DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

PHONE BOOK MANAGEMENT SYSTEM IN C++

A PROJECT REPORT OF DATA STRUCTURES

Submitted by

Ajay Raghav (22BCS16075)

Disha (22BCS16077)

Aman Pratap(22BCS16078)

To

Prof. Sheikh Afaan Farooq

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Nov 2024

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my professors and mentors at Chandigarh University, Department of Computer Science, for their guidance and support during the development of the Phone Book Management System project. Their valuable insights and encouragement have been instrumental in my learning and growth as a computer science student.

I would also like to thank my fellow students for their collaborative efforts, discussions, and feedback that contributed to the project's success.

Additionally, I appreciate the resources and materials provided by the university, which have been invaluable in acquiring the skills and knowledge necessary for this project.

Thank you to all who have played a part in this endeavor. Your support has been greatly appreciated.

THANK YOU.

CONTENT

INTRODUCTION

Introduction to Phone Book Management System in C++: In today's fast-paced digital world, managing our contacts efficiently has become an essential task. Whether it's for personal use or in a professional setting, a well-organized and user-friendly system for managing contact information is invaluable. This is where a Phone Book Management System comes into play. Our Phone Book Management System, designed and implemented in C++, is a versatile and user-friendly solution that simplifies the process of storing, retrieving, and updating contact information. This software application provides a convenient way to manage a vast database of contacts, making it easy to access and maintain essential data at your fingertips. The system is developed in C++, a powerful and versatile programming language known for its efficiency and performance. A phone book management system, developed in C++, represents a robust and efficient software application designed to streamline the organization and accessibility of contact information. In an era characterized by an ever-expanding network of personal and professional connections, this system serves as a crucial tool for individuals and businesses alike. It offers a comprehensive solution for storing, retrieving, and managing a vast array of contacts, ensuring that vital information is readily available at your fingertips. With its user-friendly interface and advanced features, this C++-powered phone book management system stands out as a versatile and reliable resource for maintaining contact details, making it an indispensable asset for users seeking to efficiently manage their communication networks. Whether you're an individual looking to keep track of friends and family or a business professional in need of a centralized directory for clients and colleagues, this system provides a seamless and efficient solution for contact management in the digital age.

PROJECT OBJECTIVES

The primary objectives of this project are as follows:

- Implement a working Phone Books Management System.
- Design a user-friendly interface.
- Provide important options such as Add Contacts, Search, Display.
- Implement logic to determine the user needs.
- Display the final output asked by the user.

PROJECT OVERVIEW

- **Objective:** The primary objective of this project is to create a user-friendly and efficient Phone Book Management System using C++ that allows users to store, organize, and retrieve contact information easily.

- **Features:** The system will include features such as adding, editing, and deleting contact entries, searching for contacts by various parameters (e.g., name, phone number, or address), and categorizing contacts into groups (e.g., family, friends, work). It will also provide the ability to import and export contact data for data backup and sharing.

- **User Interface:** The system will offer an intuitive and user-friendly command-line interface, ensuring ease of use for a wide range of users. It will provide clear menu options for performing various tasks, making it accessible to both novice and experienced users.

- **Data Security:** Data security and privacy will be a top priority. The system will implement data encryption and access control mechanisms to safeguard contact information. User data will be stored securely and protected against unauthorized access.

- **Portability and Extensibility:** The Phone Book Management System will be designed with portability in mind, allowing it to run on various C++-compatible platforms. Additionally, the system will be extensible, making it possible to add new features and functionalities in the future, ensuring it remains adaptable to changing user needs.

IMPLEMENTATION DETAILS

Header Files:

- You'll start by creating header files for classes and functions that will be used in your program. You may need headers for Contact and PhoneBook.

Contact Class:

- Create a Contact class to represent individual contacts. This class should have private data members like name, phone number, email, etc., and public member functions to manipulate and access this data.

PhoneBook Class:

- Create a PhoneBook class that will manage a list of contacts. The PhoneBook class should have a private data member, like an array or a vector, to store the contacts.
- Implement functions in the PhoneBook class for adding, deleting, updating, and searching for contacts.
- You may also want to implement functions for displaying all contacts and sorting them based on different criteria (e.g., by name, by phone number).

User Interface:

- Implement a user interface for interacting with the Phone Book. You can create a simple command-line interface or a graphical user interface, depending on your preferences.
- In a command-line interface, you can use a loop to display a menu and take user input to perform various operations.

Data Storage:

- You need a way to persistently store contact information. You can use file handling to save contacts to a file, and load them when the program starts.

Menu and Input Handling:

- Implement a menu system that allows users to select different options (e.g., add contact, delete contact, search, display all, exit).
- Handle user input and validate it to ensure it matches the expected format.

Error Handling:

- Implement error handling to deal with invalid input and potential issues when reading/writing data files.

Testing:

- Test your Phone Book Management System by adding, updating, deleting, and searching for contacts. Ensure that it handles different scenarios gracefully and accurately.

Documentation:

- Document your code using comments to make it clear and understandable for others who may work on it in the future.

Optimizations:

- If your phone book becomes large, consider optimizing the search and sorting algorithms to improve performance.

Security:

- Implement user authentication and authorization if necessary, to ensure that only authorized users can access and modify the phone book.

