CONTACT BOOK

Python Project report

Piyush kumar singh 25BAI11043
Department of Computer Science
Engineering (AI & ML)
Vellore Institute of Technology
Bhopal, Madhya Pradesh
piyush.25bai11043@vitbhopal.ac.in
25th November 2025

***Abstract**:* this is simple Contact book management project which is to manage record of contact list using basic python programming. And it performs basic File Input output operations like add contact, search contact, Update contact , delete contact. And save this data to the text file contacts.txt.

• Introduction

A Contact book management project a tool designed to maintain contact information in an organized way. In daily life, remembering contacts manually or writing it notebooks, phone notes, or scattered text files and forgetting becomes difficult to find when needed. This project provides a simple command-line window that stores contact details such as name, phone and email using Python Programming..

• Objective

The main objectives of this project are:

- Manage contact data
- Understanding input output operations using Python
- Store data permanently in a text file
- Provide a simple and user-friendly menu-driven interface
- Reduce manual workload in handling contact records
- To create a small and easy-to-use digital tool for storing contact details.
- To help users quickly search for a contact when needed.
- To make it possible to update information without having to rewrite everything.
- To allow removal of old or unwanted contact entries.
- To provide a clean, beginner-friendly menu system.
- To follow modular coding practices using Python functions.
- To ensure contacts remain saved across multiple uses of the program.

• Implementation

The system works in a step-by-step process:

1. The program displays a menu with different options
2. The user chooses an operation (e.g., Add contact)
3. The system performs the selected operation
4. Contact information is stored in the text file contacts.txt
5. File handling is used to read, write, update, or delete contacts
6. The system continues to run until the user exits

Functional requirement

. Add Contact -> Users can enter a name, phone number, and email. The details are saved in a structured format in a text file.

. Search Contact -> Users can search by name or phone number. The search is case-insensitive and shows matching results.

. List All Contacts ->Displays all saved contacts in a neat and readable format. If no contacts exist, the user is informed.

. Update Contact -> Users can update any part of a contact. They can also keep existing values by leaving inputs blank.

.Delete Contact -> Allows removal of a specific contact. If the name doesn't exist, the system informs the user.

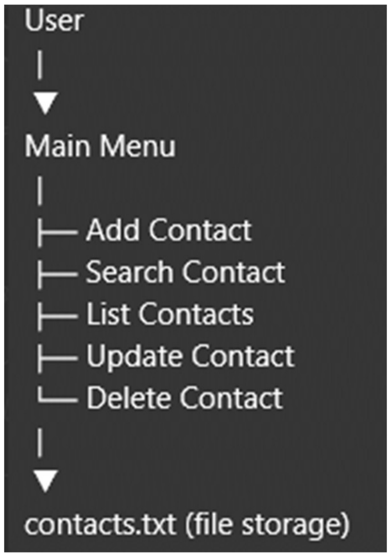
. Exit -> Closes the application safely.

Non Functional requirement

.Usability -> The system should be simple, clear, and easy for anyone to use.
.Reliability -> Data must be saved correctly and consistently, without errors.
. Maintainability -> The code should be modular, making it easy to add new features in the future.
.Performance -> The application should run quickly and not require much system memory.
. Portability -> The program should run on any operating system that supports Python.

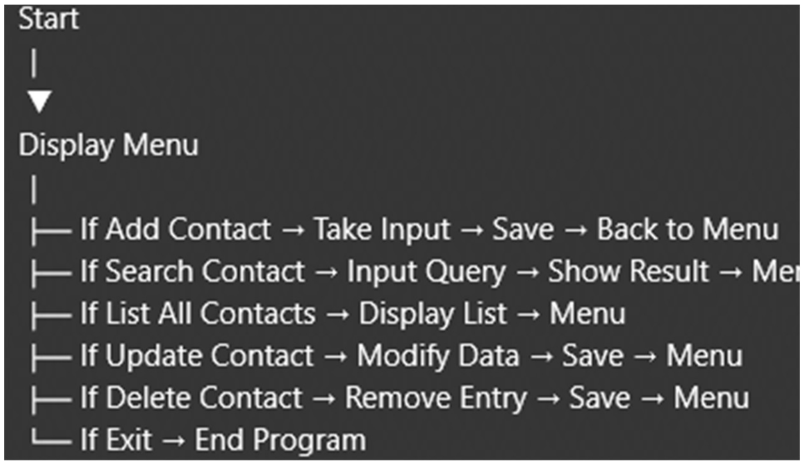
System architecture

The system follows a simple, modular architecture:



```
User
 |
 ▼
Main Menu
 |
 ├── Add Contact
 ├── Search Contact
 ├── List Contacts
 ├── Update Contact
 └── Delete Contact
 |
 ▼
contacts.txt (file storage)
```

