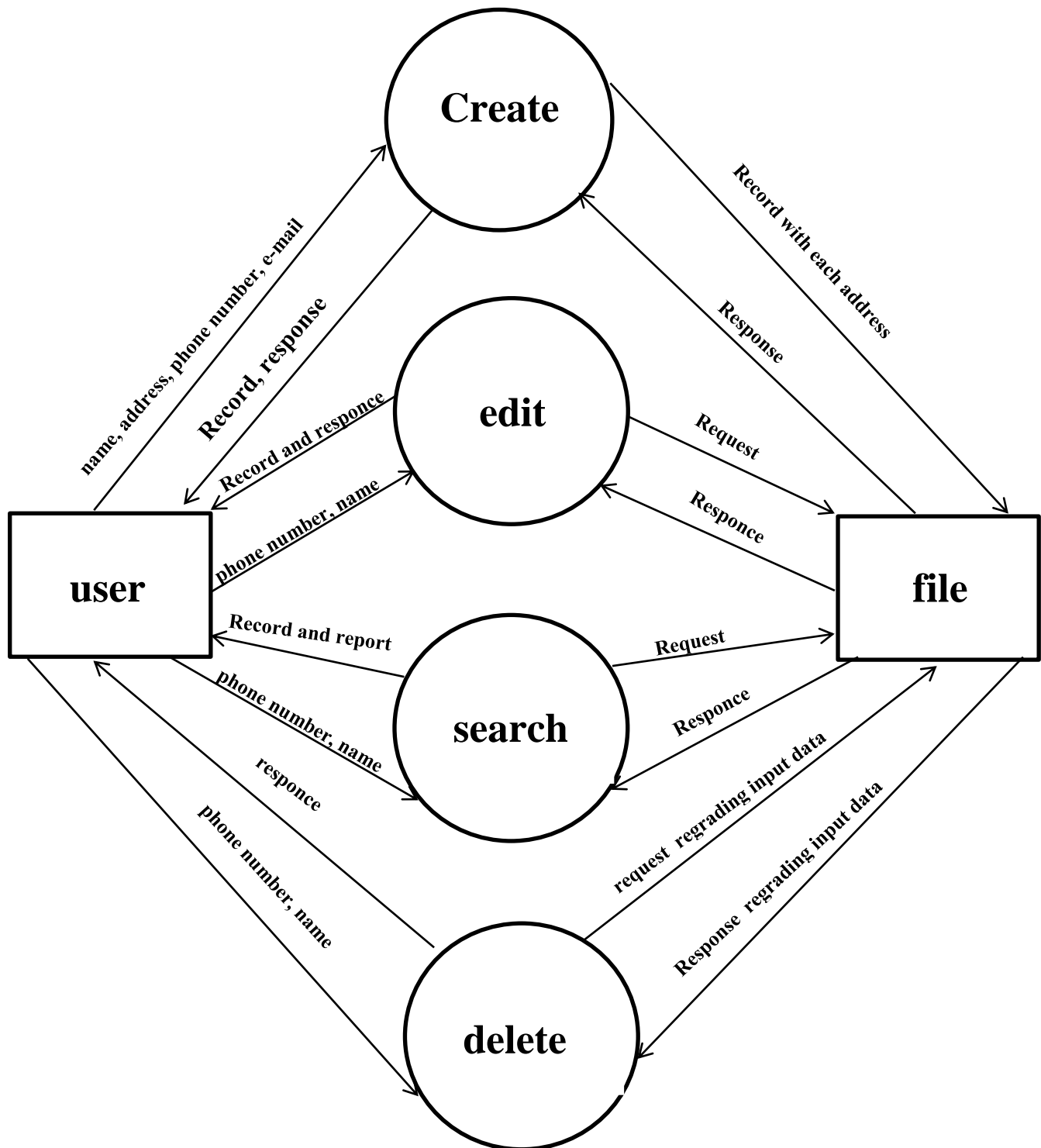
a. FLOW CHART

4.2 DATA FLOW DIAGRAM

LEVEL: 0 DFD



LEVEL: 1 DFD



**SYSTEM IMPLEMENTATION,
TESTING
&
MAINTENANCE**

5.1 SYSTEM IMPLEMENTATION

Places the emphasis on how the data are represented in the database or on how the data structures are implemented to represent what is modelled. Implementation models include the hierarchical database model, the network database model, the relational database model, and the object oriented database model. In this project the module used is the relational database model, because this model provides the following basic Structure:

- The data and relationship are represented by a collection of tables.
- Relational model does not use pointer or links but relates records by the value that contains the value.
- Each table in matrix consisting of a series of row/column intersection. Tables, also called relations, are related to each other by sharing a common entry characteristic.
- Although the tables are completely independent of one another, we can easily connect the data between tables. The relational model thus provides a minimum level of controlled redundancy to eliminate most of the redundancies commonly found in file systems.

