NAMIBIA UNIVERSITY
OF SCIENCE AND TECHNOLOGY
Faculty of Computing and Informatics
School of Computing
Department of Software Engineering

13 Jackson Kaujeua Street
Private Bag 13388
Windhoek
NAMIBIA

T: +264 61 207 2052
F: +264 61 207 9052
E: dse@nust.na
W: www.nust.na

MODE OF STUDY : FULLTIME

Group Members	
Student Number	**Full Name**
224026569	NAO NAKANGOMBE
224007629	EMELDA MUKONI
224016172	ATUHE KAMBONDE
224087479	BEN PHIRI
224065335	BERNARD FOTOLELA
223103756	SIMON NTINDA

Project : Phonebook Application

1.0. DESCRIPTION OVERVIEW

This project aims to create an efficient and user-friendly phone book application tailored for a Namibian telecommunications company. By utilizing fundamental linear data structures, the application will provide essential contact management functionalities, including insertion, searching, deletion, updating, and display of contacts. Designed with optimal performance in mind, this application prioritizes ease of use, ensuring that users can manage their contacts intuitively and effectively. Through this initiative, we seek to enhance the overall customer experience and streamline communication within the telecommunications sector.

2.0. KEY FUNCTIONALITIES

- Insert Contact: Add a new contact to the phonebook.
- Search Contact: Find a contact by name or number.
- Display All Contacts: Show all contacts in the phonebook.
- Delete Contact: Remove a contact from the phonebook.
- Update Contact Modify details of an existing contact.
- Sort Contacts: Organize contacts for faster searches.

Proposed Modules and Functions

Modules Design

1. Contact Module: Manage individual contact data.
2. Phonebook Module: Handle phonebook operations (insert, search, delete, update).
3. Display Module: Responsible for displaying contacts.
4. Sort Module: Sort contacts for efficient searching.
5. Analysis Module: Evaluate the performance of the search algorithm.

Functions Implementation

1. **Contact Module Functions:**

create_contact(name, number)`: Returns a new contact object.

2. **Phonebook Module Functions:**

insert_contact(contact)`: Adds a contact to the phonebook.

search_contact(query)`: Searches for a contact by name or number.

delete_contact(name)`: Removes a contact from the phonebook.

update_contact(name, new_info)`: Updates contact details.

3. **Display Module Functions:**

display_all_contacts()`: Prints all contacts.

4. **Sort Module Functions:**

sort_contacts()`: Sorts contacts alphabetically by name.

5. **Analysis Module Functions:**

analyze_search_efficiency()`: Evaluates search efficiency.

Pseudocode Design

1. **Insert Contact**

```
Start
  Prompt the user to enter contact details (name, number)
  Get the contact details (name, number)
  Add the contact to the phonebook list (array or linked list)
  Display success message ("Contact added")
```

```
End
```

2. Search contact

```
Start
  Prompt the user to enter the contact name or number to search
  Get the search query
  Loop through the phonebook list:
    If a contact matches the search query:
      Display the contact details
      End search
  If no match is found:
    Display "Contact not found"
End
```

3. Display all contacts

```
Start
  If the phonebook list is empty:
    Display "No contacts available"
  Else:
    Loop through the phonebook list:
      Display each contact’s name and number
End
```

4. Delete Contact

```
Start
  Prompt the user to enter the name of the contact to delete
  Get the contact name
  Loop through the phonebook list:
    If a contact matches the name:
      Delete the contact
      Display "Contact deleted"
```

```
          End
    If no match is found:
        Display "Contact not found"
End
```

5. Update Contact

```
Start
    Prompt the user to enter the name of the contact to update
    Get the contact name
    Loop through the phonebook list:
        If a contact matches the name:
            Prompt the user to enter the new contact information (name, number)
            Update the contact's details
            Display "Contact updated"
            End
    If no match is found:
        Display "Contact not found"
End
```

