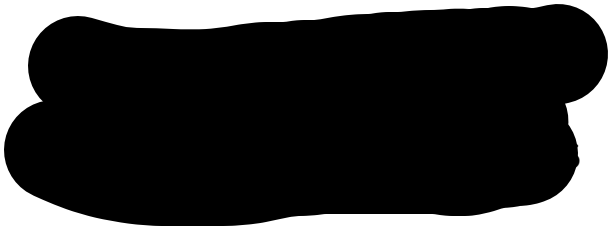

PhoneBook Contact Manager

Table of Contents

1. Abstract

The PhoneBook Contact Manager is a Qt-based GUI application designed for efficient contact management. It provides a user-friendly interface for adding, deleting, searching, and restoring contacts. The program ensures data persistence through file-based storage. Implementing a Model-View-Controller (MVC) architecture, it integrates real-time search functionality and shortcut commands to enhance usability. With minimal system requirements, the application is lightweight and optimized for cross-platform compatibility, making it an ideal solution for personal and professional contact organization.

2. Introduction

2.1 Project Overview

The **PhoneBook Contact Manager** is a graphical application developed using **C++ and the Qt framework**. It provides a seamless and efficient solution for managing contact records with an intuitive user interface. This application facilitates the **storage, retrieval, deletion, and persistent management** of contact information. The program integrates **dynamic search algorithms**, **undo functionality**, and **file I/O mechanisms** to ensure an optimal user experience.

The system utilizes:

- **Linked lists** for efficient contact storage and ordering.
- **Stacks** to enable an undo mechanism for accidental deletions.
- **File I/O** for persistent contact storage in a contacts.txt file.

2.2 Purpose and Scope

- **Purpose**: Simplify contact management through intuitive operations such as adding, deleting, searching, and undoing contact deletions.
- **Scope**: This project supports essential CRUD operations and real-time filtering but does not include advanced features such as editing, cloud sync, or contact groups.

2.3 Target Audience

- **End Users**: Individuals looking for a lightweight and effective tool to manage personal or professional contacts.
- **Developers**: Programmers interested in integrating **data structures** (linked lists, stacks) with **GUI applications** in **C++/Qt**.

3. System Requirements

3.1 Hardware Requirements

- **RAM**: Minimum 512 MB (Recommended: 2 GB for optimal performance)
- **Storage**: Minimum 10 MB of free disk space

3.2 Software Requirements

- **Qt Framework**: Version 5.15 or later
- **C++ Compiler**: GCC, MSVC, or Clang with **C++17** support

3.3 Operating System Compatibility

- **Windows**: Windows 10, Windows 11
- **macOS**: macOS 10.14+ (Mojave and later)
- **Linux**: Ubuntu 20.04+ and equivalent distributions

4. User Interface Guide

4.1 Getting Started

1. Launch the application.
2. The system will automatically load existing contacts from contacts.txt (if present).
3. The user can interact with the interface to add, delete, search, or undo actions.

4.2 Navigation and Layout

Main Window Structure

- **Top Section**: Search bar for real-time filtering.
- **Left Section**: Contact display area (tableWidget) showing stored contacts.
- **Bottom Section**: Input fields and action buttons.

4.3 Widget/Control Descriptions

Widget	Purpose
Name	Accepts a contact name.
Contact	Accepts a phone number.
Add Contact	Adds a new contact.
Delete	Deletes selected contact form table.
Undo	Restores the latest deleted contact.
Clear	Clears text from search bar.
Exit	Closes the application.

4.4 Menu and Shortcut Keys

- **Shortcut Keys**:
    - shift: Moves Focus on next interactable Field.

5. Functional Description

5.1 Core Features and Functionality

Adding a Contact

- Validates the input name and number.
- Inserts the contact into a sorted linked list.

Deleting a Contact

- Removes a contact from the list.
- Pushes the deleted contact onto an undo stack.

Searching Contacts

- Filters contacts dynamically based on name prefixes.
- Uses an optimized search algorithm for real-time results.

5.2 Data Input/Output and Storage

- **Input**: Validated name/contact entered through GUI.
- **Output**: Contacts are saved to contacts.txt upon exit.

File Format Example:

John Doe
03123456789
Alice Smith
03987654321

5.3 Error Handling and Debugging

Error	Solution
Invalid Contact Number	Ensure it starts with '03' and has 11 digits.
Duplicate Name	Ensure the contact name is unique and maximum 15 alphabets.

6. Technical Details

6.1 Architecture and Design Patterns

The application follows the **Model-View-Controller (MVC) pattern**:

- **Model**: codeManager handles contact logic.
- **View**: MainWindow manages UI elements.
- **Controller**: Qt signals/slots facilitate communication.

6.2 Libraries and Frameworks Used

- **QtWidgets**: Handles GUI components.
- **QtCore**: Provides file I/O and event-handling support.

6.3 Database Schema

- Not applicable. Data is stored in a simple text file.

7. Installation and Deployment

7.1 Installation Instructions

1. Clone the repository.
2. Open main.pro in Qt Creator.
3. Build and run the project.

7.2 Configuration Options

- Modify the path of contacts.txt in codeManager::saveToFile() if needed.

7.3 Troubleshooting Common Issues

- **Issue**: Missing Qt DLLs on Windows.
- **Fix**: Deploy using windeployqt.

8. Troubleshooting and Maintenance

8.1 Known Issues and Workarounds

- **Issue**: Undo works only once.
- **Workaround**: Implement a multi-level undo stack.

8.2 Updating and Patching

- Update the code from the repository and rebuild the application.

8.3 Backup and Recovery Procedures

- Backup: Manually copy contacts.txt regularly.

9. Future Development and Roadmap

9.1 Planned Features

- **Contact editing** functionality.
- **Multi-level undo** feature.
- **Export contacts to CSV**.

9.2 Community Feedback

- Users can submit feature requests and bug reports via GitHub.

10. Conclusion

The **PhoneBook Contact Manager** delivers an intuitive and efficient approach to contact management using C++ and Qt. By incorporating **dynamic data structures, persistent storage, and an interactive UI**, it offers a robust yet lightweight solution. While the application currently supports essential CRUD operations, future enhancements will include **contact editing, multi-level undo, and CSV export**. This project serves as both a practical tool for users and a valuable learning resource for developers exploring **Qt and C++ data structure integration**.

11. Appendices

11.1 Glossary of Terms

- **CRUD**: Create, Read, Update, Delete.
- **LIFO**: Last-In-First-Out (stack principle).

11.2 References and Resources

- Qt Documentation
- IEEE Software Documentation Standards
- **Github:** https://github.com/Muhammad-Abdullah-17/Qt-PhonebookUI.git