5.2 SYSTEM TESTING

Software Testing Fundamentals (STF) is a platform to gain (or refresh) basic knowledge in the field of Software Testing. If we are to ‘cliché’ it, the site is of the testers, by the testers, and for the testers. Our goal is to build a resourceful repository of Quality Content on Quality.

The box approach:

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

5.2.1 WHITE BOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

API testing – testing of the application using public and private APIs (application programming interfaces)

Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)

Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies

5.2.2 BLACK BOX TESTING

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.[26]

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight. "Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

5.2.3 GREY BOX TESTING

Grey-box testing involves having knowledge of internal data structures and algorithms for purposes of designing tests, while executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code.[30][not in citation given] Manipulating input data and formatting output do not qualify as grey-box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

However, tests that require modifying a back-end data repository such as a database or a log file does qualify as grey-box, as the user would not normally be able to change the data repository in normal production operations. Grey-box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

By knowing the underlying concepts of how the software works, the tester makes better-informed testing choices while testing the software from outside. Typically, a grey-box tester will be permitted to set up an isolated testing environment with activities such as seeding a database. The tester can observe the state of the product being tested after performing certain actions such as executing SQL statements against the database and then executing queries to ensure that the expected changes have been reflected. Grey-box testing implements intelligent test scenarios, based on limited information. This will particularly apply to data type handling, exception handling, and so on.

5.3 SYSTEM MAINTENANCE

An integral part of software is the maintenance one, which requires an accurate maintenance plan to be prepared during the software development. It should specify how users will request modifications or report problems. The budget should include resource and cost estimates. A new decision should be addressed for the developing of every new system feature and its quality objectives. The software maintenance, which can last for 5–6 years (or even decades) after the development process, calls for an effective plan which can address the scope of software maintenance, the tailoring of deployment process, the designation of who will provide maintenance, and an estimate of the life-cycle costs. The selection of proper enforcement of standards is the challenging task right from early stage of software engineering which has not got definite importance by the concerned stakeholders.

CONCLUSION

6.1 DRAWBACKS

Every system has some limitations or drawbacks. Some of the limitations are:

- ✓ Lack of duplicate checking.
- ✓ For searching, deleting and modifying a contact you have to provide the full name/number.
- ✓ Lack of server.

6.2 FUTURE SCOPE

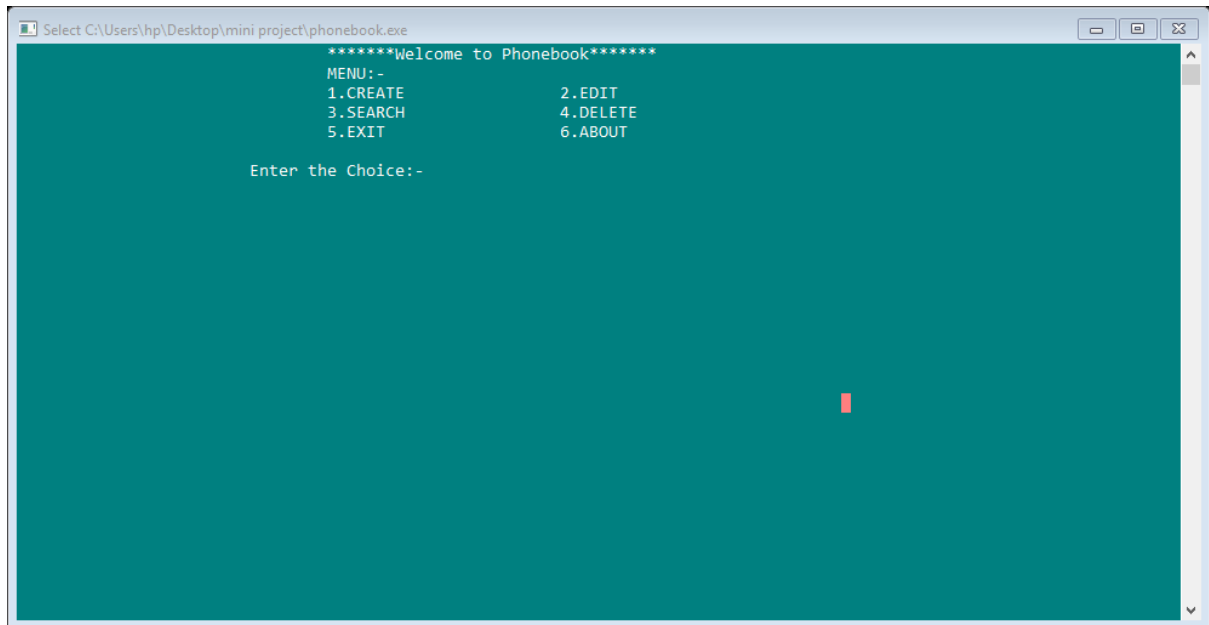
This project was developed to fulfill user requirements: however there is lots of scope to improve the performance of the KBM quiz game.