6. Sort Contacts

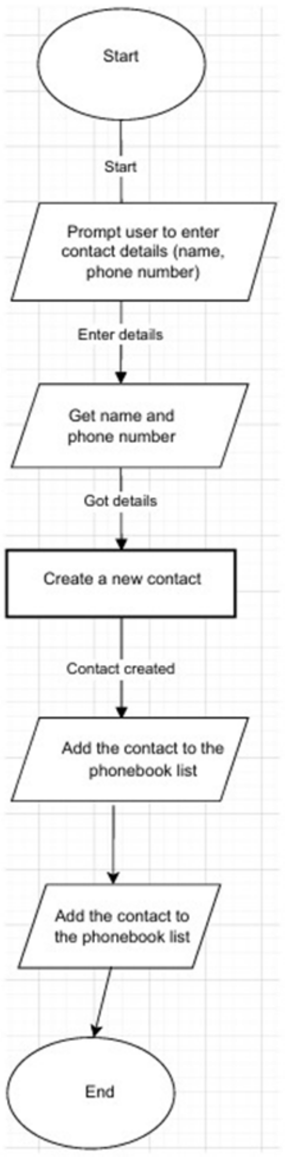
```
Start
    If the phonebook list is empty:
      Display "No contacts to sort"
    Else:
      Sort the phonebook list alphabetically by name
      Display "Contacts sorted"
End
```

7. Analyze Search Efficiency

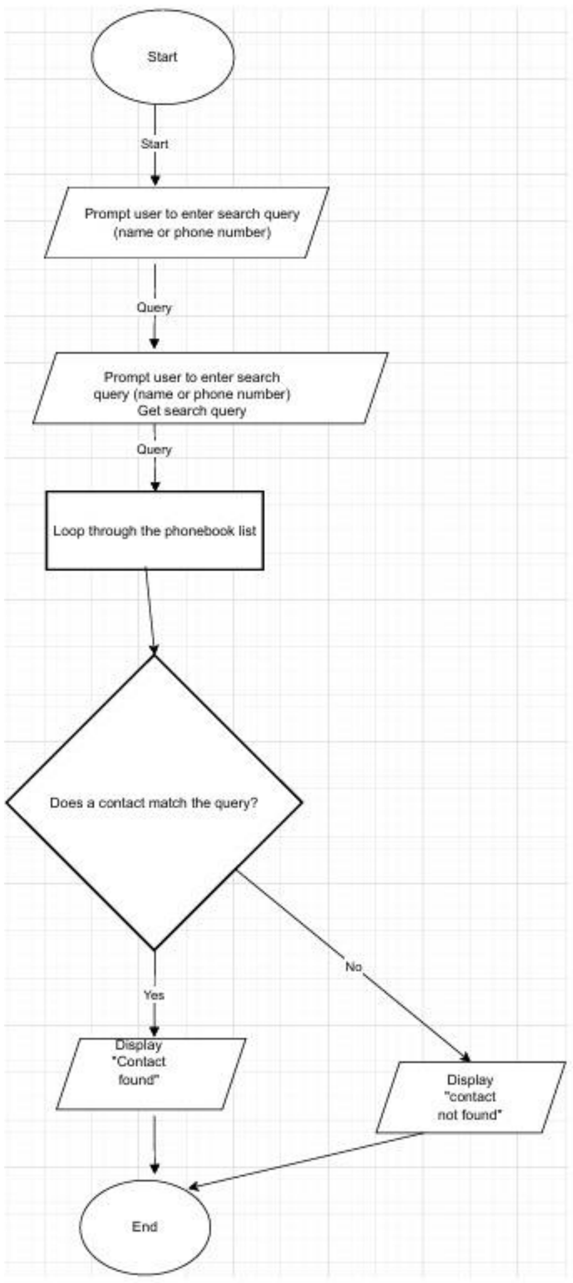
```
Start
    Display the time complexity of the search algorithm (O(n) for linear search)
End
```