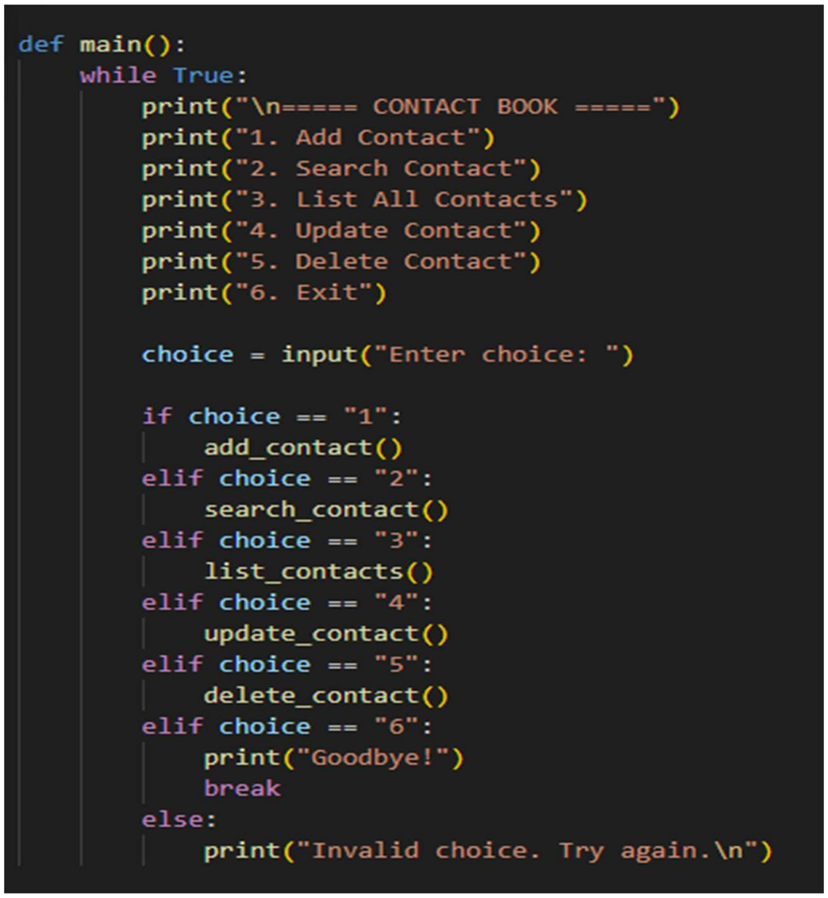
Workflow diagram



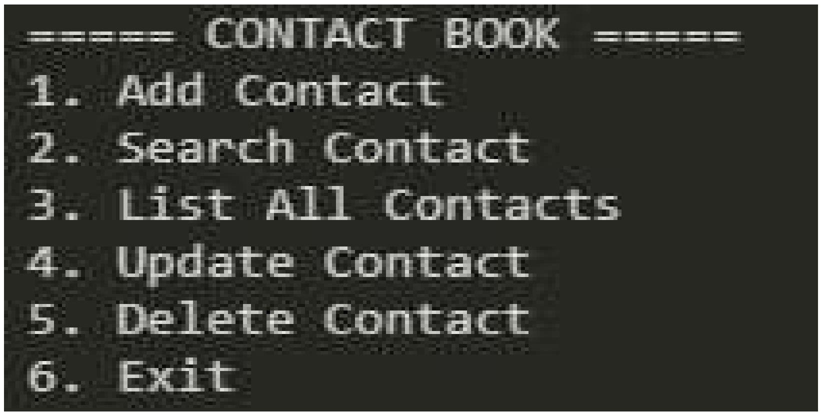
```
Start
 |
 ▼
Display Menu
 |
 ├── If Add Contact → Take Input → Save → Back to Menu
 ├── If Search Contact → Input Query → Show Result → Me
 ├── If List All Contacts → Display List → Menu
 ├── If Update Contact → Modify Data → Save → Menu
 ├── If Delete Contact → Remove Entry → Save → Menu
 └── If Exit → End Program
```