So there are many things for future enhancement of this project. The future enhancements that are possible in the project are as follows:-

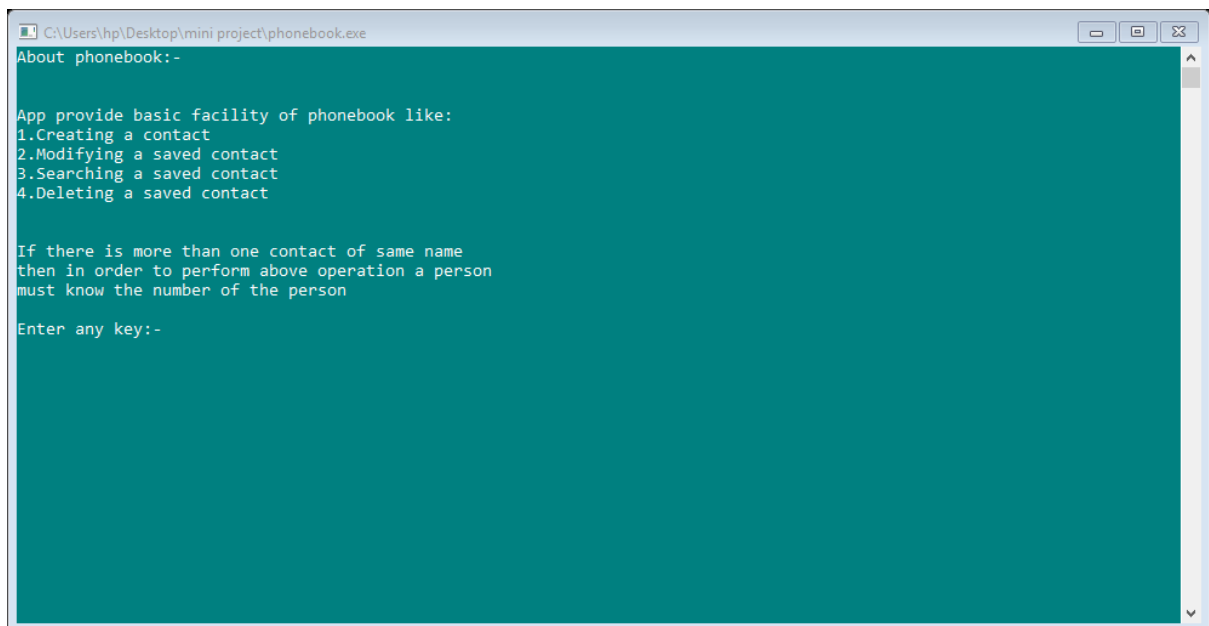
- ❖ We can add a server by which application can save contact for each user in future.
- ❖ We can give more advanced software for online application PHONEBOOK including more facilities.
- ❖ It will be more interactive in future.
- ❖ Integrate multiple load balances to distribute the loads of the system.
- ❖ We can provide feature for searching, modifying, deleting a contact with few character of name/number.