Flowchart Design

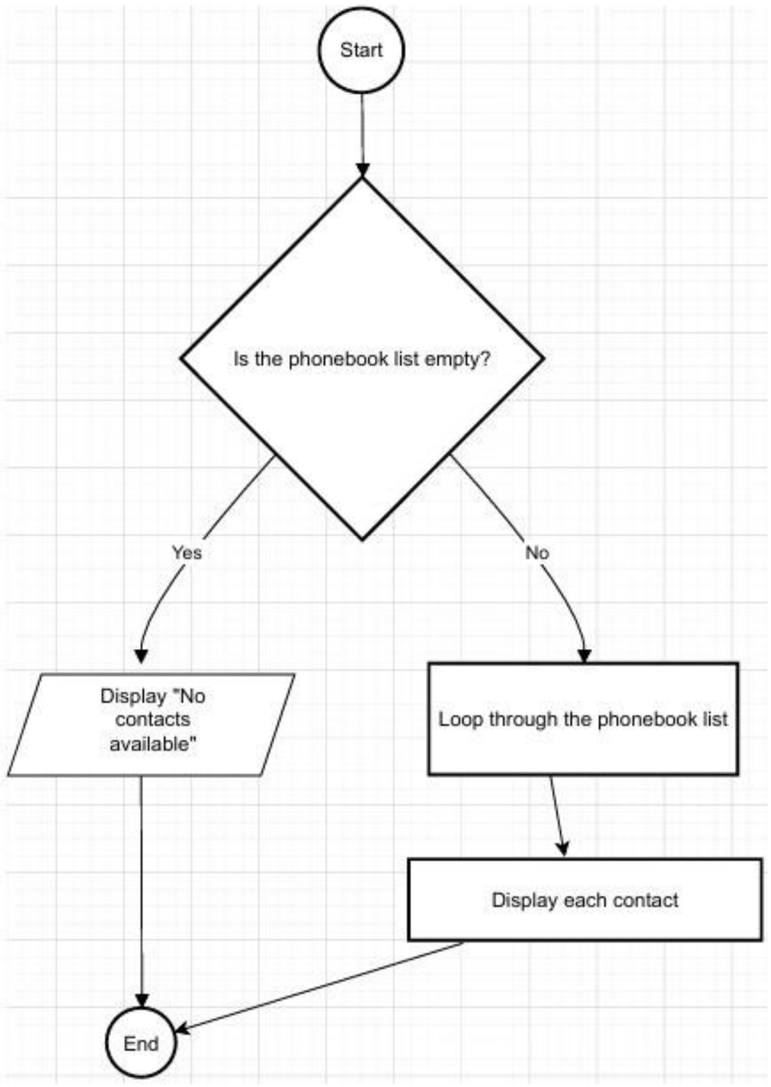
1. **Insert Contact**:


Start
Start
Prompt user to enter contact details (name, phone number)
Enter details
Get name and phone number
Got details
Create a new contact
Contact created
Add the contact to the phonebook list
Add the contact to the phonebook list
End

