**Program Documentation**

**Purpose**

The purpose of this program is to provide a graphical user interface (GUI) for managing contacts in a contact book. The program allows users to add, edit, delete, import, and export contacts in various formats such as JSON and CSV. It uses the Tkinter library for creating the GUI and SQLite for storing the contacts in a database.

**Where it can be used?**

The program can be used in different companies whose field of activity is contacting with clients, having contacts. The idea is that every corporation needs a simple software for its employees to have their own “Contact Books” where they can store actual information about the clients they are working with and their information. In the future versions of this project it is planned to add some more actual information about contacts(emails, personal notes, etc.). The contacts then may be processed not only in database, but also in some WEB Application.

**Functionality**

The program provides the following functionality:

1. Add Contact: Allows users to add a new contact by entering their name and phone number. The program validates the phone number to ensure it is a valid Poland phone number.
2. Edit Contact: Allows users to edit an existing contact by selecting it from the contacts list and modifying the name and phone number.
3. Delete Contact: Allows users to delete a contact by selecting it from the contacts list.
4. Import Contacts: Allows users to import contacts from a JSON file or a CSV file. The program reads the file, parses the contacts data, and adds them to the contact book.
5. Export Contacts: Allows users to export contacts from the contact book to a JSON file or a CSV file. The program retrieves the contacts data, converts it to the desired format, and saves it to the specified file.
6. Toggle Menu: Provides a toggle menu bar on the left side of the window, allowing users to easily access import and export functionality.
7. Clock: Provides functionality of a clock inside application
8. Hotkeys Functionality:

Hotkeys are keyboard shortcuts that allow users to perform actions or navigate through a program quickly and efficiently without using the mouse. They provide a convenient way to access frequently used features or commands. In the context of a contact management system, hotkeys can offer users a faster and more streamlined experience. Here are some examples of hotkeys that could be implemented:

1. Ctrl + N: Create a new contact.
2. Ctrl + E: Edit the selected contact.
3. Ctrl + D: Delete the selected contact.
4. Ctrl + S: Save changes to the contact book.
5. Ctrl + F: Open the search bar to quickly find a contact.
6. Ctrl + H: Return to the home screen or contact list .
7. Using search bar press „Return” to search and “Backspace” to return home

By utilizing hotkeys, users can perform common tasks with a few keystrokes, reducing the need to navigate through menus or use the mouse. This can greatly improve productivity and efficiency when working with a contact management system.

1. Search Bar: A search bar is a text input field typically placed at the top of the user interface, allowing users to search for specific information within a system or application. In the context of a contact management system, a search bar enables users to quickly locate a contact based on various search criteria.
2. Home Button: It serves as a visual anchor and provides users with a quick way to return to the main screen or starting point of an application

**Instructions**

To run the program, follow these steps:

1. Install the required dependencies:
   * Tkinter (usually comes pre-installed with Python)
   * SQLite3 (usually comes pre-installed with Python)
2. Save the program code in a file with a .py extension, e.g., contactBookApp.py.
3. Open a terminal or command prompt and navigate to the directory where the program file is saved.
4. Run the program using the command: python contactBookApp.py. Also you should run: DataBaseCreate.py
   * Note: Make sure you have Python installed and added to the system's PATH variable.
5. The program window will appear, displaying the contact book.
6. You can perform various operations such as adding, editing, deleting, importing, and exporting contacts using the provided buttons and menus.

**Dependencies**

The program requires the following dependencies:

* Python 3.x (usually comes pre-installed on most systems)
* Tkinter library (usually comes pre-installed with Python)
* SQLite3 library (usually comes pre-installed with Python)

**Examples**

**Example 1: Adding a Contact**

1. Enter the name and phone number of the contact in the respective entry fields.
2. Click the "Add Contact" button.
3. The contact will be added to the contact book and displayed in the contacts listbox.

**Example 2: Editing a Contact**

1. Select a contact from the contacts listbox by clicking on it.
2. Double click on any object in list.
3. A new window will open with the selected contact's details.
4. Modify the name or phone number as desired.
5. Click the "Save" button.
6. The contact will be updated in the contact book and the changes will be reflected in the contacts listbox.

**Example 3: Deleting a Contact**

1. Select a contact from the contacts listbox by clicking on it.
2. Click the "Delete Contact" button.
3. The selected contact will be removed from the contact book and the contacts listbox.