Working

- Main Menu

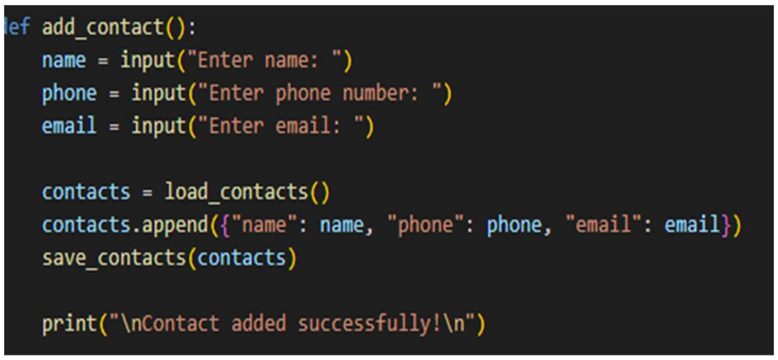


```python
def main():
    while True:
        print("\n===== CONTACT BOOK =====")
        print("1. Add Contact")
        print("2. Search Contact")
        print("3. List All Contacts")
        print("4. Update Contact")
        print("5. Delete Contact")
        print("6. Exit")

        choice = input("Enter choice: ")

        if choice == "1":
            add_contact()
        elif choice == "2":
            search_contact()
        elif choice == "3":
            list_contacts()
        elif choice == "4":
            update_contact()
        elif choice == "5":
            delete_contact()
        elif choice == "6":
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Try again.\n")
```

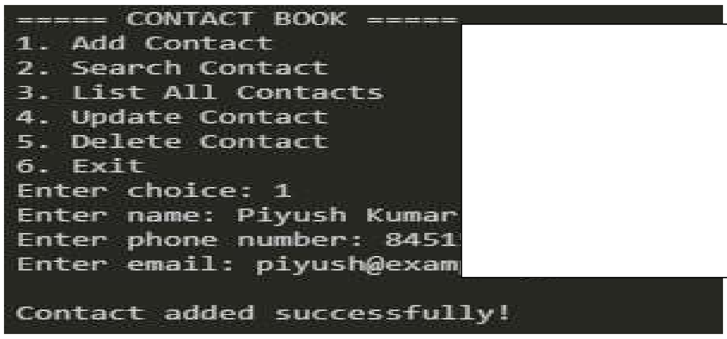


```
===== CONTACT BOOK =====
1. Add Contact
2. Search Contact
3. List All Contacts
4. Update Contact
5. Delete Contact
6. Exit
```

- Add new contacts to contact book
    - This adds new contacts

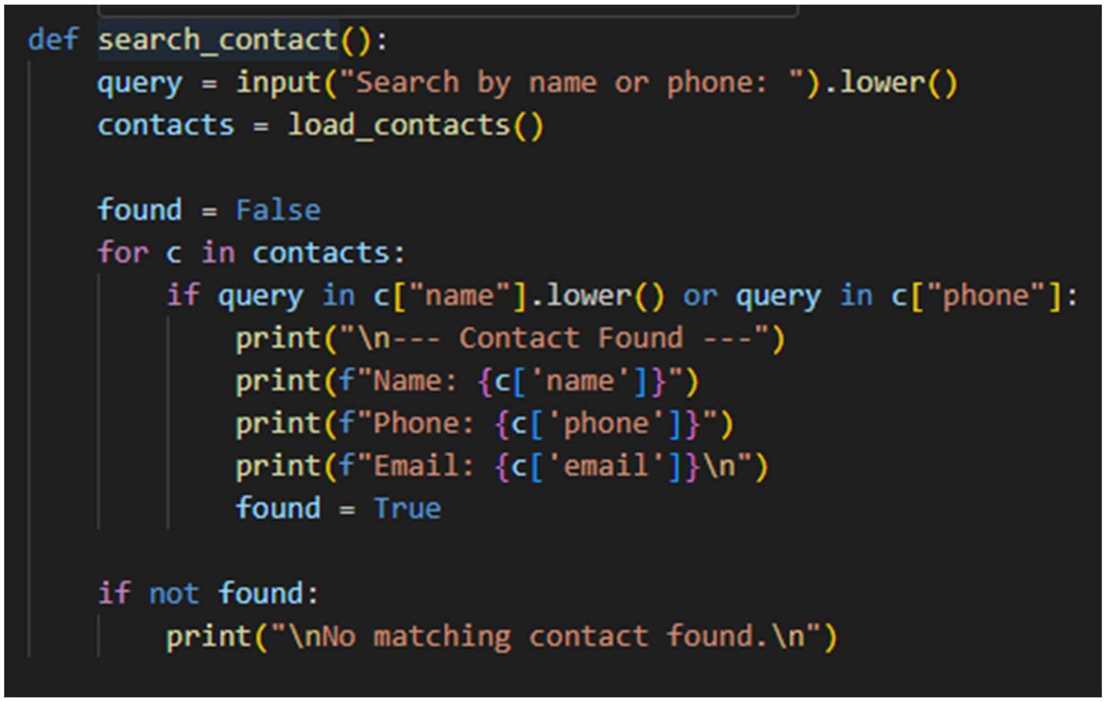


```
ef add_contact():
    name = input("Enter name: ")
    phone = input("Enter phone number: ")
    email = input("Enter email: ")

    contacts = load_contacts()
    contacts.append({"name": name, "phone": phone, "email": email})
    save_contacts(contacts)

    print("\nContact added successfully!\n")
```

