PROJECT REPORT

On

# Contact list (using Linkedlist)

Submitted to Centurion University of Technology& Management
in partial fulfillment of the requirement for award of the degree of

B. TECH.
in
COMPUTER SCIENCE & ENGINEERING

Submitted By

| | |
|---|---|
| **Vicky kumar** | **210101120004** |
| **Shivam Kumar** | **210101120009** |
| **Manish Kumar** | **210101120013** |
| **Shubham kumar** | **210101120017** |

Under the Guidance of
Prof. Soumen Chakraborty

DEPT. OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF ENGINEERING &TECHNOLOGY,
CUTM, Paralakhemundi-761200

# CERTIFICATE

This is to be certified that the minor project entitled "**Contact list ( using Linkedlist )**" has been submitted for the Bachelor of Technology in Computer Science Engineering of School of Engineering & Technology, CUTM, Paralakhemundi during the academic year 2021-2022 is a persuasive piece of project work carried out by "**VICKY KUMAR (210101120004), SHUBHAM KUMAR (210101120017), MANISH KUMAR(210101120013), SHIVAM KUMAR(210101120009)**" towards the partial fulfillment for award of the degree (B.Tech.) under the guidance of "]Prof. Soumen Chakraborty " and no part thereof has been submitted by them for any degree to the best of my knowledge.

Signature of HOD
Dr. Debendra Maharana

Signature of Project Guide
Mr. Soumen Chakraborty

# <u>EVALUATION SHEET</u>

1. Title of the Project: Contact list ( using LinkedList )
2. Year of submission: 2023
3. Name of the degree: B-Tech
4. Date of Examination / Viva:
5. Student Name with Reg No. :

    1) Vicky kumar 21010112004
    2) Shubham Kumar 210101120017
    3) Manish kumar 210101120013
    4) Shivam kumar 210101120009

6. Name of the Guide: **Prof. Soumen chakraborty**
7. Result:

[APPROVED/REJECTED]

| | |
|---|---|
| Signature of HOD | Signature of Project Guide |
| Dr. Debendra Maharana | **Mr. Soumen Chakraborty** |

Signature of External Examiner

# CANDIDATE'S DECLARATION

I "**VICKY KUMAR(210101120004), SHUBHAM KUMAR (210101120017), MANISH KUMAR(210101120013), SHIVAM KUMAR(210101120009)**", B-Tech in CSE (Semester- IV) of School of Engineering &Technology, CUTM, Paralakhemundi, hereby declare that the Project Report entitled "Contact list (using LinkedList)" is an original work and data provided in the study is authentic one. This report has not been submitted to any other Institute for the award of any other degree by me.

Signature of Student

## ACKNOWLEDGEMENT

I would like to thank my teacher Prof. Soumen chakraborty who gave me this opportunity to work on this project. I got to learn a lot from this project about Contact list ( using Linkedlist ).

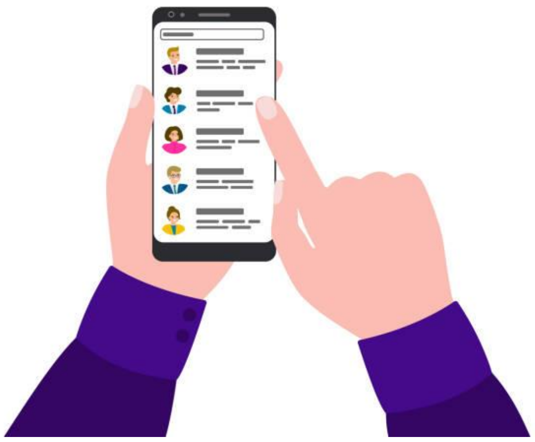
I would also like to thank our Branch H.O.D Debendra Maharana