**Example 4: Importing Contacts**

1. Click the "Import JSON" or "Import CSV" button in the toggle menu.
2. Select the JSON or CSV file containing the contacts to import.
3. The contacts from the file will be added to the contact book and displayed in the contacts listbox.

**Example 5: Exporting Contact**

1. Click the "Export JSON" or "Export CSV" button in the toggle menu.
2. Select the location and enter the file name for the exported contacts file.
3. Click the "Save" button.
4. The contacts from the contact book will be exported to the specified file in the selected format (JSON or CSV).

**Example 6: Searching Contact**

1. Click on the „Search Bar” and enter name or phone number
2. Press “Return” to search
3. Press “Backspace” to return to contact book

**Limitations**

* The program is designed to handle contact information such as name and phone number. It may not support other types of contact data like email address or address.
* The program validates phone numbers based on the format of Polish phone numbers. It may not validate phone numbers from other countries correctly.
* The program uses a simple SQLite database for storing contacts. If you need to manage a large number of contacts or require advanced database features, you may need to use a different database system.
* The program provides basic functionality for managing contacts and may not have advanced features like contact groups or advanced search capabilities.

**Faced challenges**

During the development of the contact book program, several challenges were encountered, leading to valuable lessons learned. Here are some of the challenges and lessons learned during the development process:

Challenges:

1. **User Interface Design**: Designing an intuitive and user-friendly interface using Tkinter was challenging. It required careful consideration of layout, widget placement, and aesthetics to ensure a pleasant user experience.
2. **Data Validation**: Validating user input, particularly phone numbers, posed a challenge. Implementing a validation mechanism to ensure that phone numbers were in the correct format for Poland required regular expressions and careful handling of different number formats.
3. **Database Integration**: Integrating SQLite database functionality into the program was a challenge. Setting up the connection, creating the table, and performing database operations like inserting and retrieving contacts required careful implementation and error handling  
   .
4. **Data Persistence**: Achieving data persistence was another challenge faced during the development of the application. The task involved saving and loading contact data from a file while considering various file formats and handling edge cases. Ensuring the integrity of the data and implementing robust file handling mechanisms to handle different scenarios posed significant challenges.
5. **Testing and Bug Fixing:** Testing the application thoroughly and addressing bugs and issues was a continuous challenge throughout the development process. Rigorous testing was required to cover all possible use cases, edge cases, and user interactions. Identifying and fixing bugs promptly required diligent debugging and troubleshooting.

**Lessons learned**

1. **Modular Code Organization**: Breaking down the program into smaller, reusable modules improved code readability, maintainability, and reusability. It allowed for better separation of concerns and facilitated easier debugging and testing.
2. **User Input Validation**: Implementing robust validation mechanisms for user input is crucial. It helps prevent invalid data from being processed and ensures the integrity and consistency of the program's data. Regular expressions and proper error handling are essential tools for data validation.
3. **Error Handling and User Feedback**: Providing informative error messages and user feedback is vital for a good user experience. Clear and descriptive error messages help users understand what went wrong and how to correct it, improving the overall usability of the program.
4. **Version Control and Backup**: Utilizing version control systems, such as Git, and regularly creating backups of the codebase are essential practices. They enable easy rollbacks, collaborative development, and protect against data loss or accidental code changes.
5. **User Interface Testing**: Conducting thorough testing of the user interface ensures that the program behaves as expected and provides a smooth user experience. It is important to test various scenarios, handle edge cases, and consider different user interactions to identify and resolve any issues.
6. **Documentation and Code Comments**: Writing clear and comprehensive documentation, along with appropriate code comments, is beneficial for future reference and collaboration. It helps other developers understand the program's functionality, use the code effectively, and contribute to its improvement.
7. **Continuous Learning and Adaptability:** Embracing a growth mindset and a willingness to learn is essential in the fast-paced field of software development. Staying updated with the latest technologies, frameworks, and industry trends can help in adopting more efficient and effective development practices.

**Future Improvements**

Here are some possible improvements for the program:

* Enhance contact validation to support different phone number formats and handle international phone numbers.
* Implement additional contact fields such as email address, address, and notes.
* Improve the user interface by adding more features like contact sorting, filtering, and searching.
* Implement data backup and restore functionality to protect against data loss.
* Allow users to customize the contact book's appearance and layout.
* Implement user authentication and secure the contact data to ensure privacy.
* Provide options for importing and exporting contacts from popular contact management platforms or services.
* Support synchronization with cloud-based contact storage solutions.

These improvements would enhance the functionality and usability of the program, making it more versatile and user-friendly.