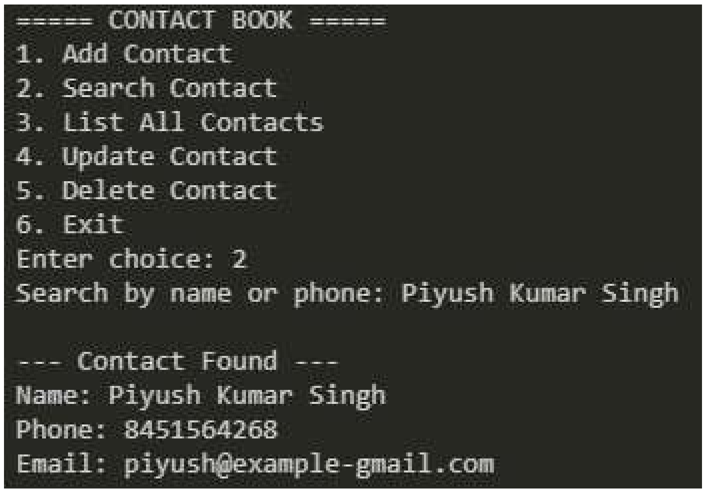


```
===== CONTACT BOOK =====
1. Add Contact
2. Search Contact
3. List All Contacts
4. Update Contact
5. Delete Contact
6. Exit
Enter choice: 1
Enter name: Piyush Kumar
Enter phone number: 8451 [redacted]
Enter email: piyush@exam

Contact added successfully!
```

- Search for contacts
    - This Function Search for contact

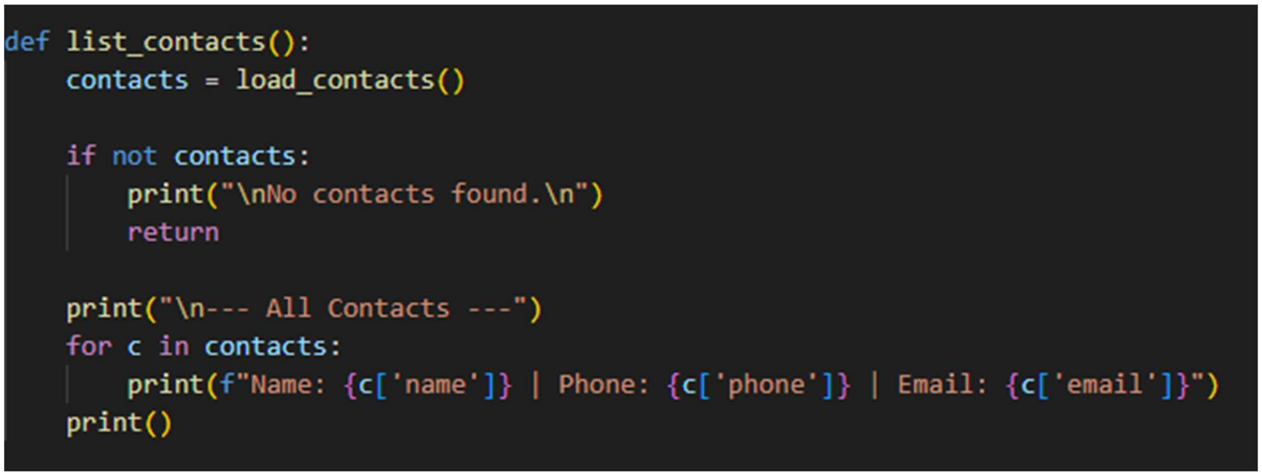


```
def search_contact():
    query = input("Search by name or phone: ").lower()
    contacts = load_contacts()

    found = False
    for c in contacts:
        if query in c["name"].lower() or query in c["phone"]:
            print("\n--- Contact Found ---")
            print(f"Name: {c['name']}")
            print(f"Phone: {c['phone']}")
            print(f"Email: {c['email']}\n")
            found = True

    if not found:
        print("\nNo matching contact found.\n")
```