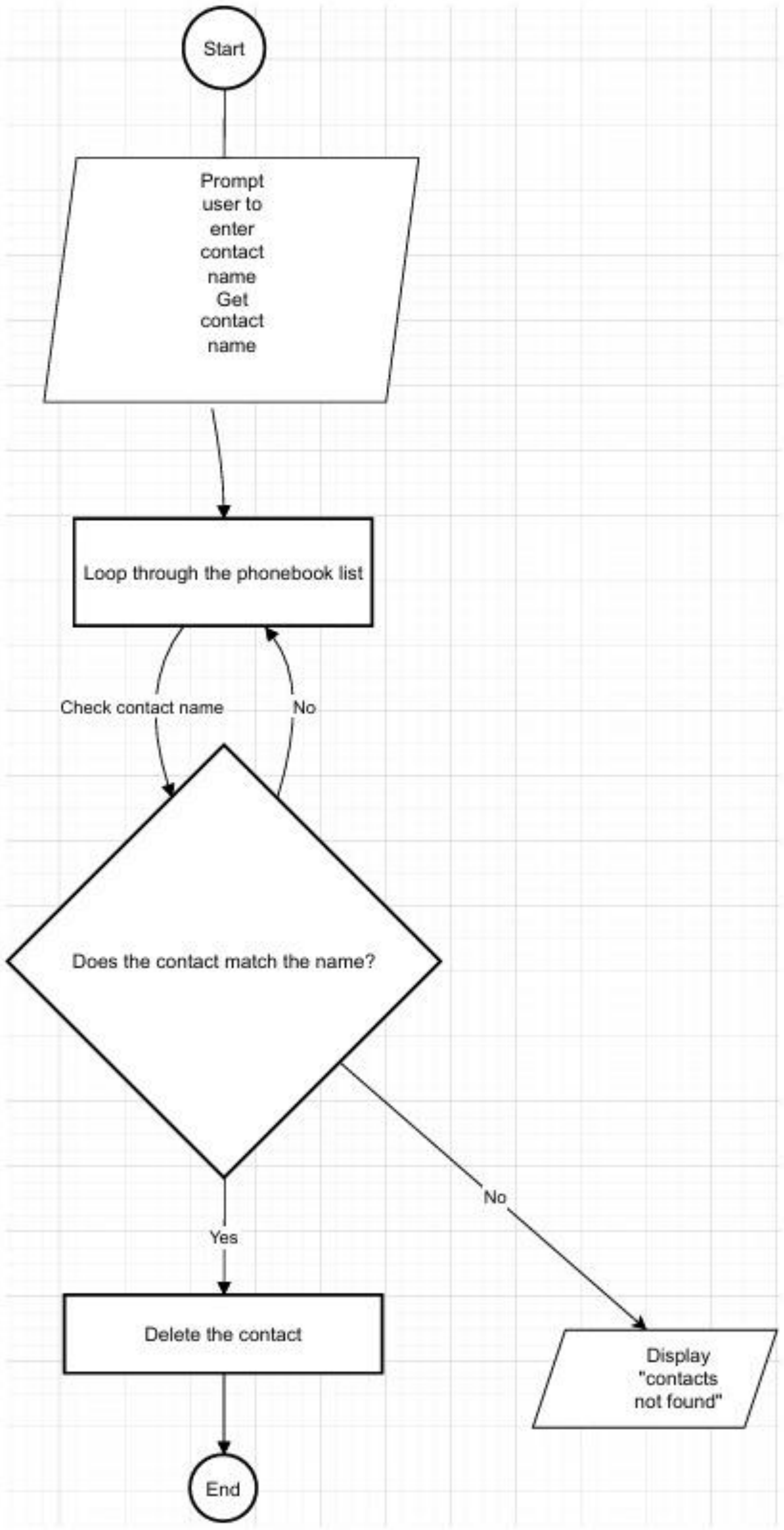
2. Search Contact


Start
Start
Prompt user to enter search query (name or phone number)
Query
Prompt user to enter search query (name or phone number) Get search query
Query
Loop through the phonebook list
Does a contact match the query?
Yes
No
Display "Contact found"
Display "contact not found"
End

