Mini Project Report on “Efficient Phonebook Application”

Vivekanand Education Society’s Institute of Technology

Department of Information Technology

DSA Mini Project (A.Y. 2025–26)

Student Name: Kesar .A. Duseja

Class: D10C **Roll No:** 10

Mentor: Dipti Karani Ma’am

Domain: Data Structures and Algorithms (DSA)

Sustainability Goal: "Promotes paperless record-keeping".

Acknowledgement

I would like to express my sincere gratitude to my mentor, Mrs. Dipti Karani , for her invaluable guidance, constant support, and encouragement throughout the development of this mini project. Her insights were crucial in shaping the **"Efficient Phonebook Application using Data Structures"**.

I also extend my thanks to the Department of Information Technology at **Vivekanand Education Society's Institute of Technology** for providing the necessary academic resources and a supportive environment for this DSA mini project.

Kesar Anil Duseja

1. Introduction to the Project

The Phonebook Application is a **user-friendly, Java-based project** designed to digitally store, manage, and efficiently organize contact information. It serves as an **eco-friendly digital solution** that replaces traditional paper-based record systems. The application is built using core programming concepts and data structures such as **Arrays, Lists, and File Handling** to manage data storage and retrieval. By automating contact management, it minimizes errors and promotes the sustainability goal of **"Promotes paperless record-keeping"**.

2. Problem Statement

In the current digital environment, managing contact information manually is often **inefficient and error-prone**. Individuals and organizations frequently struggle to maintain accurate, accessible, and up-to-date contact records when using unorganized or paper-based methods.

This project addresses the need for a **digital solution** that simplifies the storing, searching, updating, and deleting of contact information efficiently. The goal is to develop a Java-based application utilizing data structures like **Arrays and File Handling** to ensure quick data retrieval, reliability, and support paperless record management.

3. Objectives of the Project

The project's main objectives are:

- To design a digital phonebook system for **efficient storage, management, and retrieval** of contact information.
- To apply core programming and Data Structure concepts, specifically **Arrays and File Handling**, for effective data organization.
- To develop a **user-friendly graphical interface** using **Java Swing** for intuitive interaction.
- To **reduce manual effort** and time spent on managing contact details through automation.
- To promote **sustainability** by encouraging paperless, eco-friendly digital record management practices.

4. System Requirements

Software Requirements

- **Operating System**: Windows / macOS / Linux
- **Programming Language**: Java (JDK 21)
- **IDE / Editor**: IntelliJ IDEA / Eclipse / NetBeans
- **Libraries Used**: javax.swing (for GUI components) , java.awt (for graphics and layout) , java.util (for ArrayList, Random, Timer).
- **Database**: Not required (data handled using Arrays, Stacks, and Queues, and in-memory structures).

Hardware Requirements

- **Processor**: Minimum 1 GHz or higher
- **RAM**: 2 GB or more recommended (Minimum 512 MB for Java runtime)
- **Storage**: At least 100 MB of free disk space

- **Display**: Standard monitor with 1024×768 resolution or higher

5. Data Structure & Concepts Used

Data Structures Used

- **ArrayList** (ArrayList<Contact> contactList): Used to **dynamically store all contact objects**, allowing for easy addition, deletion, and iteration over contacts.
- **Objects & Classes**: Each contact is modeled as a **Contact object**, encapsulating attributes like name and phone number.
- **DefaultTableModel & JTable**: Manages the tabular display of contact data and efficiently updates the UI.

Key Concepts Used

- **Object-Oriented Programming (OOP)**
- **Event-Driven Programming**: Swing components respond to user actions via listeners for interactive functionality.
- **Sequential Access & Iteration**: Operations like Search, Delete, and Display systematically traverse the contact list.
- **Validation and Error Handling**

6. Algorithm Explanation

The application's core functions are implemented through simple, sequential algorithms:

1. **Add Contact**: The user enters a name and number, the input is checked, the contact is **added to the list**, and the table is updated.
2. **Search Contact**: The user types a keyword, the program **finds matching contacts** (using filters), and shows the results in the table.
3. **Delete Contact**: The user selects a contact, confirms deletion, the contact is **removed from the list**, and the table refreshes.
4. **Display All Contacts**: The application shows all saved contacts in the table.

7. Time and Space Complexity

Operation	Complexity	Description
Add Contact	$O(1)$	Adding a contact to the list is constant time[42].
Search Contact	$O(n)$	May require checking all n contacts in the list[43].
Delete Contact	$O(n)$	May require finding the contact before deletion[44].
Display All Contacts	$O(n)$	Requires traversing the entire list of n contacts[45].

Space Complexity

- **Contacts List**: O(n) – Space usage **grows linearly** with the number (n) of contacts.
- **Table Display / Temporary Storage**: O(n) – Space is needed for showing n contacts in the table.
- **Other Variables**: O(1) – Constant space for counters, flags, and temporary variables.