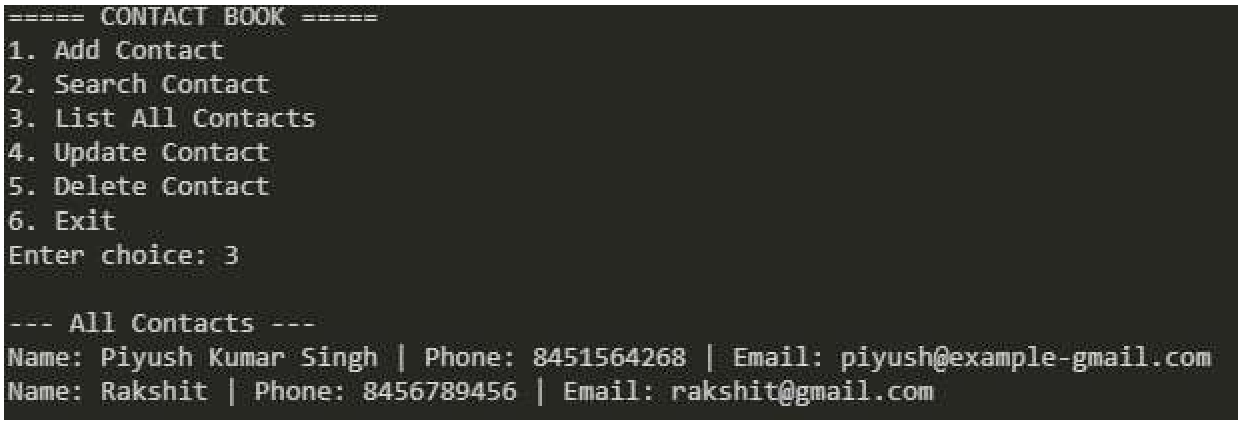


```
===== CONTACT BOOK =====
1. Add Contact
2. Search Contact
3. List All Contacts
4. Update Contact
5. Delete Contact
6. Exit
Enter choice: 2
Search by name or phone: Piyush Kumar Singh

--- Contact Found ---
Name: Piyush Kumar Singh
Phone: 8451564268
Email: piyush@example-gmail.com
```

- List all contacts
    - This function lists the contacts



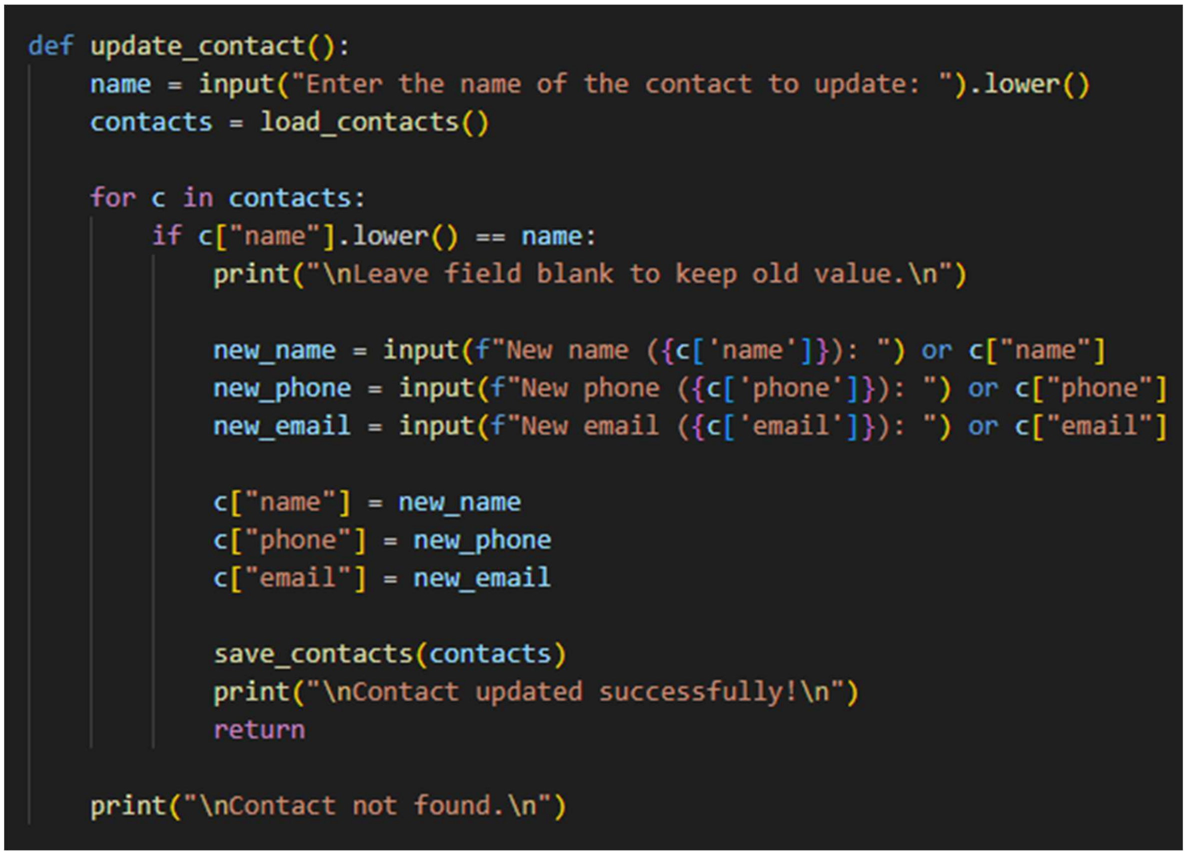
```python
def list_contacts():
    contacts = load_contacts()

    if not contacts:
        print("\nNo contacts found.\n")
        return

    print("\n--- All Contacts ---")
    for c in contacts:
        print(f"Name: {c['name']} | Phone: {c['phone']} | Email: {c['email']}")
    print()
```