SCREENSHOTS

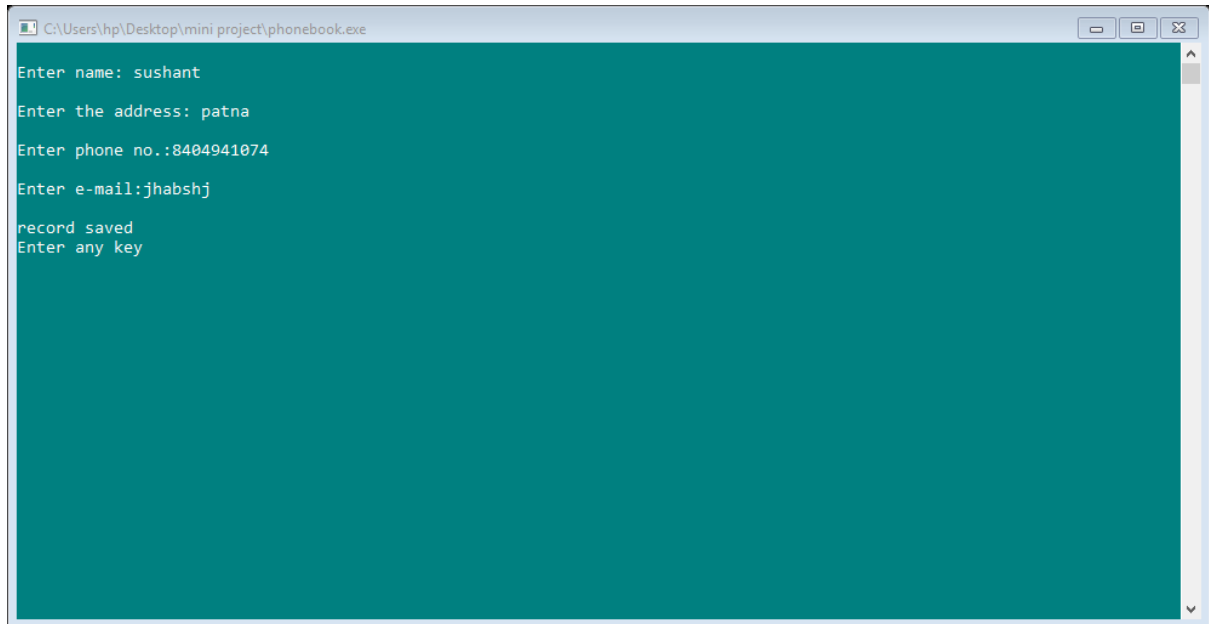
MAIN MENU



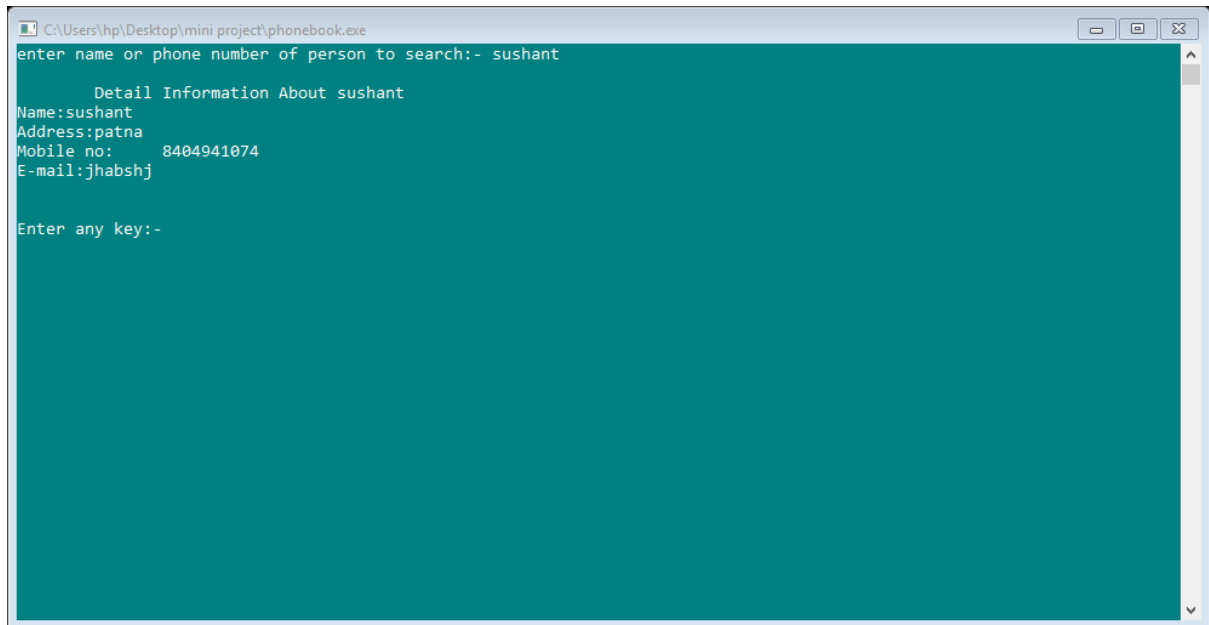
ABOUT PAGE



CREATE



SEARCH

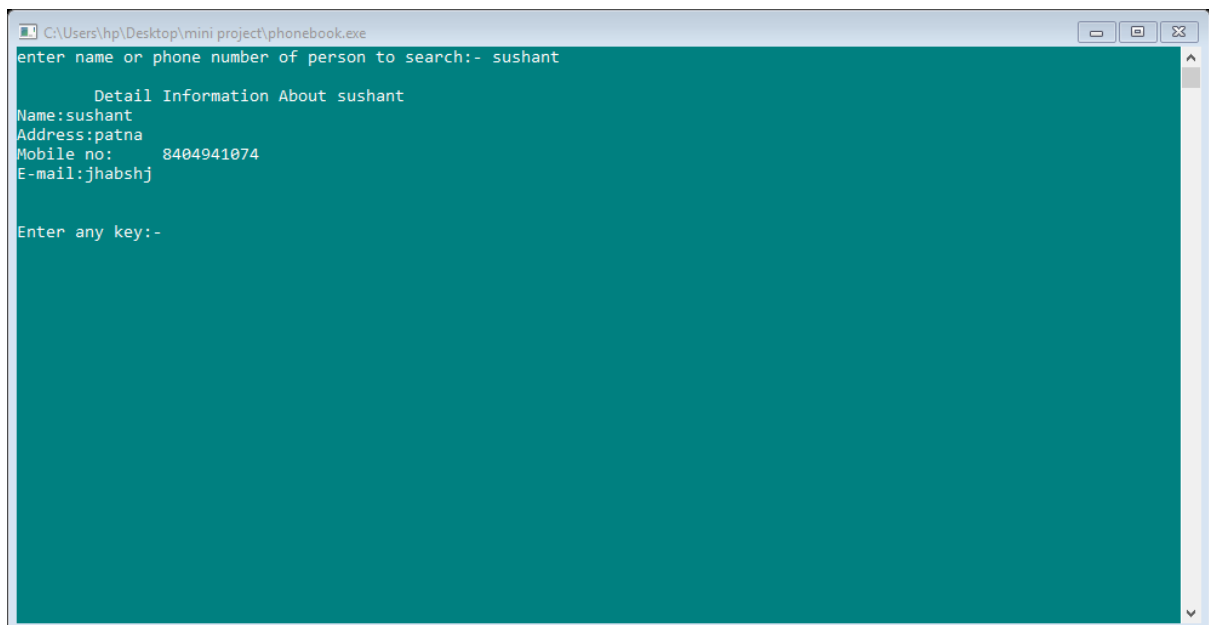


```
C:\Users\hp\Desktop\mini project\phonebook.exe
enter name or phone number of person to search:- sushant

    Detail Information About sushant
Name:sushant
Address:patna
Mobile no:      8404941074
E-mail:jhabshj

Enter any key:-
```