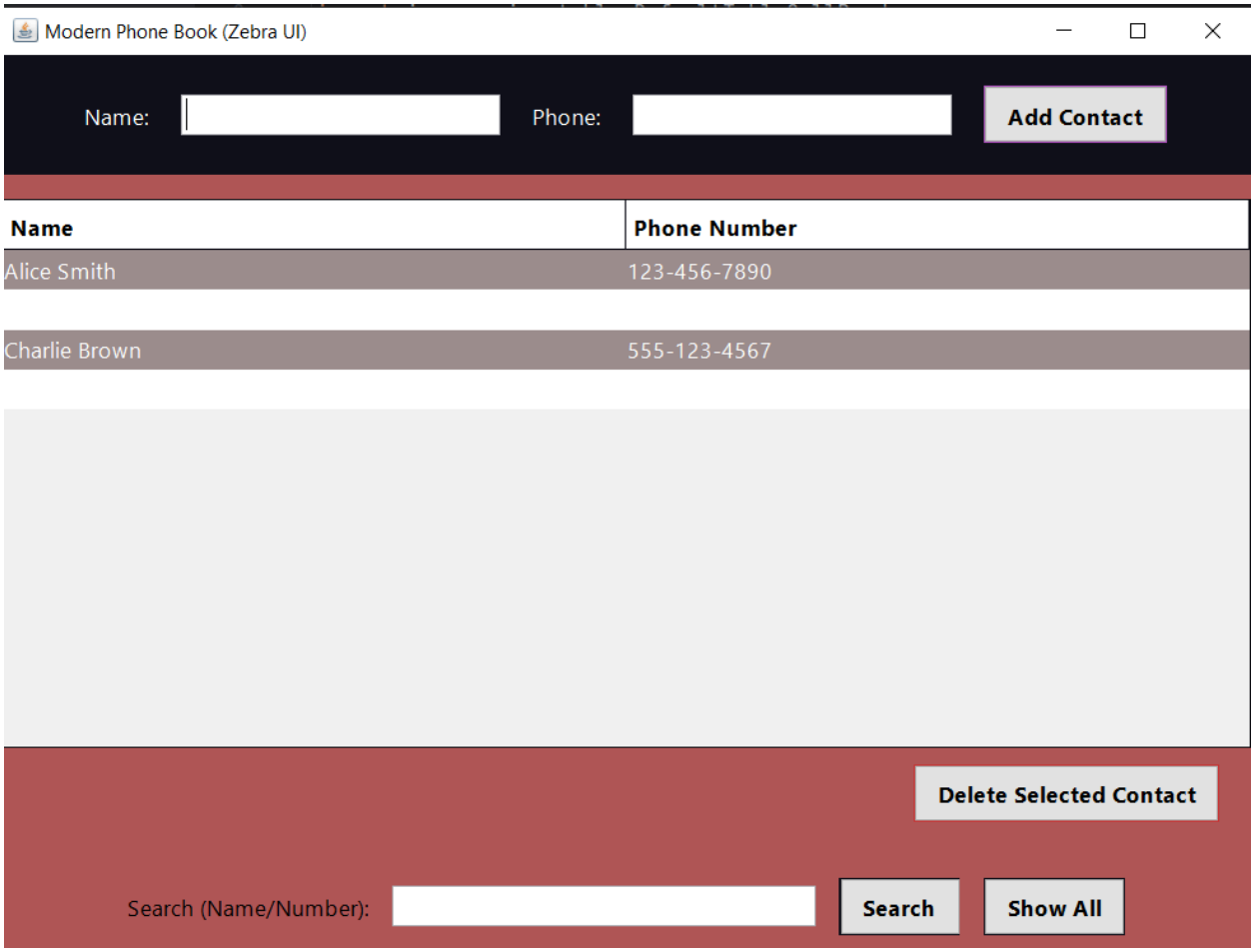
8. Front End and Implementation

The front-end is developed using **Java Swing** , providing an interactive and user-friendly interface.

- **Components**: Key components include JFrame (main window), JPanel (organizes fields and buttons), JButton (for actions like Add, Search, Delete), JTextField (for input), and **JTable** and **DefaultTableModel** to display the contact list in a structured format.
- **Implementation**: The application is built using a **Contact class** (for name & phone number) and an **ArrayList** to manage contacts dynamically. Buttons handle operations through **ActionListeners** , and **JOptionPane** is used to show feedback messages. The

application features a **user-friendly interface** with clear visual feedback and table formatting.

9. Output



10. Conclusion

The **Efficient Phonebook Application** is an interactive, easy-to-use, and effective tool for managing contacts. It successfully demonstrates the application of **Object-Oriented Programming**, event-driven GUI design, and dynamic data handling. The project efficiently supports essential contact management operations—Add, Search, Delete, and Display—with real-time updates. This application provides a valuable example of practical DSA use in an engaging digital tool.

11. References

1. **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.** (2022). *Introduction to Algorithms*. (4th ed.). MIT Press. (Source for Hash Table, Hashing Functions, and Algorithmic Analysis concepts).
2. **Knuth, D. E. (1998).** The Art of Computer Programming, Volume 3: Sorting and Searching. (2nd ed.). Addison-Wesley Professional. (Advanced concepts for hash function design and analysis).
3. **Vivekanand Education Society's Institute of Technology. (2025).** Data Structures & Algorithms Course Material.