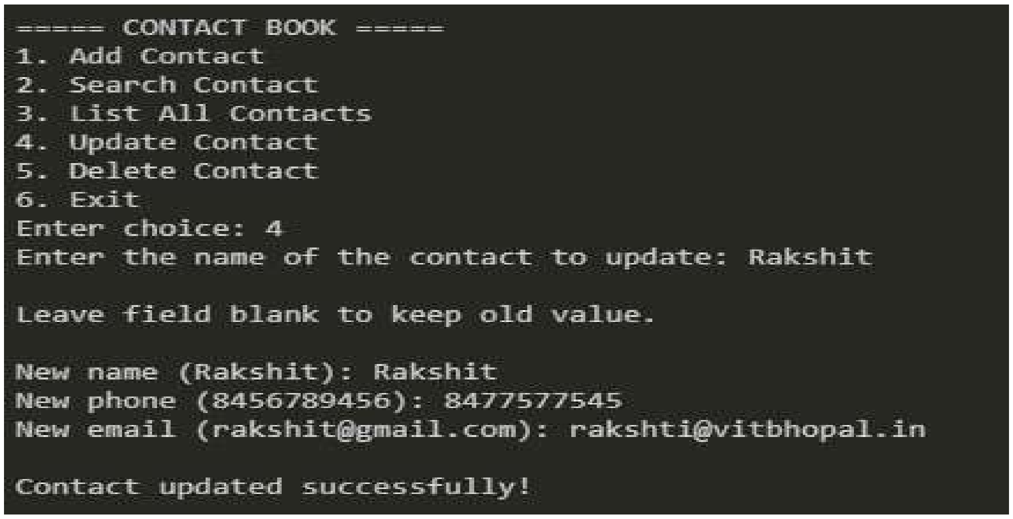
```
===== CONTACT BOOK =====
1. Add Contact
2. Search Contact
3. List All Contacts
4. Update Contact
5. Delete Contact
6. Exit
Enter choice: 3

--- All Contacts ---
Name: Piyush Kumar Singh | Phone: 8451564268 | Email: piyush@example-gmail.com
Name: Rakshit | Phone: 8456789456 | Email: rakshit@gmail.com
```

Update contact record

- This function update the record of contact



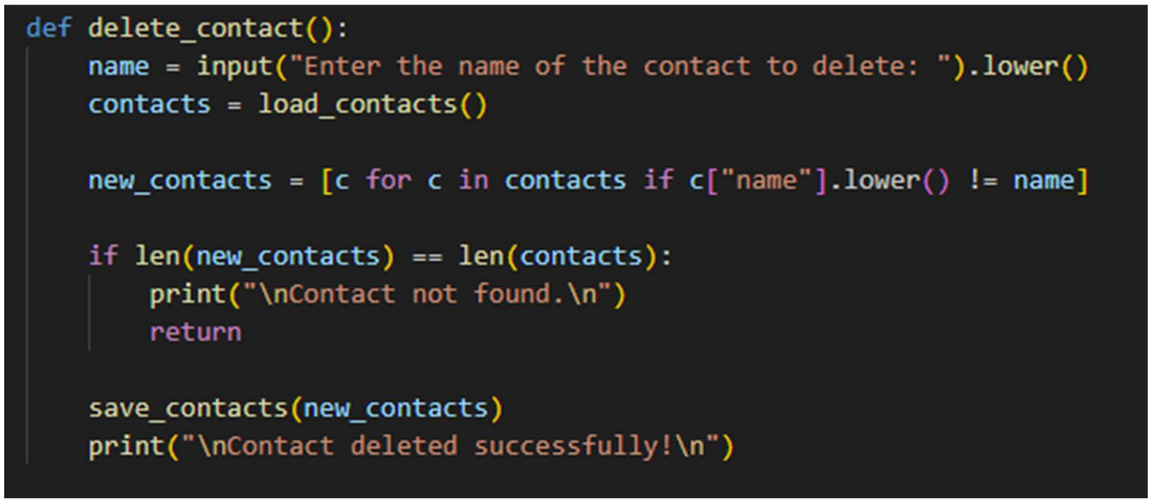
```python
def update_contact():
    name = input("Enter the name of the contact to update: ").lower()
    contacts = load_contacts()

    for c in contacts:
        if c["name"].lower() == name:
            print("\nLeave field blank to keep old value.\n")

            new_name = input(f"New name ({c['name']}): ") or c["name"]
            new_phone = input(f"New phone ({c['phone']}): ") or c["phone"]
            new_email = input(f"New email ({c['email']}): ") or c["email"]

            c["name"] = new_name
            c["phone"] = new_phone
            c["email"] = new_email

            save_contacts(contacts)
            print("\nContact updated successfully!\n")
            return

    print("\nContact not found.\n")
```



