CONTACT MANAGER SYSTEM

PYTHON 3

Project Title: Contact Manager System (CSV Based)

Language: Python 3

Storage: CSV (Comma Separated Values)

Submitted By: Guggilam Bala Yoshik

Date: November 2025

INTRODUCTION

The Contact Manager System is a lightweight software application designed to store, retrieve, and manage contact information efficiently. In an era where digital communication is paramount, managing a personal or professional directory is a fundamental necessity. This project utilizes Python to create a Command Line Interface (CLI) tool that persists data using a CSV file. It allows users to perform CRUD (Create, Read, Update, Delete) operations on contact details, ensuring data is saved even after the program terminates.

PROBLEM STATEMENT

Traditional paper-based address books are prone to damage and loss, while complex database software can be resource-heavy and difficult to navigate for simple tasks. Users need a lightweight, portable, and fast solution to manage names and phone numbers without requiring internet connectivity or expensive software licenses. This project addresses the need for a simple, persistent digital address book that can be run on any standard computer.

FUNCTIONAL REQUIREMENTS

The system implements the following functionalities:

1. **Data Persistence:** Automatically load existing contacts from contacts.csv on startup and save changes immediately upon modification.
2. **Add Contact:** Allow users to input a name and phone number. The system sanitizes the name (removes spaces, converts to lowercase) for consistent storage.
3. **Search Contact:** Retrieve a phone number by entering the contact's name.
4. **Delete Contact:** Remove a specific contact from the database permanently.
5. **View All:** Display a sorted list of all contacts currently stored in the system.
6. **Default Initialization:** If no database exists, the system automatically creates one with a predefined set of sample contacts.

NON-FUNCTIONAL REQUIREMENTS

1. **Performance:** The system provides near-instantaneous search results using Hash Map (Dictionary) implementation (Time Complexity: $O(1)$).
2. **Usability:** A clear, menu-driven Command Line Interface that is easy to understand.
3. **Reliability:** Handles file input/output operations safely, preventing data corruption if the file is missing.

4. **Portability:** The application runs on any machine with Python installed and uses standard CSV files compatible with Excel.

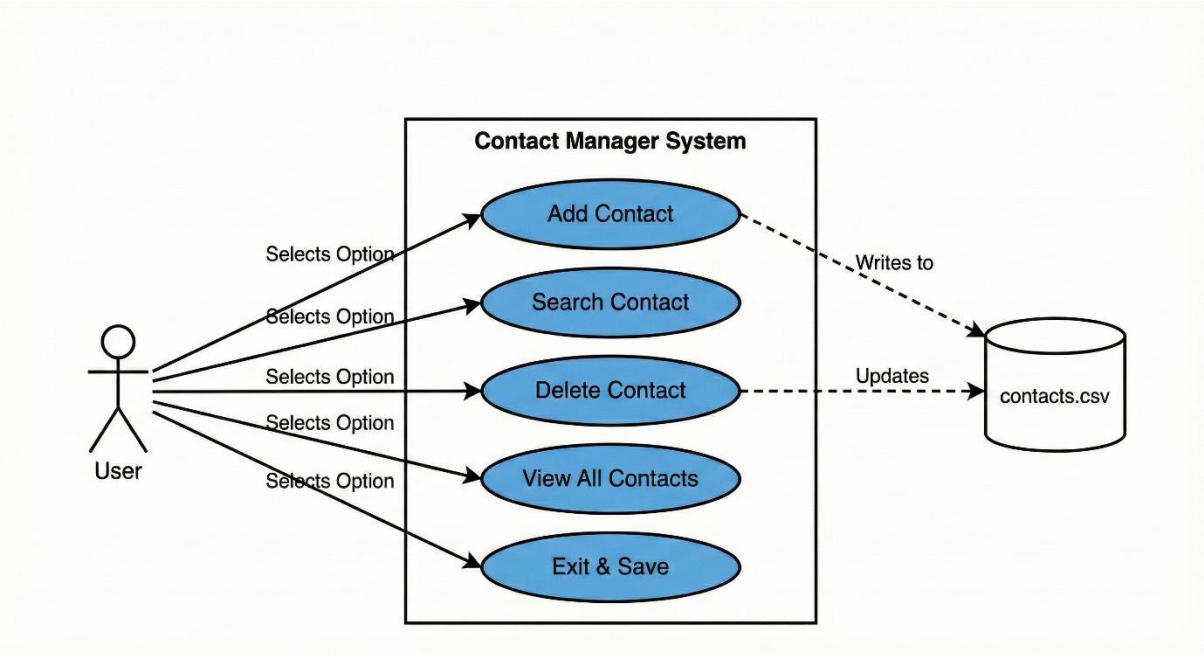
SYSTEM ARCHITECTURE

The system follows a modular architecture:

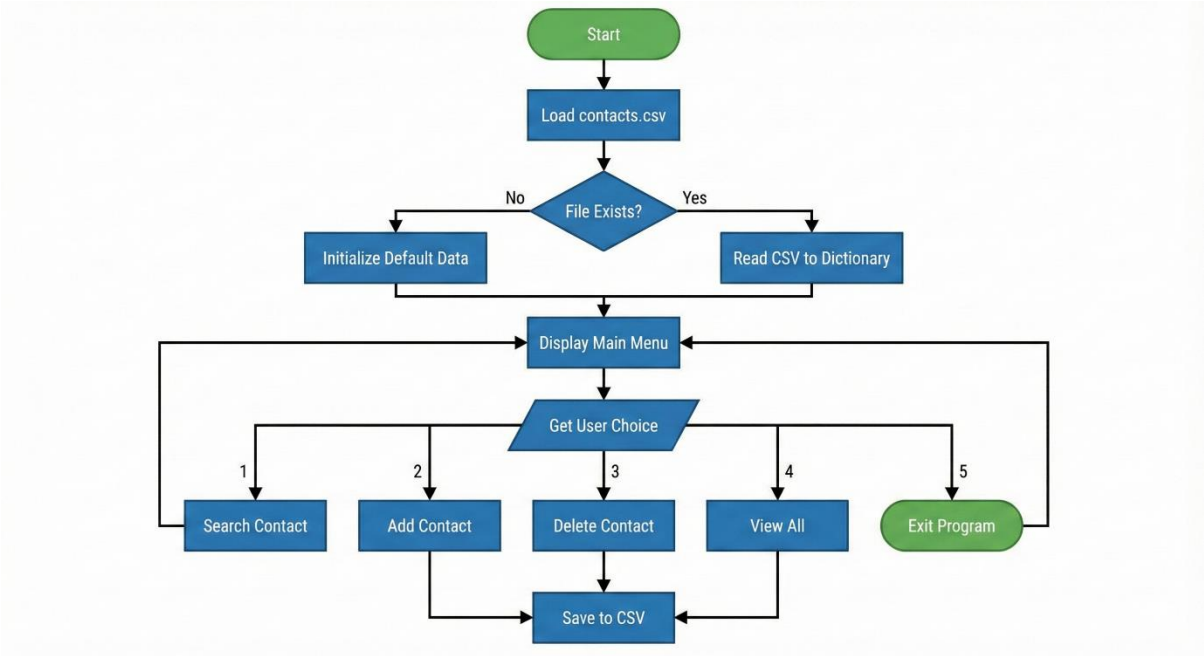
- **Presentation Layer:** The $main()$ loop handles user input and displays outputs.
- **Logic Layer:** Functions like add_contact, Contacts, and delete_contact process the data.
- **Data Layer:** The csv module manages the physical storage of data in contacts.csv.