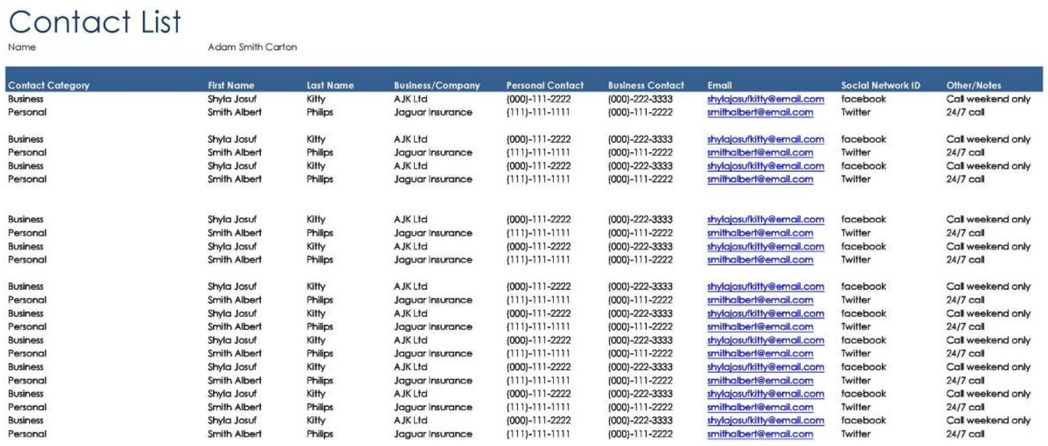
At last, I would like to extend my heartfelt thanks to my parents because without their help this project would not have been successful. Finally, I would like to thank my dear friends who have been with me all the time.

STUDENT NAMES & REG No.

| | |
|---|---|
| Vicky kumar | 210101120004 |
| Shubham Kumar | 210101120017 |
| Manish Kumar | 210101120013 |
| Shivam Kumar | 210101120009 |

# Contact list ( using linkedlist)





Contact List

Name Adam Smith Carton

| Contact Category | First Name | Last Name | Business/Company | Personal Contact | Business Contact | Email | Social Network ID | Other/Notes |
|---|---|---|---|---|---|---|---|---|
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |
| Business | Shyla Josuf | Kitty | AJK Ltd | (000)-111-2222 | (000)-222-3333 | shylajosufkitty@email.com | facebook | Call weekend only |
| Personal | Smith Albert | Philips | Jaguar Insurance | (111)-111-1111 | (000)-111-2222 | smithalbert@email.com | Twitter | 24/7 call |

INDEX:

## Introduction: -

Authentication A contact list is a collection of screen names. It is a commonplace feature of instant messaging, Email clients, online games and mobile phones. It has various trademarked and proprietary names in different contexts.

Contacts lists' windows show screen names that represent actual other people. To communicate with someone on the list, the user can select a name and act upon it, for example open a new E-mail editing session, instant message, or telephone call. In some programs, if your contact list shows someone, their list will show yours. Contact lists for mobile operating systems are often shared among several mobile apps.

Some text message clients allow you to change your display name at will while others only allow you to reformat your screen name (Add/remove spaces and capitalize letters). Generally, it makes no difference other than how it's displayed.

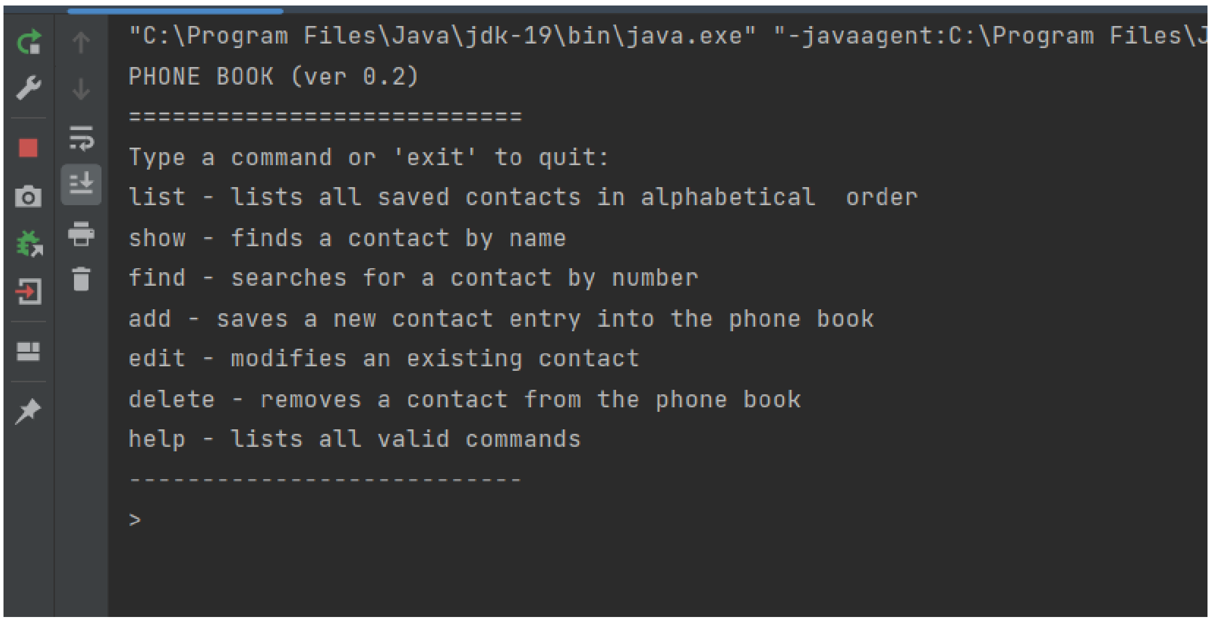
With most programs, the contact list can be minimized to keep it from getting in the way, and is accessed again by selecting its icon.