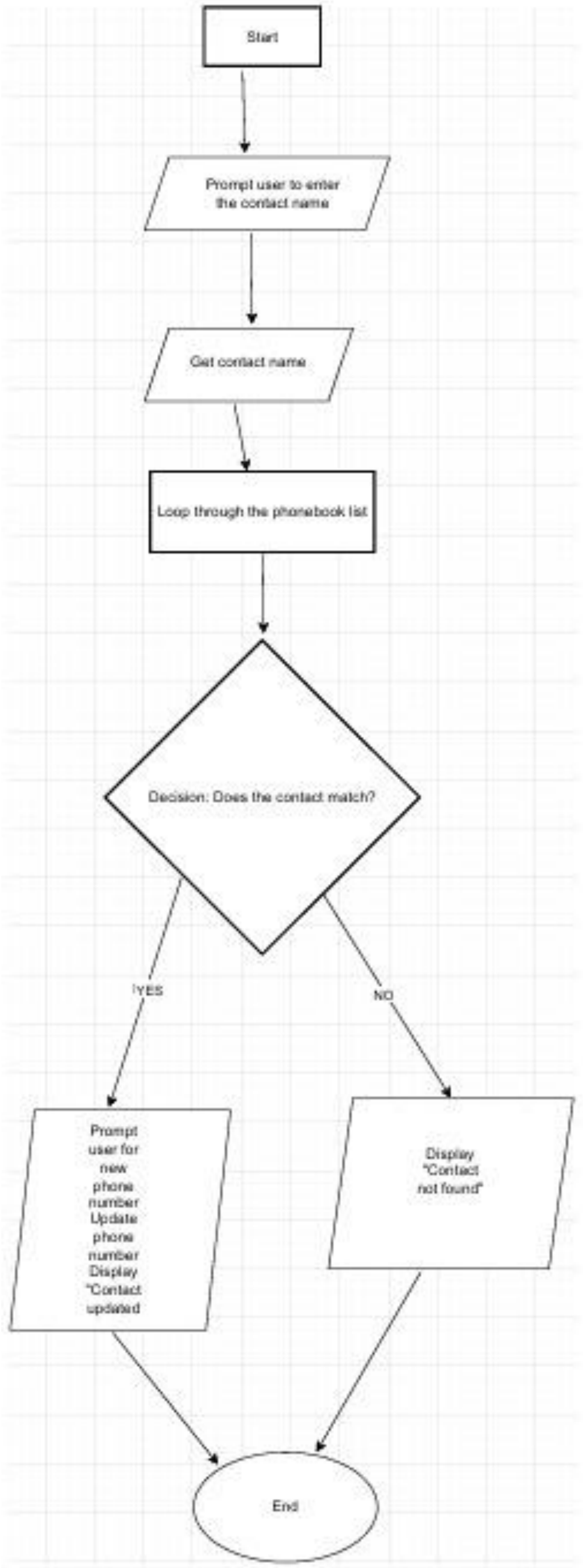
3. Display Contact Flowchart


Start
Is the phonebook list empty?
Yes
No
Display "No contacts available"
Loop through the phonebook list
Display each contact
End

4. Delete Contact Flowchart

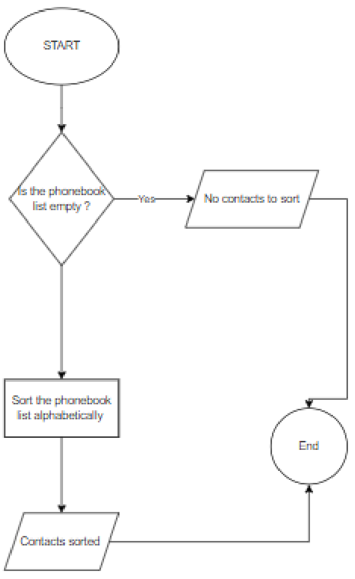

Start
Prompt user to enter contact name Get contact name
Loop through the phonebook list
Check contact name
No
Does the contact match the name?
No
Yes
Delete the contact
Display "contacts not found"
End

5. Update Contact Flowchart

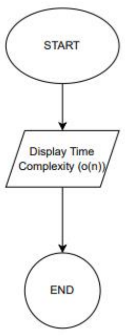


Start
Prompt user to enter the contact name
Get contact name
Loop through the phonebook list
Decision: Does the contact match?
YES
NO
Prompt user for new phone number Update phone number Display "Contact updated
Display "Contact not found"
End

6. Sort Contact Flowchart


START
Is the phonebook list empty ?
Yes
No contacts to sort
Sort the phonebook list alphabetically
End
Contacts sorted

7. Analyze Search Efficiency Flowchart



CODES FOR EACH KEY FUNCTIONALITY

- Insert Contact:

```
import java.util.ArrayList;
class Contact {
    private String name;
    private String phoneNumber;

    public Contact(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    public String getName() {
        return name;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Phone: " + phoneNumber;
    }
}

public class PhoneBook {
    private ArrayList<Contact> contacts;
    public PhoneBook() {
        contacts = new ArrayList<>();
    }

    public void insertContact(String name, String phoneNumber) {
        // Check for duplicates
        for (Contact contact : contacts) {
            if (contact.getName().equalsIgnoreCase(name)) {
                System.out.println("Contact already exists. Please use a
different name.");
                return;
            }
        }
        contacts.add(new Contact(name, phoneNumber));
        System.out.println("Contact " + name + " added successfully!");
    }

    public void displayContacts() {
        for (Contact contact : contacts) {
```

```java
            System.out.println(contact);
        }
    }

    public static void main(String[] args) {
        PhoneBook phoneBook = new PhoneBook();
        phoneBook.insertContact("Marcel", "085-456-3489");
        phoneBook.insertContact("Josh", "085-654-0984");
        phoneBook.insertContact("Eric", "081-555-4390"); // Duplicate name

        // Display the phone book
        phoneBook.displayContacts();
    }
}
```