FOUND

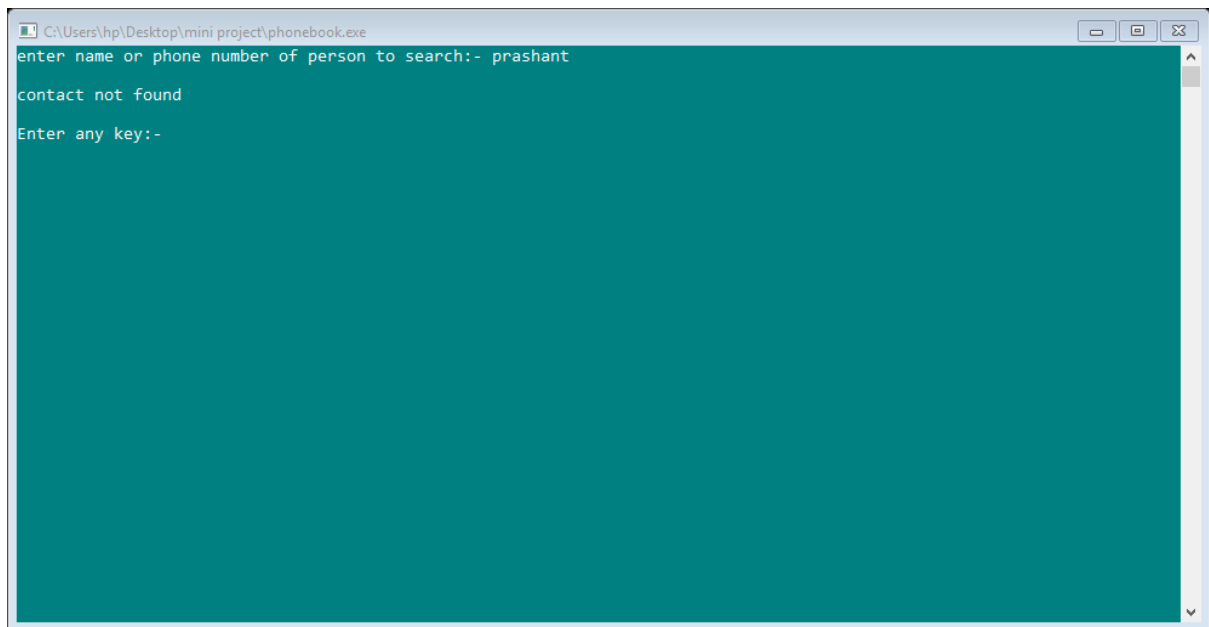


```
C:\Users\hp\Desktop\mini project\phonebook.exe
enter name or phone number of person to search:- sushant

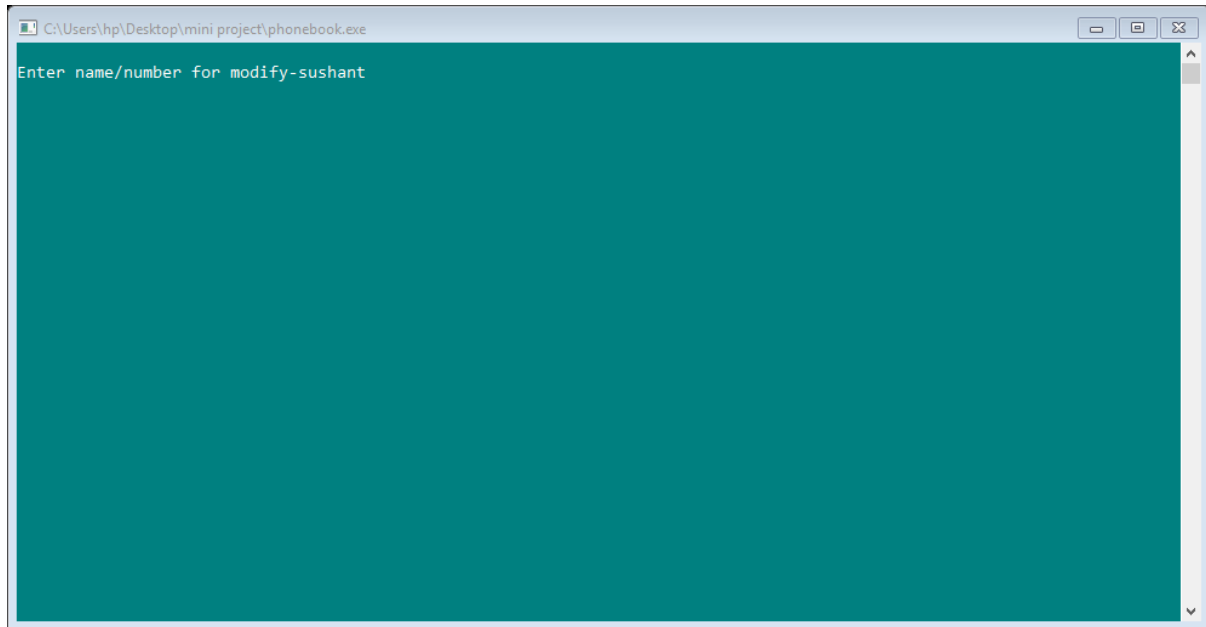
    Detail Information About sushant
Name:sushant
Address:patna
Mobile no:      8404941074
E-mail:jhabshj

Enter any key:-
```