A contact list can be used for forming social networks, distribution lists and contact networks. Information about users is readily available in existing contact lists, and can be tailored to suit different needs. For organizations or sales, a contact list adds to the capacity and a strong resource for communication. Having a list of people with similar interests could help in responding to different needs of the group. From a psychological perspective, users often find a great sense of achievement and satisfaction in building a contact list. It can help in building long-term relationships as well. From a sales perspective, having a contact list lessens the need to directly “sell the group.” Rather, they can concentrate on sending the message, as they already know the people and what they do.

## Proposed methodology/Algorithm: -

When user run this program, he gets an output interface (fig3) in which have 3 options users have to select any one within this these are:

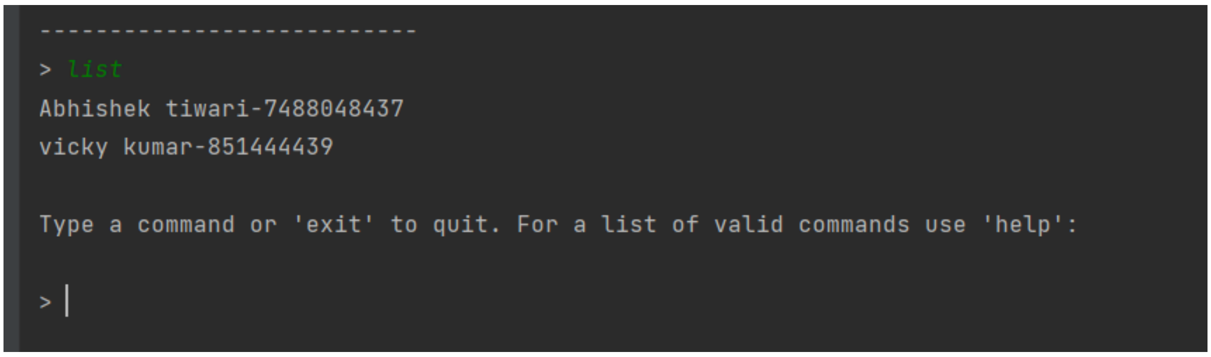
1.showing the options



```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\J
PHONE BOOK (ver 0.2)
===========================
Type a command or 'exit' to quit:
list - lists all saved contacts in alphabetical  order
show - finds a contact by name
find - searches for a contact by number
add - saves a new contact entry into the phone book
edit - modifies an existing contact
delete - removes a contact from the phone book
help - lists all valid commands
---------------------------
>
```

If choice 1: -

Showing the list of contact



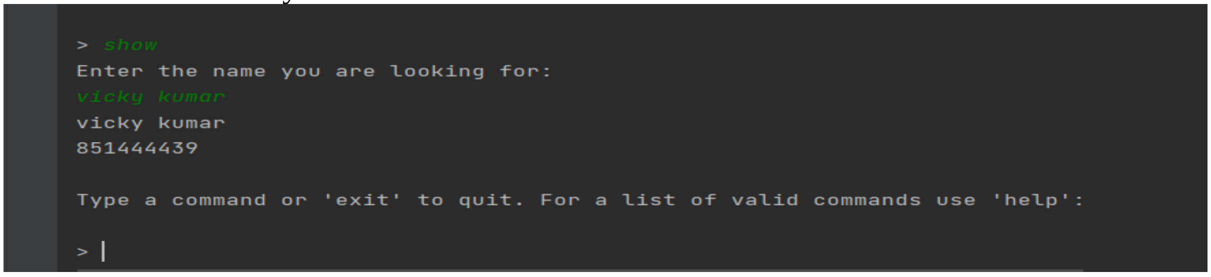
```
---------------------------
> list
Abhishek tiwari-7488048437
vicky kumar-851444439

Type a command or 'exit' to quit. For a list of valid commands use 'help':

> |
```