Enhancements:

- Depending on your project requirements, you can add more features like contact groups, reminders, or a more advanced user interface.

CODE

```
#include <iostream>
#include <string>
#include <map>

using namespace std;

class PhoneBook {
public:

    void addContact(const string& name, const string&
phoneNumber) {
        contacts[name] = phoneNumber;
        cout << "Contact added: " << name << " - " <<
phoneNumber << endl;
    }
    void searchContact(const string& name) {
        auto it = contacts.find(name);
        if (it != contacts.end()) {
            cout << "Contact found: " << it->first << " - " <<
it->second << endl;
        } else {
            cout << "Contact not found: " << name << endl;
        }
    }
```

```
  void displayContacts() {

  cout << "Phone Book Contacts:" << endl;
      for (const auto& contact : contacts) {
          cout << contact.first << " - " << contact.second <<
endl;
      }
   }

private:
   map<string, string> contacts;
};

int main() {
   PhoneBook phoneBook;
   int choice;
   string name, phoneNumber;

   while (true) {
      cout << "Phone Book Management System" <<
endl;
      cout << "1. Add Contact" << endl;
      cout << "2. Search Contact" << endl;
      cout << "3. Display Contacts" << endl;
      cout << "4. Quit" << endl;
      cout << "Enter your choice: ";
```

```
 cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1:
                cout << "Enter name: ";
                getline(cin, name);

cout << "Enter phone number: ";
                cin >> phoneNumber;
                phoneBook.addContact(name, phoneNumber);
                break;
            case 2:
                cout << "Enter name to search: ";
                cin.ignore();
                getline(cin, name);
                phoneBook.searchContact(name);
                break;
            case 3:
                phoneBook.displayContacts();
                break;
            case 4:
                cout << "Goodbye!" << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." <<
endl;
```

```
    }
     }

     return 0;
}
```

FUTURE ENHANCEMENT

Certainly, here are some potential future enhancements for a Phone Book Management System in C++:

1. Multi-platform Compatibility:

- Adapt the application for multiple platforms, including Windows, macOS, and Linux. This can involve creating platform-specific user interfaces or using cross-platform libraries to ensure broad compatibility.

2. Graphical User Interface (GUI):

- Enhance the user experience by developing a graphical user interface (GUI) using a library like Qt or wx Widgets. A GUI can make the application more user-friendly and visually appealing.

3. Cloud Integration:

- Implement cloud integration to synchronize contacts across multiple devices. This feature would allow users to access their phone book from anywhere and ensure data consistency.

4. Backup and Restore:

- Create a feature for users to back up their contact data and restore it when needed. This can be implemented as an export/import functionality for contacts.

5. Contact Import/Export Formats:

- Add support for importing and exporting contact data in various formats such as vCard, CSV, or JSON. This will make it easier for users to migrate data to and from other applications.

6. Contact Sharing:

- Enable contact sharing through various means, such as email or messaging apps, allowing users to share their contacts with others more conveniently.

7. Advanced Search and Filtering:

- Enhance the search and filtering capabilities, enabling users to search for contacts using multiple criteria or filter contacts based on custom-defined.

8. Integration with Other Apps:

- Integrate the phone book with other applications or services, such as email clients, calendars, or task management tools, to create a more comprehensive personal information management system.

9. Contact Photo Support:

- Add the ability to associate photos with contacts. This can make it easier to recognize and connect with people.

10. Data Encryption and Security Features:

- Enhance data security by implementing encryption and secure authentication methods to protect sensitive contact information.

11. Accessibility Features:

- Ensure the application complies with accessibility standards, making it usable by people with disabilities.

12. Customizable Themes and Layouts:

- Allow users to customize the visual appearance of the application, including themes, layouts, and color schemes.

When considering future enhancements, it's important to prioritize features based on user feedback and emerging trends, as well as the specific needs of your target audience. Regular updates and improvements can help keep the Phone Book Management System relevant and competitive in the ever-evolving technology landscape.

CONCLUSION

Certainly, here's a conclusion for the Phone Book Management System project:

In developing the Phone Book Management System in C++, we have created a versatile and practical tool for organizing and managing contact information. This project aimed to provide users with a streamlined and efficient solution for storing, accessing, and manipulating their contact data. Throughout the project, we focused on practical implementation, usability, and the potential for future enhancements.

The core functionalities of this system allow users to add, update, delete, search for, and display contacts, making it easy to maintain an up-to-date and comprehensive phone book. The ability to save and load contact data from files ensures that the user's information remains intact even after closing and reopening the application.

In terms of future enhancements, there are numerous exciting possibilities for expanding the capabilities of the Phone Book Management System. These include integrating with cloud services for synchronization, creating a graphical user interface for improved user experience, and adding features like contact sharing, reminder notifications, and advanced search options. These enhancements can transform the system into a comprehensive personal information management tool.

In conclusion, the Phone Book Management System project demonstrates the practical application of C++ in building a useful and extensible tool. It serves as a foundation for further development and refinement, and its adaptability and scalability make it a valuable addition to personal or professional information management. This project showcases the potential for software to evolve, responding to user feedback and technological advancements, and it represents an ongoing commitment to providing users with efficient and secure ways to manage their contact information.