DESIGN DIAGRAMS

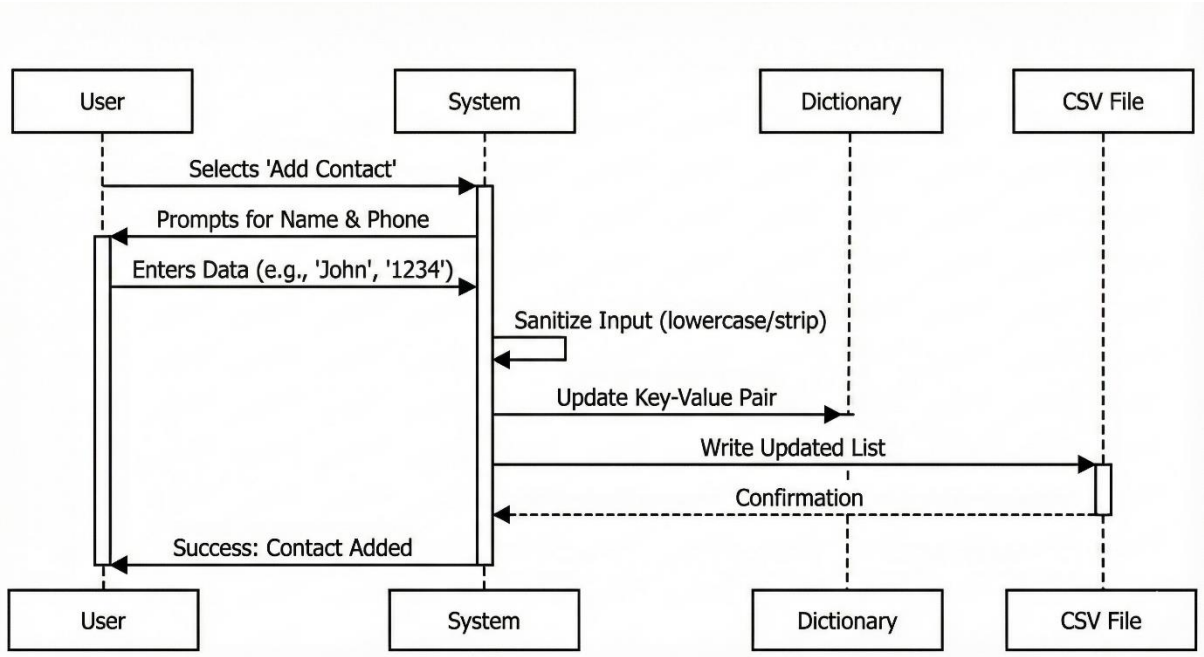
Case Diagram



Flowchart



Sequence Diagram



DESIGN DECISIONS & RATIONALE

- **Use of Python Dictionary:** A Python dictionary was chosen as the runtime data structure because it offers $O(1)$ average time complexity for lookups, making search operations extremely fast compared to iterating through lists $(O(n))$.
- **Use of CSV for Storage:** CSV was selected over SQL databases (like SQLite) for simplicity and portability. CSV files are human-readable and can be opened in Microsoft Excel or Google Sheets without requiring specialized database viewers.
- **Data Sanitization:** Names are stored in lowercase with whitespace stripped. This decision prevents duplicate entries caused by capitalization errors (e.g., "Sundar" vs "sundar").

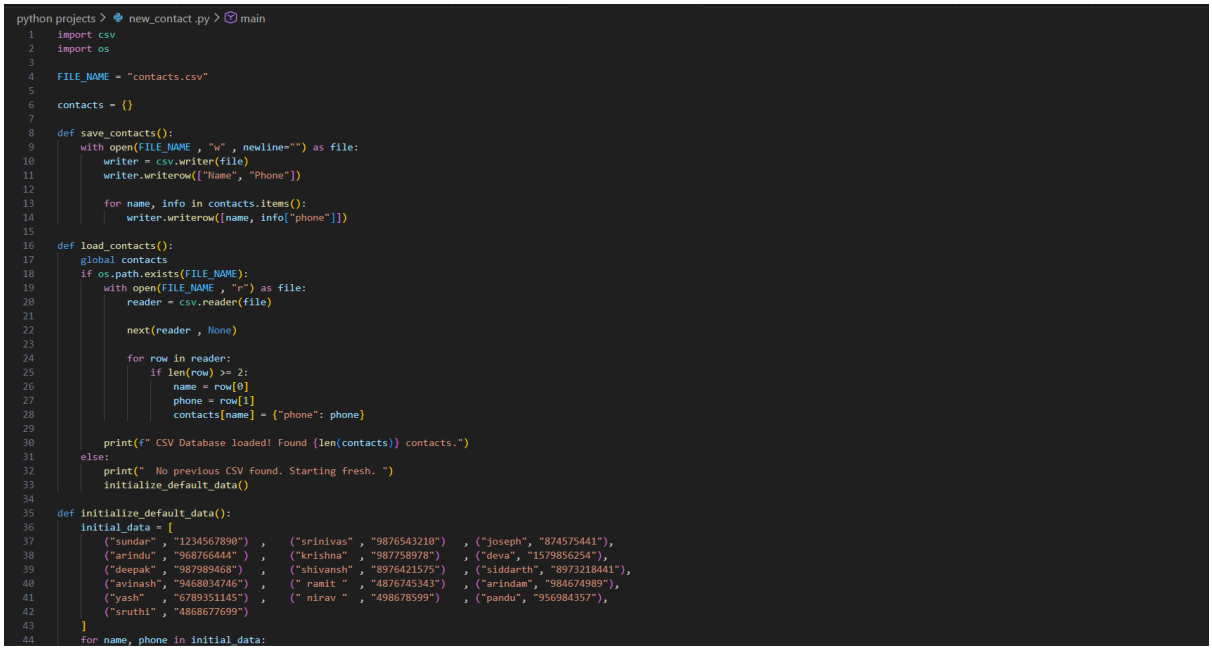
IMPLEMENTATION DETAILS

The core logic is implemented in main.py. Below is the source code used for the project:

import csv import os FILE_NAME = "contacts.csv" contacts = {} def save_contacts(): with open(FILE_NAME, "w", newline="") as file: writer = csv.writer(file) writer.writerow(["Name", "Phone"]) for name, info in contacts.items(): writer.writerow([name, info["phone"]]) def load_contacts(): global contacts if os.path.exists(FILE_NAME): with open(FILE_NAME, "r") as file: reader = csv.reader(file) next(reader, None) # Skip header for row in reader: if len(row) >= 2: contacts[row[0]] = {"phone": row[1]} print(f" CSV Database loaded! Found {len(contacts)} contacts.") else: print(" No previous CSV found. Starting fresh.") initialize_default_data() def initialize_default_data(): # ... (Default data initialization logic) save_contacts() def add_contact(name, phone, save=True): clean_name = name.strip().lower() contacts[clean_name] = {"phone": phone} if save: save_contacts() # ... (Rest of the CRUD functions: delete, search, show_all) def main(): load_contacts() while True: # ... (Menu loop logic) pass if __name__ == "__main__": main()

SCREENSHOTS / RESULTS

1. **Main Menu:** Shows the interactive console with 5 options.



```
python projects > new_contact.py > main
import csv
import os

FILE_NAME = "contacts.csv"

contacts = {}

def save_contacts():
    with open(FILE_NAME , "w" , newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["Name", "Phone"])

        for name, info in contacts.items():
            writer.writerow([name, info["phone"]])

def load_contacts():
    global contacts
    if os.path.exists(FILE_NAME):
        with open(FILE_NAME , "r") as file:
            reader = csv.reader(file)

            next(reader , None)

            for row in reader:
                if len(row) >= 2:
                    name = row[0]
                    phone = row[1]
                    contacts[name] = {"phone": phone}

        print(f" CSV Database loaded! Found {len(contacts)} contacts.")
    else:
        print("  No previous CSV found. Starting fresh. ")
        initialize_default_data()

def initialize_default_data():
    initial_data = [
        ("sundar" , "1234567890")  ,    ("srinivas" , "9876543210")   , ("joseph", "874575441"),
        ("arindu" , "968766444" )  ,    ("krishna"  , "987758978")    , ("deva", "1579856254"),
        ("deepak" , "987989468")   ,    ("shivansh" , "8976421575")   , ("siddarth", "8973218441"),
        ("avinash", "9468034746")  ,    (" ramit "  , "4876745343")   , ("arindam", "984674989"),
        ("yash"   , "6789351145")  ,    (" nirav "  , "498678599")    , ("pandu", "956984357"),
        ("sruthi" , "4868677699")
    ]
    for name, phone in initial_data:
```

```
def initialize_default_data():
    initial_data = [
        ("sundar" , "1234567890")  ,    ("srinivas" , "9876543210")   , ("joseph", "874575441"),
        ("arindu" , "968766444" )  ,    ("krishna"  , "987758978")    , ("deva", "1579856254"),
        ("deepak" , "987989468")   ,    ("shivansh" , "8976421575")   , ("siddarth", "8973218441"),
        ("avinash", "9468034746")  ,    (" ramit "  , "4876745343")   , ("arindam", "984674989"),
        ("yash"   , "6789351145")  ,    (" nirav "  , "498678599")    , ("pandu", "956984357"),
        ("sruthi" , "4868677699")
    ]
    for name, phone in initial_data:
        add_contact (name, phone , save=False)
    save_contacts()

def add_contact (name, phone , save=True):
    clean_name = name.strip() .lower()
    contacts[clean_name] = {"phone": phone}
    print (f"  Success: Contact '{name}' added/updated. ")
    if save:
        save_contacts()

def delete_contact(name):
    clean_name = name.strip().lower()
    if clean_name in contacts:
        del contacts [clean_name]
        save_contacts()                (parameter) name: Any
        print(f" Deleted: Contact '{name}' removed.")
    else:
        print(f" Error: Contact '{name}' not found.")

def search_contact(name):
    clean_name = name.strip().lower()
    if clean_name in contacts:
        phone = contacts[clean_name]["phone"]
        print(f"   Found: {name.title()} ->   {phone}")
    else:
        print(f"  Contact  '{name}'  not found.")

def show_all_contacts():
    print("\n---  ----- Contact List (CSV)  ---- ---")
    if not contacts:
        print("No contacts available.")
    else:
        for name in sorted(contacts.keys()):
```