If choice 2: -

Find contact by name



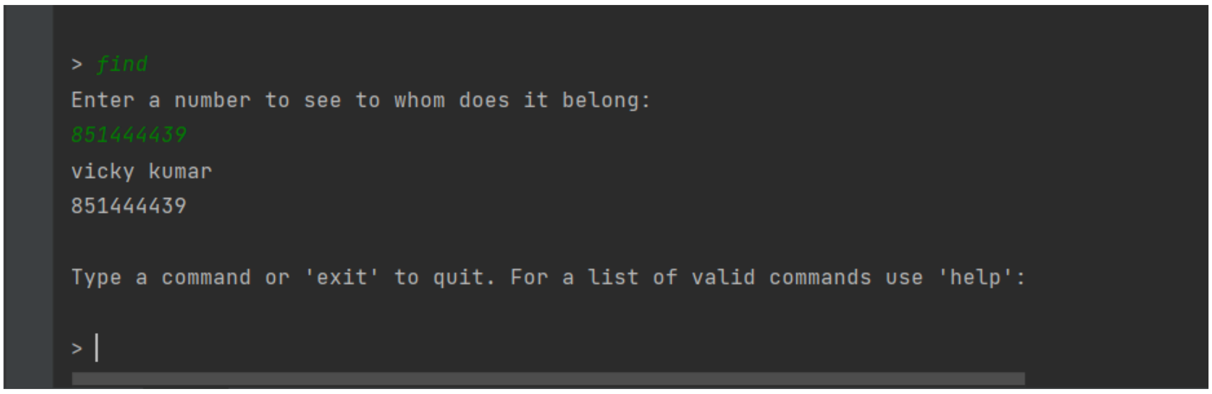
```
> show
Enter the name you are looking for:
vicky kumar
vicky kumar
851444439

Type a command or 'exit' to quit. For a list of valid commands use 'help':

> |
```

If choice 3: -

i. Search contact by numbe



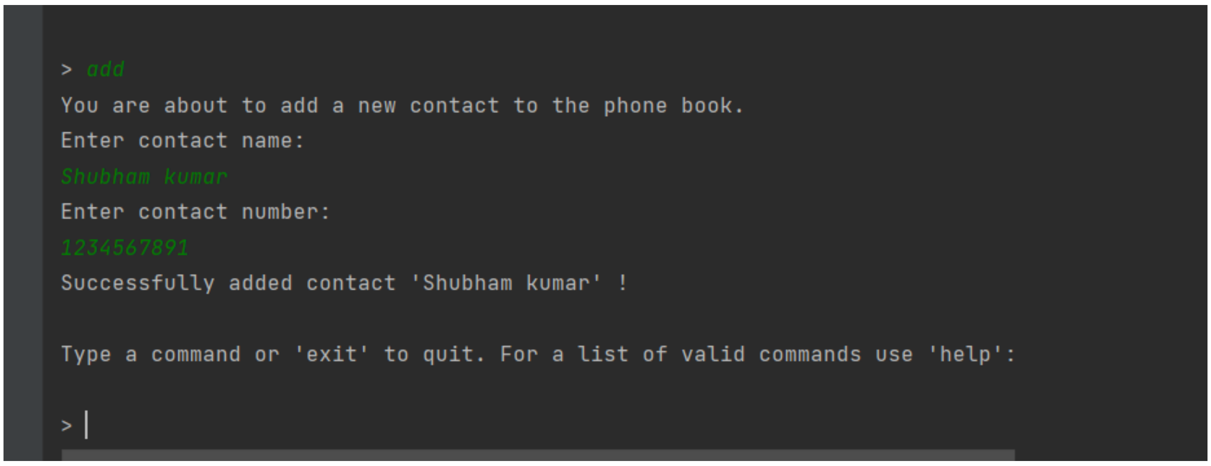
```
> find
Enter a number to see to whom does it belong:
851444439
vicky kumar
851444439

Type a command or 'exit' to quit. For a list of valid commands use 'help':

> |
```