- **Search Contact**

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Contact {
    String name;
    String number;

    public Contact(String name, String number) {
        this.name = name;
        this.number = number;
    }
    @Override
    public String toString() {
        return "Name: " + name + ", Number: " + number;
    }
}

public class Phonebook {
    private final List<Contact> contacts;

    public Phonebook() {
        contacts = new ArrayList<>();
    }
    public void addContact(String name, String number) {
        contacts.add(new Contact(name, number));
    }
    public void searchContact(String query) {
        for (Contact contact : contacts) {
            if (contact.name.equalsIgnoreCase(query) ||
contact.number.equals(query)) {
                System.out.println("Contact found: " + contact);
                return; // End search if contact is found
            }
        }
        System.out.println("Contact not found");
    }
    public static void main(String[] args) {
        Phonebook phonebook = new Phonebook();
        Scanner scanner = new Scanner(System.in);

        // Adding some sample contacts
        phonebook.addContact("Emmie", "3353801");
        phonebook.addContact("Renate", "5759914");
        phonebook.addContact("Ben", "7957478");

        // Search for a contact
        System.out.print("Enter the contact name or number to search: ");
        String query = scanner.nextLine();

        // Perform the search
        phonebook.searchContact(query);

        scanner.close();
    }
}
```

- Display Contacts

```java
public class Phonebook {
    // 2D array to store contacts (name, phone number)
    String[][] contacts;

    // Constructor to initialize the phonebook with some contacts
    public Phonebook() {
        // Initial contacts
        contacts = new String[][] {
                {"Bernard", "081-123-4567"},
                {"Eurico", "082-987-8900"},
                {"Angel", "083-456-9700"}
        };
    }

    // Method to display all contacts
    public void displayAllContacts() {
        if (contacts.length == 0) {
            System.out.println("No contacts available.");
        } else {
            System.out.println("Contact List:");
            for (int i = 0; i < contacts.length; i++) {
                System.out.println((i + 1) + ". Name: " + contacts[i][0] +
", Phone Number: " + contacts[i][1]);
            }
        }
    }

    // Main method to test the display function
    public static void main(String[] args) {
        Phonebook phonebook = new Phonebook();
        phonebook.displayAllContacts();
    }
}
```

- **Delete Contact**

```java
import java.util.Scanner;
public class PhoneBook {
    // 2D array to store contacts (name, phone number)
    String[][] contacts;

    // Constructor to initialize the phonebook with some contacts
    public PhoneBook() {
        // Initial contacts
        contacts = new String[][] {
                {"Alice", "081-993-4567"},
                {"Bob", "082-457-6543"},
                {"Charlie", "083-908-7890"}
        };
    }


    // Method to display all contacts
    public void displayAllContacts() {
        if (contacts.length == 0) {
            System.out.println("No contacts available.");
        } else {
            System.out.println("Contact List:");
            for (int i = 0; i < contacts.length; i++) {
                System.out.println((i + 1) + ". Name: " + contacts[i][0] +
", Phone Number: " + contacts[i][1]);
            }
        }
    }

    // Method to delete a contact by name
    public void deleteContact(String name) {
        int indexToDelete = -1;

        // Find the index of the contact to delete
        for (int i = 0; i < contacts.length; i++) {
            if (contacts[i][0].equalsIgnoreCase(name)) {
                indexToDelete = i;
                break;
            }
        }

        // If contact is found, create a new array without the contact
        if (indexToDelete != -1) {
            String[][] newContacts = new String[contacts.length - 1][2];
            for (int i = 0, j = 0; i < contacts.length; i++) {
                if (i != indexToDelete) {
                    newContacts[j++] = contacts[i];
                }
            }
            contacts = newContacts; // Update contacts to the new array
            System.out.println(name + " has been deleted from the
phonebook.");
        } else {
            System.out.println("Contact not found: " + name);
        }
    }

    // Main method to test the functionality
    public static void main(String[] args) {
        PhoneBook phonebook = new PhoneBook();
        Scanner scanner = new Scanner(System.in);

        // Display all contacts before deleting
        phonebook.displayAllContacts();

        // Prompt user for the contact to delete
        System.out.print("Enter the name of the contact you want to delete: 
");
        String contactToDelete = scanner.nextLine();
        phonebook.deleteContact(contactToDelete);
        phonebook.displayAllContacts();
        scanner.close();
    }
}
```

- **Update Contact.**