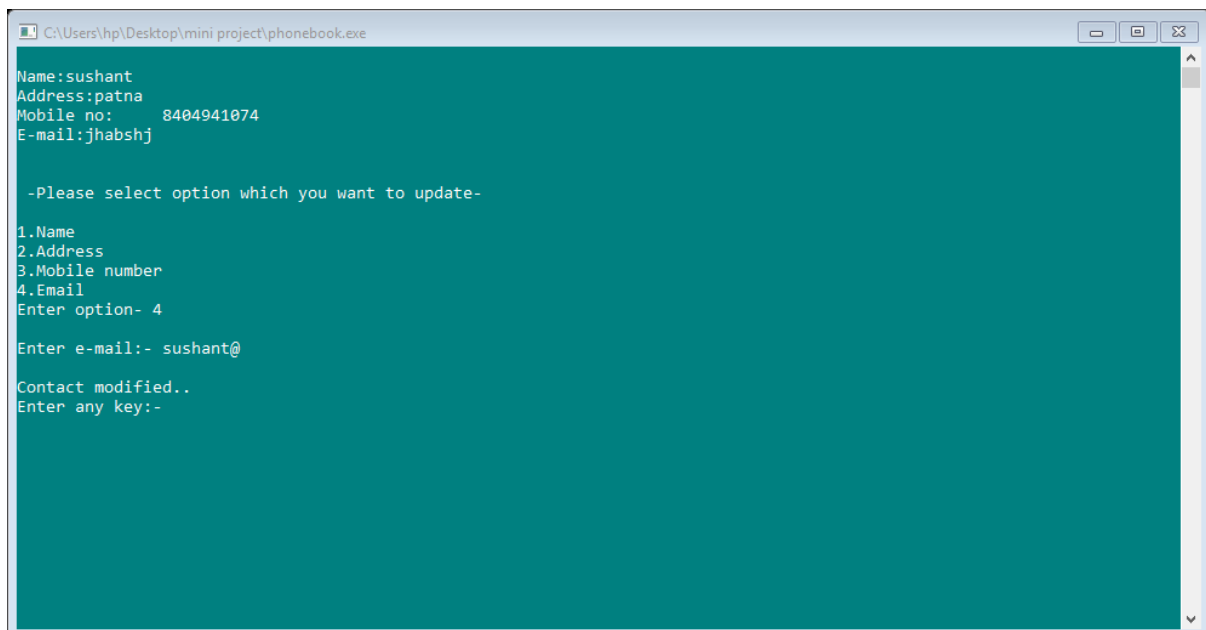

NOT FOUND



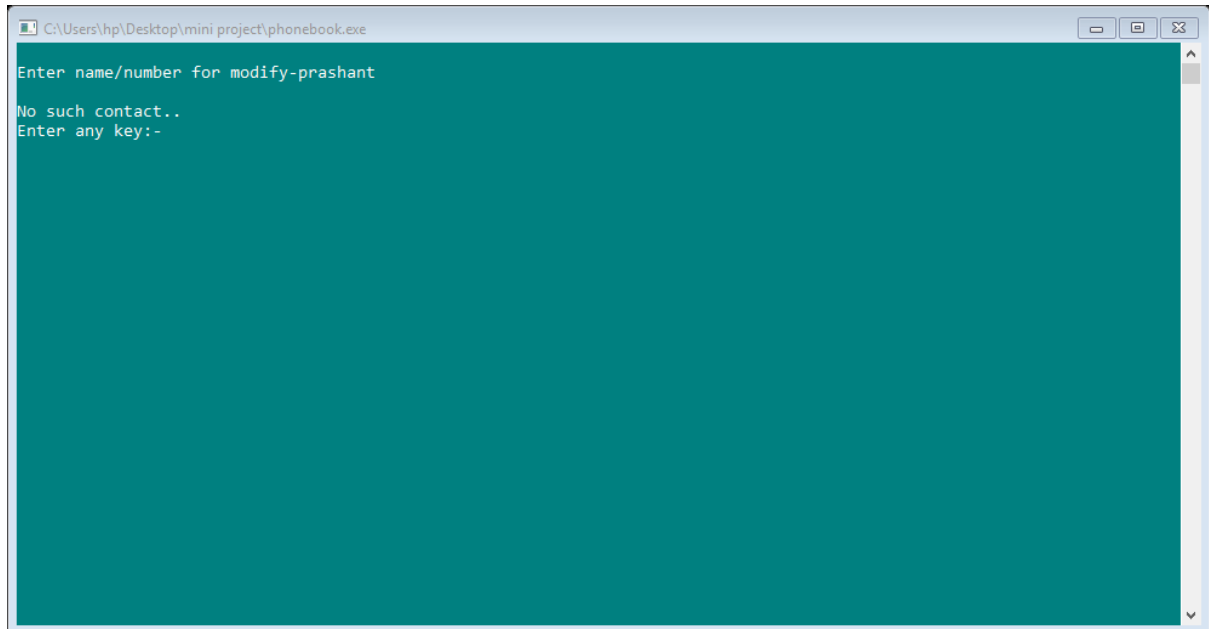
EDIT



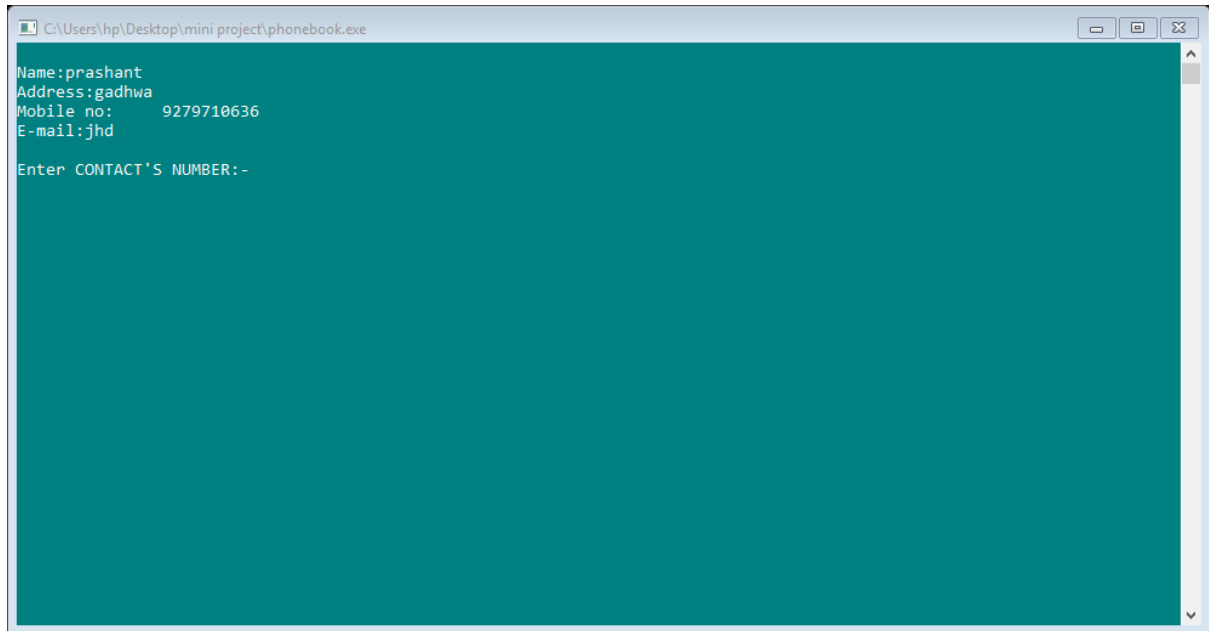
FOUND



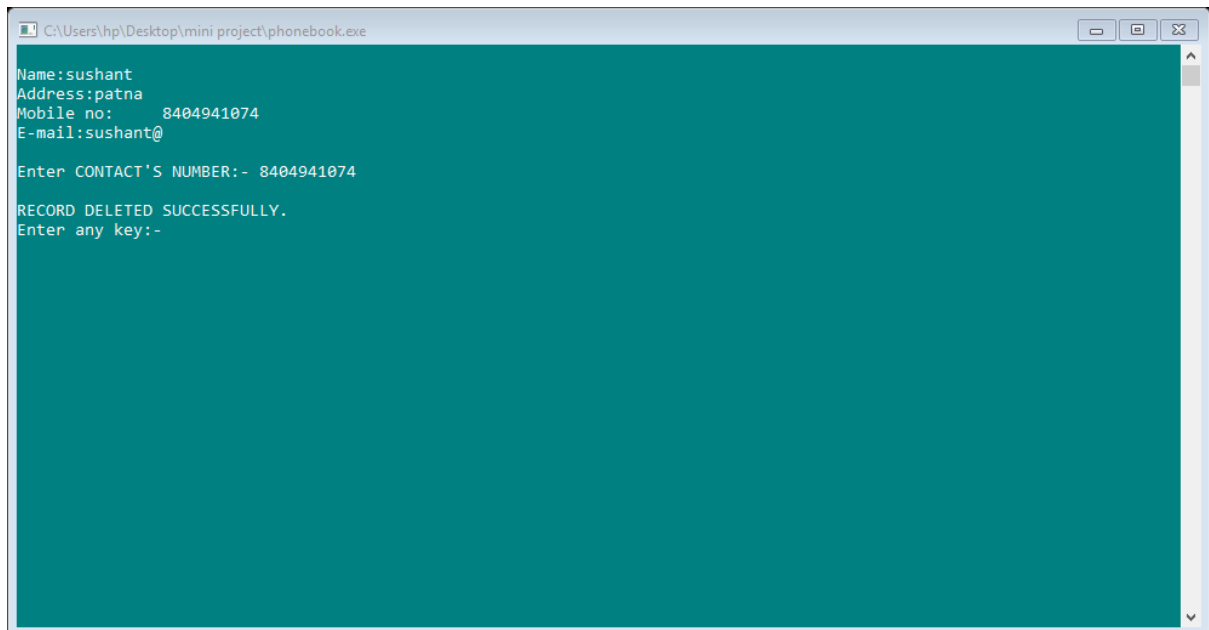