If choice 4:-

Add new contact



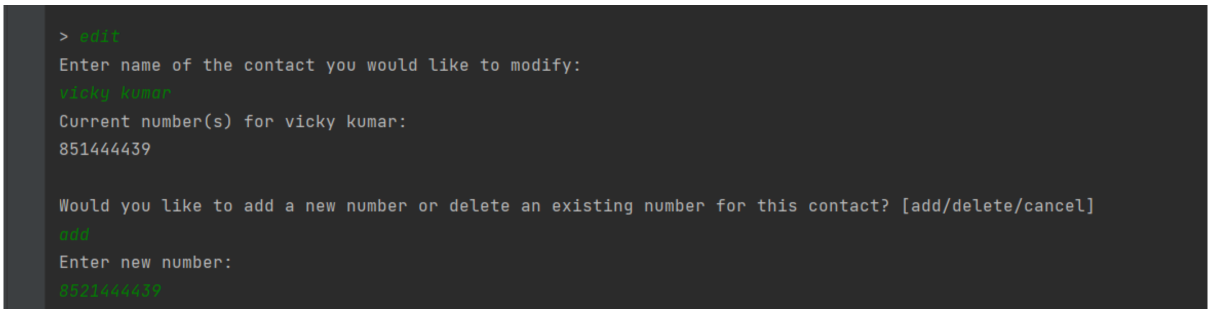
```
> add
You are about to add a new contact to the phone book.
Enter contact name:
Shubham kumar
Enter contact number:
1234567891
Successfully added contact 'Shubham kumar' !

Type a command or 'exit' to quit. For a list of valid commands use 'help':

> |
```

If choice 5:-

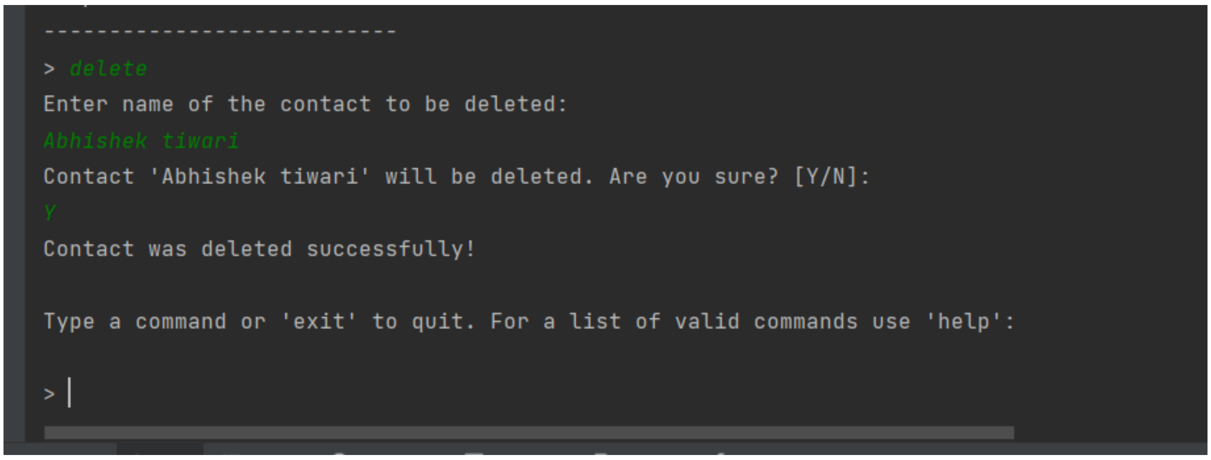
Edit – add delete remove numver


```
> edit
Enter name of the contact you would like to modify:
vicky kumar
Current number(s) for vicky kumar:
851444439

Would you like to add a new number or delete an existing number for this contact? [add/delete/cancel]
add
Enter new number:
8521444439
```