```
import java.util.ArrayList;
import java.util.List;

class Contact {
    String name;
    String number;

    public Contact(String name, String number) {
        this.name = name;
        this.number = number;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Number: " + number;
    }
}
public class Phonebook {
    private final List<Contact> contacts;
    public Phonebook() {
        contacts = new ArrayList<>();
    }

    public void addContact(String name, String number) {
        contacts.add(new Contact(name, number));
    }

    public void updateContact(String oldName, String newName, String 
newNumber) {
        for (Contact contact : contacts) {
            if (contact.name.equalsIgnoreCase(oldName)) {
                contact.name = newName;
                contact.number = newNumber;
                System.out.println("Contact updated: " + contact);
                return;
            }
```

```java
        }
        System.out.println("Contact not found");
    }

    public static void main(String[] args) {
        Phonebook phonebook = new Phonebook();
        phonebook.addContact("Emelda", "081-123-4567");
        phonebook.addContact("Ben", "082-987-6543");
        phonebook.addContact("Nao", "083-456-7890");

        // Update a contact
        phonebook.updateContact("Emelda", "Emily", "081-765-4321");
    }
}
```

- **Sort Contacts**

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;

class Contact {
    private String name;
    private String phoneNumber;

    public Contact(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    public String getName() {
        return name;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Phone: " + phoneNumber;
    }
}

public class PhoneBook {
    private Contact[] contactsArray;
    private ArrayList<Contact> contactsList;
    private int size;

    public PhoneBook(int capacity) {
        contactsArray = new Contact[capacity];
        contactsList = new ArrayList<>();
        size = 0;
    }
    public void insertContact(String name, String phoneNumber) {
        if (size >= contactsArray.length) {
            System.out.println("Phone book is full. Cannot add more
contacts.");
            return;
        }

        // Check for duplicates
        for (Contact contact : contactsList) {
            if (contact.getName().equalsIgnoreCase(name)) {
                System.out.println("Contact already exists. Please use a
different name.");
                return;
            }
        }

        Contact newContact = new Contact(name, phoneNumber);
        contactsArray[size] = newContact; // Add to array
        contactsList.add(newContact); // Add to ArrayList
        size++;
        System.out.println("Contact " + name + " added successfully!");
    }

    public void displayContacts() {
        System.out.println("Contacts in Array:");
        for (int i = 0; i < size; i++) {
            System.out.println(contactsArray[i]);
        }

        System.out.println("Contacts in List:");
        for (Contact contact : contactsList) {
            System.out.println(contact);
        }
    }

    public void sortContacts() {
        Collections.sort(contactsList, new Comparator<Contact>() {
            @Override
            public int compare(Contact c1, Contact c2) {
                return c1.getName().compareToIgnoreCase(c2.getName());
            }
        });

        System.out.println("Contacts sorted by name:");
        for (Contact contact : contactsList) {
            System.out.println(contact);
        }
    }

    public static void main(String[] args) {
        PhoneBook phoneBook = new PhoneBook(5);
        phoneBook.insertContact("Amunyela", "081-972-1681");
        phoneBook.insertContact("Kapuire", "085-654-5380");
        phoneBook.insertContact("Susan", "085-555-6690");

        System.out.println("Contacts before sorting:");
        phoneBook.displayContacts();

        phoneBook.sortContacts();
    }
}
```