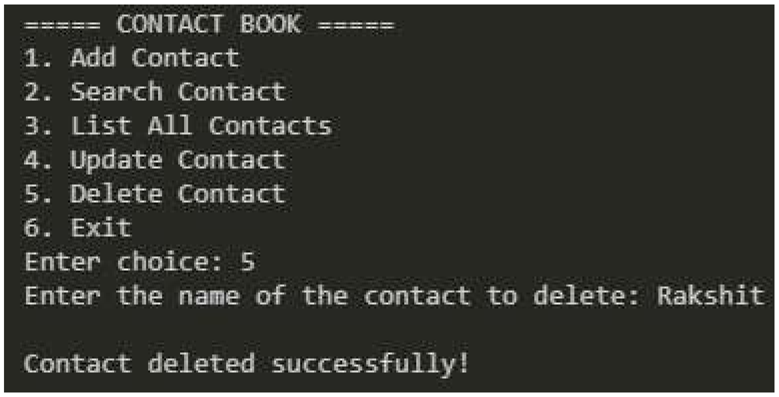
```
===== CONTACT BOOK =====
1. Add Contact
2. Search Contact
3. List All Contacts
4. Update Contact
5. Delete Contact
6. Exit
Enter choice: 4
Enter the name of the contact to update: Rakshit

Leave field blank to keep old value.

New name (Rakshit): Rakshit
New phone (8456789456): 8477577545
New email (rakshit@gmail.com): rakshti@vitbhopal.in

Contact updated successfully!
```

Delete contact record

- This function delete the record of contact



```python
def delete_contact():
    name = input("Enter the name of the contact to delete: ").lower()
    contacts = load_contacts()

    new_contacts = [c for c in contacts if c["name"].lower() != name]

    if len(new_contacts) == len(contacts):
        print("\nContact not found.\n")
        return

    save_contacts(new_contacts)
    print("\nContact deleted successfully!\n")
```



```
===== CONTACT BOOK =====
1. Add Contact
2. Search Contact
3. List All Contacts
4. Update Contact
5. Delete Contact
6. Exit
Enter choice: 5
Enter the name of the contact to delete: Rakshit

Contact deleted successfully!
```

• Implementation Details

The application is built using Python 3 and uses:

Basic input/output functions

Lists and dictionaries to store contact information temporarily

The os module to check for file existence

File reading/writing for persistent storage

A loop-based menu system to manage user actions

All core features are implemented inside individual functions such as:

add_contact()

search_contact()

update_contact()

delete_contact()

list_contacts()

load_contacts()

save_contacts()

This modular approach makes the system easy to understand and modify.

• Code Summary (High-Level)

Contacts are stored in contacts.txt in the format:

Name|Phone|Email

load_contacts() reads the file and loads contacts into a list.

save_contacts() writes updated contacts to the file.

Each menu option is mapped to a specific function.

The main() function controls overall execution.

• Testing

The application was manually tested using various cases:

Test Cases

Adding multiple contacts

Searching existing and non-existing contacts

Updating fields partially and fully

Deleting contacts

Listing contacts when file is empty

Verifying file persistence after restarting the program

All features worked as expected, and no major issues were encountered.

• Challenges Faced

Ensuring that updates didn’t overwrite unchanged data.

Maintaining correct formatting when reading/writing the text file.

Handling cases where a user tries to update or delete a non-existing contact.

Avoiding duplicate entries during accidental repeat additions.

Learnings & Key Takeaways

Through this project, the following concepts were strengthened:

Python file handling

Modular programming

Data storage techniques

Designing menu-driven systems

Debugging errors and validating user input

This project also helped build confidence in structuring programs cleanly.

