K. R. MANGALAM UNIVERSITY

ASSIGNMENT 2

Course Name: Programming for Problem Solving Using Python

Assignment Title: Contact Book

Student Name: Tushar Singh

Roll No.: 2501940055

Program: MCA (AI & ML)

Semester: I

Session: 2025-26

Faculty Name: Ms. Neha Kaushik

Date of Submission: 19/11/2025

Department: SOET	**Session:** 2025-26
Program: MCA (AI/ML)	**Semester: 1**
Course Code: ETCCPP171	**College Roll no:** 2501940055
Course Name: Programming for Problem Solving Using Python	
Submitted by: Tushar Singh	**Faculty:** Ms. Neha Kaushik

GitHub Link :- https://github.com/singhtushar738-prog/Contact-Book-Assignment.git

Name: Tushar Singh
Date: 19/11/2025
Assignment: Contact Book

Code:

```
import csv
import json
import os
from datetime import datetime

CSV_PATH = "contact_list.csv"
JSON_PATH = "contact_list.json"
LOG_PATH = "error_log.txt"
CSV_COLUMNS = ["name", "phone", "email"]

def log_err(message, operation):
    """Append an error message with timestamp and operation to the log file."""
    try:
        with open(LOG_PATH, "a", encoding="utf-8") as log:
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            log.write(f"[{timestamp}] {operation}: {message}\n")
    except Exception:
        # if logging fails we silently ignore to avoid crashing the program
        pass

def ensure_csv():
    """Create CSV file with header row if it doesn't exist."""
    if not os.path.exists(CSV_PATH):
        try:
            with open(CSV_PATH, "w", newline="", encoding="utf-8") as f:
                writer = csv.DictWriter(f, fieldnames=CSV_COLUMNS)
                writer.writeheader()
        except Exception as e:
            log_err(str(e), "Ensure CSV")

def load_contacts():
    """Return list of contact_list as dicts. If file missing or empty returns
empty list."""
    contact_list = []
    try:
        with open(CSV_PATH, "r", newline="", encoding="utf-8") as f:
            reader = csv.DictReader(f)
            for row in reader:
                # normalize keys and ensure fields exist

                contact_list.append({
                    "name": (row.get("name") or "").strip(),
                    "phone": (row.get("phone") or "").strip(),
                    "email": (row.get("email") or "").strip(),
```

```
                })
    except FileNotFoundError:
        # handled by caller or will be created
        pass
    except Exception as e:
        log_err(str(e), "Read Contacts")
    return contact_list
def save_contacts(contact_list):
    """Overwrite CSV with contact_list (list of dicts)."""
    try:
        with open(CSV_PATH, "w", newline="", encoding="utf-8") as f:
            writer = csv.DictWriter(f, fieldnames=CSV_COLUMNS)
            writer.writeheader()
            for c in contact_list:
                writer.writerow({k: c.get(k, "") for k in CSV_COLUMNS})
    except Exception as e:
        log_err(str(e), "Write Contacts")
        raise
def create_contact():
    try:
        name = input("Enter Name: ").strip()
        if not name:
            print("Name cannot be empty. Aborting add.\n")
            return
        phone = input("Enter Phone Number: ").strip()
        email = input("Enter Email Address: ").strip()
        contact_list = load_contacts()

        # Prevent exact duplicate by name+phone
        for c in contact_list:
            if c["name"].lower() == name.lower() and c["phone"] == phone:
                print("A contact with the same name and phone already exists.\n")
                return
        contact_list.append({"name": name, "phone": phone, "email": email})
        save_contacts(contact_list)
        print("Contact added successfully.\n")

    except Exception as e:
        log_err(str(e), "Add Contact")
        print("Error while adding contact.\n")

def show_contacts():
    try:
        contact_list = load_contacts()
        if not contact_list:
            print("No contact_list found.\n")
            return

        # nice table-like printing with index
        print("\nIndex | Name                     | Phone           | Email")
        print("-" * 70)
        for i, c in enumerate(contact_list, start=1):
            print(f"{i:5d} | {c['name'][:24]:24} | {c['phone'][:15]:15} |
{c['email']}")
        print()

    except Exception as e:
        log_err(str(e), "Display Contacts")
        print("Error displaying contact_list.\n")


def find_contact():
```

```
    query = input("Enter name or phone to search (partial allowed):
").strip().lower()
    if not query:
        print("Empty query.\n")
        return

    try:
        contact_list = load_contacts()
        results = [c for c in contact_list if query in c["name"].lower() or query
in c["phone"].lower()]

        if results:
            print(f"\nFound {len(results)} matching contact(s):")
            print("-" * 50)
            for c in results:
                print(f"Name: {c['name']} | Phone: {c['phone']} | Email:
{c['email']}")
            print()
        else:
            print("Contact not found.\n")

    except Exception as e:
        log_err(str(e), "Search Contact")
        print("Error searching contact.\n")

def modify_contact():
    try:
        contact_list = load_contacts()
        if not contact_list:
            print("No contact_list to update.\n")
            return

        show_contacts()
        choice = input("Enter the Index of the contact to update (or name):
").strip()

        # find contact by index or name
        contact = None
        if choice.isdigit():
            idx = int(choice) - 1
            if 0 <= idx < len(contact_list):
                contact = contact_list[idx]
        else:
            for c in contact_list:
                if c["name"].lower() == choice.lower():
                    contact = c
                    break

        if not contact:
            print("Contact not found.\n")
            return

        print("Selected:")
        print(f"Name: {contact['name']} | Phone: {contact['phone']} | Email:
{contact['email']}")
        print("What do you want to update?")
        print("1. Name")
        print("2. Phone")
        print("3. Email")
        print("4. Cancel")
        opt = input("Enter choice: ").strip()
        if opt == "1":
```

```
            new = input("Enter new name: ").strip()
            if new:
                contact["name"] = new
        elif opt == "2":
            new = input("Enter new phone: ").strip()
            if new:
                contact["phone"] = new
        elif opt == "3":
            new = input("Enter new email: ").strip()
            if new:
                contact["email"] = new
        else:
            print("Update cancelled.\n")
            return

        save_contacts(contact_list)
        print("Contact updated successfully.\n")
    except Exception as e:
        log_err(str(e), "Update Contact")
        print("Error updating contact.\n")

def remove_contact():
    try:
        contact_list = load_contacts()
        if not contact_list:
            print("No contact_list to delete.\n")
            return
        show_contacts()
        choice = input("Enter the Index of the contact to delete (or exact name):
").strip()

        # find index

        del_idx = None
        if choice.isdigit():
            idx = int(choice) - 1
            if 0 <= idx < len(contact_list):
                del_idx = idx
        else:
            for i, c in enumerate(contact_list):
                if c["name"].lower() == choice.lower():
                    del_idx = i
                    break

        if del_idx is None:
            print("Contact not found.\n")
            return
        confirm = input(f"Are you sure you want to delete
'{contact_list[del_idx]['name']}'? (y/N): ").strip().lower()
        if confirm != "y":
            print("Delete cancelled.\n")
            return
        removed_contact = contact_list.pop(del_idx)
        save_contacts(contact_list)
        print(f"Deleted contact: {removed_contact['name']}\n")

    except Exception as e:
        log_err(str(e), "Delete Contact")
        print("Error deleting contact.\n")

def export_json():
    try:
```

```
        contact_list = load_contacts()
        with open(JSON_PATH, "w", encoding="utf-8") as file:
            json.dump(contact_list, file, indent=4, ensure_ascii=False)
        print("Contacts exported to JSON successfully.\n")
    except Exception as e:
        log_err(str(e), "Export JSON")
        print("Error exporting to JSON.\n")

def import_json():
    try:
        if not os.path.exists(JSON_PATH):
            print("JSON file not found.\n")
            return
        with open(JSON_PATH, "r", encoding="utf-8") as file:
            contact_list = json.load(file)

        if not contact_list:
            print("No contact_list in JSON file.\n")
            return

        print("\nContacts from JSON:")
        print("-" * 50)
        for c in contact_list:
            print(f"Name: {c.get('name','')} | Phone: {c.get('phone','')} | Email:
{c.get('email','')}")
        print()

    except Exception as e:
        log_err(str(e), "Load JSON")
        print("Error loading JSON.\n")

def menu_main():
    ensure_csv()
    print("Welcome to the Contact Book Manager!")

    while True:
        print("1. Add Contact")
        print("2. View Contacts")
        print("3. Search Contact")
        print("4. Update Contact")
        print("5. Delete Contact")
        print("6. Export to JSON")
        print("7. Load from JSON")
        print("8. Exit")

choice = input("Enter your choice: ").strip()

        if choice == "1":
            create_contact()
        elif choice == "2":
            show_contacts()
        elif choice == "3":
            find_contact()
        elif choice == "4":
            modify_contact()
        elif choice == "5":
            remove_contact()
        elif choice == "6":
            export_json()
        elif choice == "7":
            import_json()
        elif choice == "8":
```

```
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Try again.\n")

if __name__ == "__main__":
    menu_main()
```