NOT FOUND



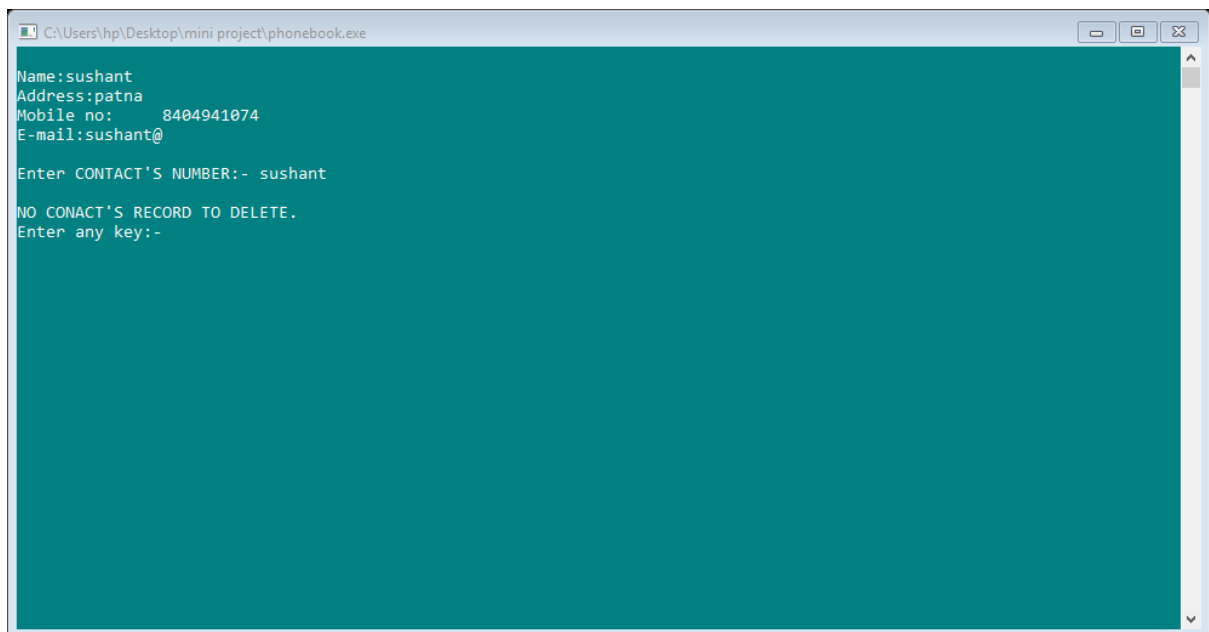
DELETE



FOUND



NOT FOUND



BIBLIOGRAPHY

WEBSITES REFERRED:

- ✓ www.google.com
- ✓ www.wikipedia.com
- ✓ www.computerhope.com

BOOKS REFERRED:

- ✓ **PROGRAMMING IN C by E.BALGURUSAMY**
- ✓ **LET US C by YASHWANT KANETKAR**
- ✓ **PROGRAMMING IN C by PRADIP DEY and
MANAS GHOSH**