• Future Enhancements

Add data validation for correct email and phone formats.

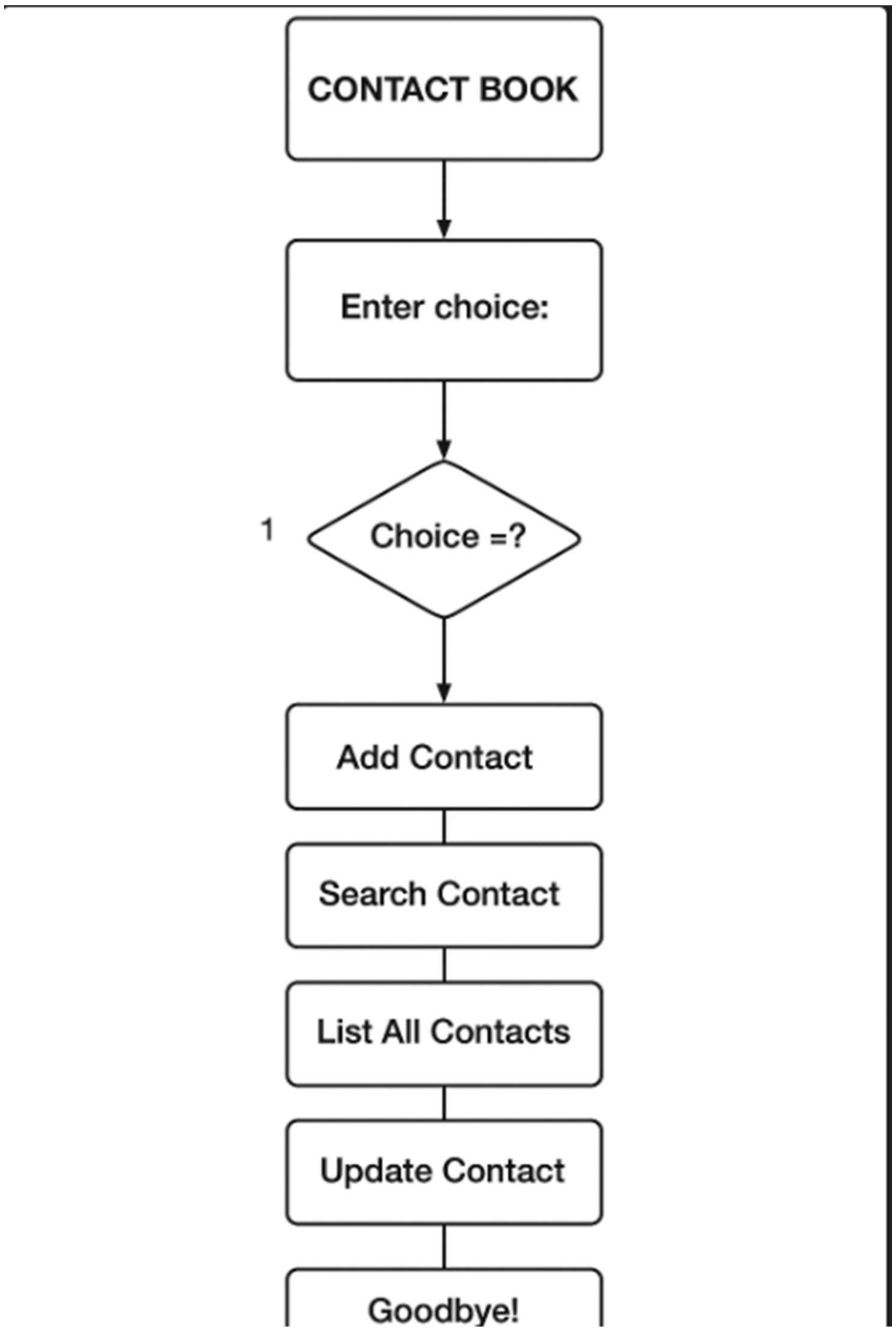
Add separate categories for contacts (personal, academic, professional).

Integrate with a database for more secure storage.

Add GUI support using Tkinter or create a web-based version.

Export contacts to CSV/Excel formats.

- **FlowChart**


CONTACT BOOK
Enter choice:
Choice =?
1
Add Contact
Search Contact
List All Contacts
Update Contact
Goodbye!

• Conclusions

The Contact Book Application successfully provides a simple and effective solution for storing and managing contact information. It meets all the functional requirements and demonstrates practical implementation of Python basics. The system is easy to use, well-structured, and can be expanded with more advanced features in the future.

Reference

- Python Documentation
- Class Notes
- Online Tutorials
- VITyarthi Project Instructions PDF