- **Analyze Search Efficiency**

```
import java.util.Arrays;
import java.util.Scanner;

public class Phonebook {
    // 2D array to store contacts (name, phone number)
    private String[][] contacts;
    private int size;

    // Constructor to initialize the phonebook
    public Phonebook() {
        contacts = new String[10][2]; // Initial capacity of 10
        size = 0; // Number of contacts in the phonebook
    }

    // Method to insert a new contact
    public void insertContact(String name, String phoneNumber) {
        if (size == contacts.length) {
            expandArray(); // Expand the array when it reaches capacity
        }
        contacts[size][0] = name;
        contacts[size][1] = phoneNumber;
        size++;
        System.out.println("Contact added: " + name + ", " + phoneNumber);
    }

    // Method to search for a contact
    public void searchContact(String query) {
        boolean found = false;
        for (int i = 0; i < size; i++) {
            if (contacts[i][0].equalsIgnoreCase(query) ||
contacts[i][1].equals(query)) {
                System.out.println("Contact found: Name: " + contacts[i][0]
+ ", Phone Number: " + contacts[i][1]);
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Contact not found.");
        }
    }

    // Method to display all contacts
    public void displayAllContacts() {
        if (size == 0) {
            System.out.println("No contacts available.");
        } else {
            System.out.println("Contact List:");
            for (int i = 0; i < size; i++) {
                System.out.println((i + 1) + ". Name: " + contacts[i][0] +
", Phone Number: " + contacts[i][1]);
            }
```

```java
        }
    }

    // Method to delete a contact
    public void deleteContact(String name) {
        boolean found = false;
        for (int i = 0; i < size; i++) {
            if (contacts[i][0].equalsIgnoreCase(name)) {
                System.out.println("Contact deleted: " + contacts[i][0]);
                // Shift all elements after the deleted contact to the left
                for (int j = i; j < size - 1; j++) {
                    contacts[j] = contacts[j + 1];
                }
                size--;
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Contact not found.");
        }
    }

    // Method to update a contact's phone number
    public void updateContact(String name, String newPhoneNumber) {
        boolean found = false;
        for (int i = 0; i < size; i++) {
            if (contacts[i][0].equalsIgnoreCase(name)) {
                contacts[i][1] = newPhoneNumber;
                System.out.println("Contact updated: Name: " +
contacts[i][0] + ", New Phone Number: " + newPhoneNumber);
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Contact not found.");
        }
    }

    // Method to sort contacts alphabetically by name
    public void sortContacts() {
        Arrays.sort(contacts, 0, size, (a, b) ->
a[0].compareToIgnoreCase(b[0]));
        System.out.println("Contacts sorted alphabetically by name.");
    }

    // Method to analyze the efficiency of the search algorithm (linear
search)
    public void analyzeSearchEfficiency() {
        System.out.println("Search algorithm efficiency: O(n), where n is
the number of contacts.");
    }

    // Private method to expand the array when it's full
    private void expandArray() {
        String[][] newContacts = new String[contacts.length * 2][2];
        for (int i = 0; i < size; i++) {
            newContacts[i] = contacts[i];
        }
        contacts = newContacts;
    }

    // Main method to test the phonebook functionality
    public static void main(String[] args) {
        Phonebook phonebook = new Phonebook();
        Scanner scanner = new Scanner(System.in);

        int choice;
        do {
            System.out.println("\nPhonebook Menu:");
            System.out.println("1. Insert Contact");
            System.out.println("2. Search Contact");
            System.out.println("3. Display All Contacts");
            System.out.println("4. Delete Contact");
            System.out.println("5. Update Contact");
            System.out.println("6. Sort Contacts");
            System.out.println("7. Analyze Search Efficiency");
            System.out.println("8. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter contact name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter phone number: ");
                    String phoneNumber = scanner.nextLine();
                    phonebook.insertContact(name, phoneNumber);
                    break;
                case 2:
                    System.out.print("Enter contact name or phone number to
search: ");
                    String query = scanner.nextLine();
                    phonebook.searchContact(query);
                    break;
                case 3:
                    phonebook.displayAllContacts();
                    break;
                case 4:
                    System.out.print("Enter contact name to delete: ");
                    String deleteName = scanner.nextLine();
                    phonebook.deleteContact(deleteName);
                    break;
                case 5:
                    System.out.print("Enter contact name to update: ");
                    String updateName = scanner.nextLine();
                    System.out.print("Enter new phone number: ");
                    String newPhoneNumber = scanner.nextLine();
                    phonebook.updateContact(updateName, newPhoneNumber);
                    break;
                case 6:
                    phonebook.sortContacts();
                    break;
                case 7:
                    phonebook.analyzeSearchEfficiency();
                    break;
                case 8:
                    System.out.println("Exiting the program.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try
again.");
            }
        } while (choice != 8);

        scanner.close();
    }
}
```

Link to Git : https://github.com/PapiBlonco/PhoneBook

Conclusion

In conclusion, this structured approach to the phone book application guarantees a robust and efficient solution that meets the project's requirements. By leveraging linear data structures, we achieve simplicity and clarity in the implementation, making the codebase easy to maintain and understand. Furthermore, this foundation allows for future scalability and the integration of advanced data structures, paving the way for potential enhancements that can improve functionality and performance.