If choice 6:-

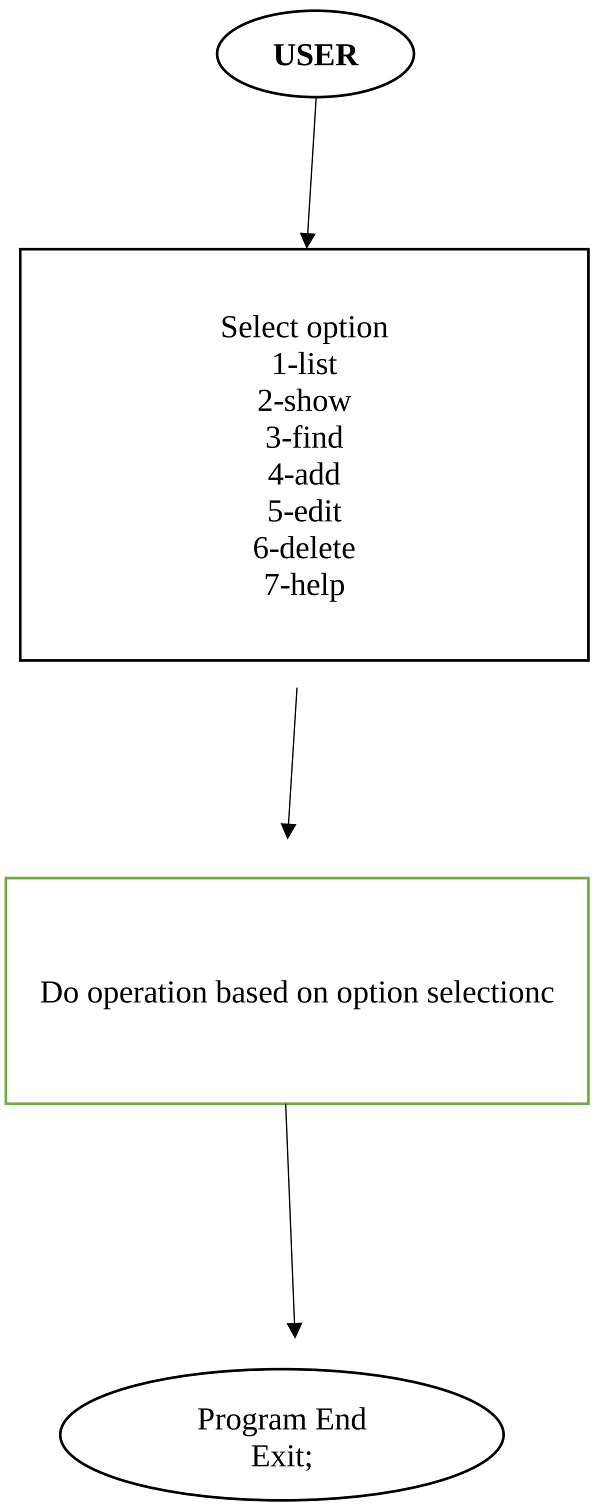
Delete- delete a contact from list


```
---------------------------
> delete
Enter name of the contact to be deleted:
Abhishek tiwari
Contact 'Abhishek tiwari' will be deleted. Are you sure? [Y/N]:
Y
Contact was deleted successfully!

Type a command or 'exit' to quit. For a list of valid commands use 'help':

> 
```

## Flow Chart: -


USER
Select option
1-list
2-show
3-find
4-add
5-edit
6-delete
7-help
Do operation based on option selectionc
Program End
Exit;

## Method and class used in this program: -

Here some Methods that are used to make this program
User defined function

i. takepassword (): - it is a user define function that take the password safely and store in give variable address.
ii. Savecontacts():-for saving the contact in contacts.csv file
iii. Listcommand;- for printing the list of operation
iv. Loadcontact:- for loading contact from csv file
v. Listcontact:- for sowing list of contact
vi. Showcontact: - search and show contact by name
vii. Findcontact : search and show contact by number
viii. Addcontact:- add a new contact in phonebook
ix. Editcontact:- add or delete number of any contact
x. Deletecontact:- delete any contact

Standard methos and class: -

1) main()-from here the code start execute
2) System.out.println() – to print the data to screen and in file
3) Sccanner- this is a class for taking user input
4) while() -this looping some block of codes
5) if else() – there are some given condition if condition are true inside the if block will be execute otherwise else block will be execute.
6) switch(case)-it is a case check statement it run respective case labels.
7) All methods of linkedlist class