2. Search Result: Shows successful retrieval of a phone number.

```
def initialize_default_data():
    initial_data = [
        ("sundar" , "1234567890")  ,    ("srinivas" , "9876543210")   , ("joseph", "874575441"),
        ("arindu" , "968766444" )  ,    ("krishna"  , "987758978")    , ("deva", "1579856254"),
        ("deepak" , "987989468")   ,    ("shivansh" , "8976421575")   , ("siddarth", "8973218441"),
        ("avinash", "9468034746")  ,    (" ramit "  , "4876745343")   , ("arindam", "984674989"),
        ("yash"   , "6789351145")  ,    (" nirav "  , "498678599")    , ("pandu", "956984357"),
        ("sruthi" , "4868677699")
    ]
    for name, phone in initial_data:
        add_contact (name, phone , save=False)
    save_contacts()

def add_contact (name, phone , save=True):
    clean_name = name.strip() .lower()
    contacts[clean_name] = {"phone": phone}
    print (f"  Success: Contact '{name}' added/updated. ")
    if save:
        save_contacts()

def delete_contact(name):
    clean_name = name.strip().lower()
    if clean_name in contacts:
        del contacts [clean_name]
                                (parameter) name: Any
        save_contacts()
        print(f" Deleted: Contact '{name}' removed.")
    else:
        print(f" Error: Contact '{name}' not found.")

def search_contact(name):
    clean_name = name.strip().lower()
    if clean_name in contacts:
        phone = contacts[clean_name]["phone"]
        print(f"   Found: {name.title()} ->   {phone}")
    else:
        print(f"  Contact  '{name}'  not found.")

def show_all_contacts():
    print("\n---  ----- Contact List (CSV)  ---- ---")
    if not contacts:
        print("No contacts available.")
    else:
        for name in sorted(contacts.keys()):
```

```
def main():
            phone = input("Enter phone number: ")
            add_contact(name , phone)

        elif choice == '3':
            name = input("  Enter name to delete:   ")
            delete_contact(name)

        elif choice == '4':
            show_all_contacts()
        elif choice == '5':
            print("   Exiting.. 📬 📬. Data saved in 'contacts.csv'. Open it in Excel!=  ")
            break
        else:
            print("  Invalid choice. Please enter 1-5 ..... ")

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

3.Excel View: Shows the contacts.csv file opened in spreadsheet software.

TESTING APPROACH

The system was tested using **Black Box Testing** techniques:

1. **Positive Testing:** Entering valid names and numbers.
    - *Result:* Data saved successfully.
2. **Negative Testing:** Searching for a non-existent user.
    - *Result:* "Contact not found" error displayed gracefully.
3. **Boundary Testing:** Deleting the last remaining contact.
    - *Result:* System handles empty dictionary correctly without crashing.
4. **Persistence Testing:** Closing the program and reopening it.
    - *Result:* Previously added contacts were successfully loaded from the CSV.

CHALLENGES FACED

- **Case Sensitivity:** Initially, inputs like "Arindu" and "arindu" were treated as different contacts. This was resolved by converting all inputs to.lower() before storage in the dictionary.
- **File Not Found Error:** On the very first run, the program crashed because contacts.csv did not exist. This was fixed by adding an os.path.exists() check in the load_contacts function to initialize default data if the file is missing.

LEARNINGS & KEY TAKEAWAYS

- Learned how to manipulate CSV files using Python's csv library.
- Understood the importance of separating data storage (CSV) from runtime data structures (Dictionaries).
- Gained experience in designing a user-friendly CLI loop (while True).
- Learned how to handle basic file system errors and initialize default states.

FUTURE ENHANCEMENTS

1. **Input Validation:** Implement Regex to ensure phone numbers are exactly 10 digits.
2. **GUI Implementation:** Upgrade the CLI to a Graphical User Interface using Tkinter or PyQt.
3. **Multiple Fields:** Add support for email addresses and home addresses alongside phone numbers.

REFERENCES

1. RG Dromey Text Book
2. Python Essential