Output:

```
Welcome to the Contact Book Manager!
1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 1
Enter Name: Tushar Sharma
Enter Phone Number: 9941545
Enter Email Address: hsbdyfbsegj@gmail.com
Contact added successfully.

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 7
JSON file not found.

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 6
Contacts exported to JSON successfully.

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 7

Contacts from JSON:
--------------------------------------------------
Name: Tushar Sharma | Phone: 9941545 | Email: hsbdyfbsegj@gmail.com
```

```
Enter your choice: 1
Enter Name: Vinayak Sharma
Enter Phone Number: 9595952141
Enter Email Address: vinayakshar@gmail.com
Contact added successfully.

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 7

Contacts from JSON:
------------------------------------------------
Name: Tushar Sharma | Phone: 9941545 | Email: hsbdyfbsegj@gmail.com

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 6
Contacts exported to JSON successfully.

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 7

Contacts from JSON:
------------------------------------------------
Name: Tushar Sharma | Phone: 9941545 | Email: hsbdyfbsegj@gmail.com
Name: Vinayak Sharma | Phone: 9595952141 | Email: vinayakshar@gmail.com

1. Add Contact
2. View Contacts
3. Search Contact
4. Update Contact
5. Delete Contact
6. Export to JSON
7. Load from JSON
8. Exit
Enter your choice: 8
```