## Implementation Code: -

```
import java.util.*;
import java.io.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PhoneBook {

    private static final String DATA_PATH = "src/contacts.csv";

    private static void saveContacts(Map<String, List<String>> contacts) {
        try (PrintWriter writer = new PrintWriter(DATA_PATH)) {
            if (!contacts.isEmpty()) {
                for (Map.Entry<String, List<String>> entry : contacts.entrySet()) {
                    String line = String.format("%s,\"%s\"",
                            entry.getKey(), entry.getValue().toString().replaceAll("\\[|]", ""));
                    writer.println(line);
                }
            }

        } catch (IOException ioex) {
            System.err.println(ioex.getMessage());
        }
    }

    private static void loadContacts(Map<String, List<String>> contacts) {
        try (BufferedReader reader = new BufferedReader(new FileReader(DATA_PATH))) {

            Pattern pattern = Pattern.compile("^([^,\"]{2,50}),\"([0-9+, ]+)\"$");

            while (true) {
                String line = reader.readLine();
                if (line == null) {
                    break;
                }

                Matcher matcher = pattern.matcher(line);
                if (matcher.find()) {
                    String[] numbers = matcher.group(2).split(",\\s*");
                    contacts.put(matcher.group(1), Arrays.asList(numbers));
                }
            }

        } catch (IOException ioex) {
            System.err.println("Could not load contacts, phone book is empty!");
        }
    }

    private static void listCommands() {
        System.out.println("list - lists all saved contacts in alphabetical  order");
        System.out.println("show - finds a contact by name");
        System.out.println("find - searches for a contact by number");
        System.out.println("add - saves a new contact entry into the phone book");
        System.out.println("edit - modifies an existing contact");
        System.out.println("delete - removes a contact from the phone book");
        System.out.println("help - lists all valid commands");
        System.out.println("---------------------------");
    }

    private static void listContacts(Map<String, List<String>> contacts) {
        if (!contacts.isEmpty()) {
            for (Map.Entry<String, List<String>> entry : contacts.entrySet()) {
                System.out.print(entry.getKey());
                for (String number : entry.getValue()) {
                    System.out.print("-"+number);
                }
                System.out.println();
            }
        } else {
            System.out.println("No records found, the phone book is empty!");
        }

        System.out.println();
        System.out.println("Type a command or 'exit' to quit. For a list of valid commands use
'help':");
    }

    private static void showContact(Map<String, List<String>> contacts, Scanner input) {
        System.out.println("Enter the name you are looking for:");
        String name = input.nextLine().trim();

        if (contacts.containsKey(name)) {
            System.out.println(name);
            for (String number : contacts.get(name)) {
                System.out.println(number);
            }
        } else {
            System.out.println("Sorry, nothing found!");
        }

        System.out.println();
        System.out.println("Type a command or 'exit' to quit. For a list of valid commands use
'help':");
    }

    private static void findContact(Map<String, List<String>> contacts, Scanner input) {
        System.out.println("Enter a number to see to whom does it belong:");
        String number = input.nextLine().trim();
        while (!number.matches("^\\+?[0-9 ]{3,25}$")) {
            System.out.println("Invalid number! May contain only digits, spaces and '+'. Min
length 3, max length 25.");
            System.out.println("Enter number:");
            number = input.nextLine().trim();
        }

        for (Map.Entry<String, List<String>> entry : contacts.entrySet()) {
            if (entry.getValue().contains(number)) {
                System.out.println(entry.getKey());
                System.out.println(number);
            }
        }

        System.out.println();
        System.out.println("Type a command or 'exit' to quit. For a list of valid commands use
'help':");
    }

    private static void addContact(Map<String, List<String>> contacts, Scanner input) {
        System.out.println("You are about to add a new contact to the phone book.");
        String name;
        String number;

        while (true) {
            System.out.println("Enter contact name:");
            name = input.nextLine().trim();
            if (name.matches("^.{2,50}$")) {
                break;
            } else {
                System.out.println("Name must be in range 2 - 50 symbols.");
            }
        }

        while (true) {
            System.out.println("Enter contact number:");
            number = input.nextLine().trim();
            if (number.matches("^\\+?[0-9 ]{3,25}$")) {
                break;
            } else {
                System.out.println("Number may contain only '+', spaces and digits. Min length 3,
max length 25.");
            }
        }

        if (contacts.containsKey(name)) {
            System.out.printf("'%s' already exists in the phone book!\n", name);

            if (contacts.get(name).contains(number)) {
                System.out.printf("Number %s already available for contact '%s'.\n", number,
name);
            } else {
                contacts.get(name).add(number);
                saveContacts(contacts);
                System.out.printf("Successfully added number %s for contact '%s'.\n", number,
name);
            }

        } else {
            List<String> numbers = new ArrayList<>();
            numbers.add(number);
            contacts.put(name, numbers);
            saveContacts(contacts);
            System.out.printf("Successfully added contact '%s' !\n", name);
        }

        System.out.println();
        System.out.println("Type a command or 'exit' to quit. For a list of valid commands use
'help':");
    }

    private static void editContact(Map<String, List<String>> contacts, Scanner input) {
        System.out.println("Enter name of the contact you would like to modify:");
        String name = input.nextLine().trim();

        if (contacts.containsKey(name)) {
            List<String> numbers = new ArrayList<>(contacts.get(name));
            System.out.printf("Current number(s) for %s:\n", name);
            for (String number : numbers) {
                System.out.println(number);
            }
            System.out.println();
            System.out.println("Would you like to add a new number or delete an existing
number for this contact? [add/delete/cancel]");
            String editOption = input.nextLine().trim().toLowerCase();
            boolean addNumber = false;
            boolean delNumber = false;

            option:
            while (true) {
                switch (editOption) {
                    case "add":
                        addNumber = true;
                        break option;
                    case "delete":
                        delNumber = true;
                        break option;
                    case "cancel":
                        System.out.println("Contact was not modified!");
                        break option;
                    default:
                        System.out.println("Use 'add' to save a new number, 'delete' to remove an
existing number or 'cancel' to go back.");
                        editOption = input.nextLine().trim().toLowerCase();
                        break;
                }
            }

            if (addNumber) {
                while (true) {
                    System.out.println("Enter new number:");
                    String number = input.nextLine().trim();
                    if (number.matches("^\\+?[0-9 ]{3,25}$")) {
                        contacts.get(name).add(number);
                        saveContacts(contacts);
                        System.out.printf("Number %s was successfully added, record updated!\n",
number);
                        break;
                    } else {
                        System.out.println("Number may contain only '+', spaces and digits. Min
length 3, max length 25.");
                    }
                }
            }

            if (delNumber) {
                while (true) {
                    System.out.println("Enter the number you want to delete:");
                    String number = input.nextLine().trim();
                    if (numbers.contains(number)) {
                        numbers.remove(number);
                        contacts.put(name, numbers);
                        saveContacts(contacts);
                        System.out.printf("Number %s was removed from the record for '%s'\n",
number, name);
                        break;
                    } else {
                        System.out.printf("Number does not exist! Current number(s) for %s:\n",
name);
                        for (String num : numbers) {
                            System.out.println(num);
                        }
                    }
                }
            }

        } else {
            System.out.println("Sorry, name not found!");
        }
        System.out.println();
        System.out.println("Type a command or 'exit' to quit. For a list of valid commands use
'help':");
    }

    private static void deleteContact(Map<String, List<String>> contacts, Scanner input) {
        System.out.println("Enter name of the contact to be deleted:");
        String name = input.nextLine().trim();

        if (contacts.containsKey(name)) {
            System.out.printf("Contact '%s' will be deleted. Are you sure? [Y/N]:\n", name);
            String confirmation = input.nextLine().trim().toLowerCase();
            confirm:
            while (true) {
                switch (confirmation) {
                    case "y":
                        contacts.remove(name);
                        saveContacts(contacts);
                        System.out.println("Contact was deleted successfully!");
                        break confirm;
                    case "n":
                        break confirm;
                    default:
                        System.out.println("Delete contact? [Y/N]:");
                        break;
                }
                confirmation = input.nextLine().trim().toLowerCase();
            }

        } else {
            System.out.println("Sorry, name not found!");
        }

        System.out.println();
        System.out.println("Type a command or 'exit' to quit. For a list of valid commands use
'help':");
    }

    public static void main(String[] args) {

        System.out.println("PHONE BOOK (ver 0.2)");
        System.out.println("==========================");
        System.out.println("Type a command or 'exit' to quit:");
        listCommands();
        System.out.print("> ");

        Map<String, List<String>> contacts = new TreeMap<>();
        loadContacts(contacts);

        Scanner input = new Scanner(System.in);
        String line = input.nextLine().trim();

        while (!line.equals("exit")) {

            switch (line) {
                case "list":
                    listContacts(contacts);
                    break;
                case "show":
                    showContact(contacts, input);
                    break;
                case "find":
                    findContact(contacts, input);
                    break;
                case "add":
                    addContact(contacts, input);
                    break;
                case "edit":
                    editContact(contacts, input);
                    break;
                case "delete":
                    deleteContact(contacts, input);
                    break;
                case "help":
                    listCommands();
                    break;
                default:
                    System.out.println("Invalid command!");
                    break;
            }


            System.out.print("\n> ");
            line = input.nextLine().trim();
        }

        System.out.println("'Phone Book 0.2' terminated.");
    }
}
```

**Implementation issues and challenges: -**

The major challenge of the project is the shortage of time because time is needed to learning the code and need to implement the system. The time also uses to find out the better solution from the others proposed solution. The proposed solution needed to study and find out their strength and weaknesses to be improved so that can learn from their problem. The next challenge of the project will be the limitation of understanding the code used to implement the system. The coding will be a very fresh programming language for me since it is very new for me.

## Conclusion: -

With the rapid evolution of the wireless communication technology, user authentication is important in order to ensure the security of the wireless communication technology. Password play an important role in the process of authentication. In the process of authentication, the password enter by the user will be transmitted along the traffic to the authentication server in order to allow the server to grant access to the authorized user. Due to the issues, there are many solutions has been proposed to improve the security of wireless communication technology.

## References: -

1) http://eprints.utar.edu.my/2855/1/CS-2018-1405547-1.pdf
2) https://www.freeprojectz.com/java-project/contact list
3) https://github.com/topics/contact-list
4) https://www.w3schools.com/java/
5) https://www.javatpoint.com/java-tutorial
6) https://www.tutorialspoint.com/java/index.htm
7) https://docs.oracle.com/javase/tutorial/