# PROJECT REPORT

## CONTACTS MANAGEMENT SYSTEM

**Name :** Naman Singh

**Roll Number:** 25BAI11153

**Course: CSE 1021 Introduction to Problem Solving and**

## Table of Contents

## 1. Introduction

The **Contacts Management System** is a simple Python program that runs in the command line (CLI). It helps users store and find contact details easily. Keeping track of names, phone numbers, and addresses is important for everyone, and this project provides a basic way to do that without needing a physical notebook.

I built this project to practice the core programming concepts I've learned, such as writing functions, using dictionaries, and handling user input. It uses a simple menu that lets you add, view, update, and delete contacts.

## 2. Problem Statement

Many people and even small organizations struggle to keep their contact lists organized. Using paper notebooks or sticky notes has several problems:

- **Getting Lost:** It's easy to lose a piece of paper.
- **Hard to Search:** You have to flip through pages to find a specific number.
- **Messy Updates:** Crossing out old numbers and writing new ones makes records look untidy.

There was a need for a simple, digital way to store this information. This project solves that problem by letting users manage contacts on their computer, ensuring the data is neat and easy to find.

## 3. Functional Requirements

The system allows the user to do the following:

1. **Add a Contact:**
    a. You can save a new contact by entering a First Name, Last Name, and Phone Number.
    b. The system checks if the phone number is valid (between 10 and 12 digits).
2. **View All Contacts:**
    a. It shows a list of every contact currently saved in the system.
3. **Search:**
    a. **By Name:** Enter a full name to find their phone number.
    b. **By Phone:** Enter a number to see who it belongs to.
4. **Update Details:**
    a. You can change a person's phone number if it changes.
    b. You can also fix a mistake in a person's name.
5. **Delete a Contact:**
    a. If you don't need a contact anymore, you can remove it permanently.
6. **Add More Info:**
    a. You can add extra details like Date of Birth, Address, Email, or a quick Note to any existing contact.

## 4. Non-functional Requirements

- **Easy to Use:** The menu is text-based and asks simple questions, so anyone can use it without confusion.
- **Fast:** Since the program stores data in the computer's memory (RAM), searching and adding contacts happens instantly.
- **Error Checking:** The program tries to prevent mistakes, like warning you if you try to add a contact that already exists or if you type an invalid phone number.
- **Portable:** The code is written in standard Python 3, so it works on any computer that has Python installed.

## 5. System Architecture

The program is structured in a simple, modular way. It revolves around a "Main Loop" that keeps the program running until the user chooses to exit.

- **User Interface:** This is the Command Line Interface (CLI) where the user types options (like '1', '2', etc.).
- **Logic: Use of** separate functions for each task, such as `add_contacts()` and `delete_contact()`.
- **Data Storage:** global variables (Dictionaries) to hold the data while the program is running:
  - `contact = {}`: Stores the Name and Phone Number.
  - `ad = {}`: Stores extra details like Address and Email.

## 6. Design Decisions & Rationale

- **Why Python?** I chose Python because the syntax is clean and easy to read. It allowed me to focus on the logic of the application rather than worrying about complex syntax.
- **Why Dictionaries?** I decided to use Python Dictionaries (`dict`) to store the data because they are very fast for looking things up. If I had used a list, searching for a name would take longer as the list grew. Dictionaries make finding a key (the name) instant.
- **Why CLI?** Building a Graphical User Interface (GUI) adds a lot of complexity. Since the goal was to master logic and data handling, a text-based menu was the best choice.

## 7. Implementation Details

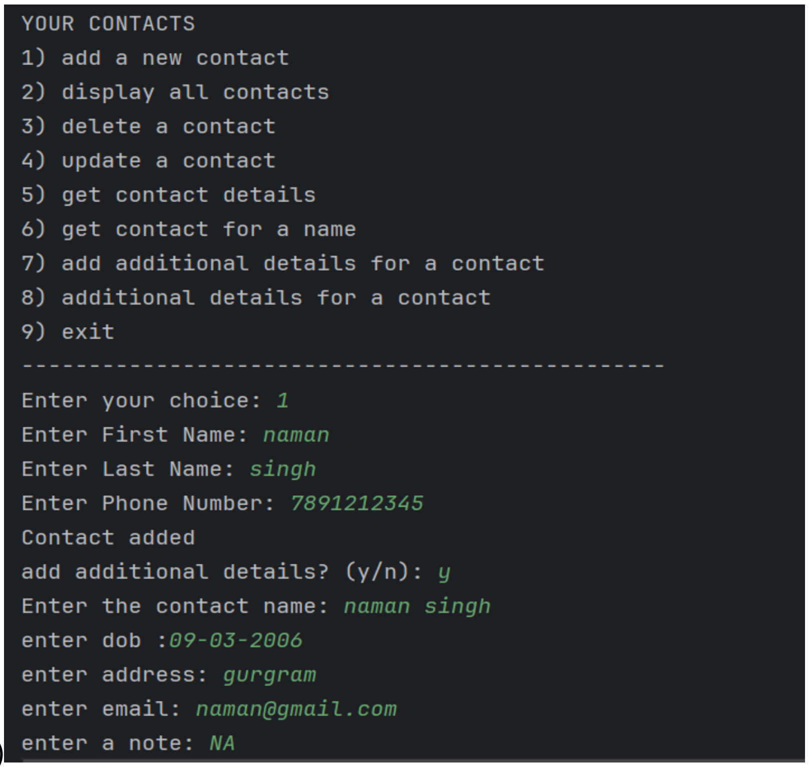
The entire project is written in one file named `contacts.py`.

- **Storage:**

```
contact = {} # Holds the main contact list
ad = {}      # Holds the additional details
```

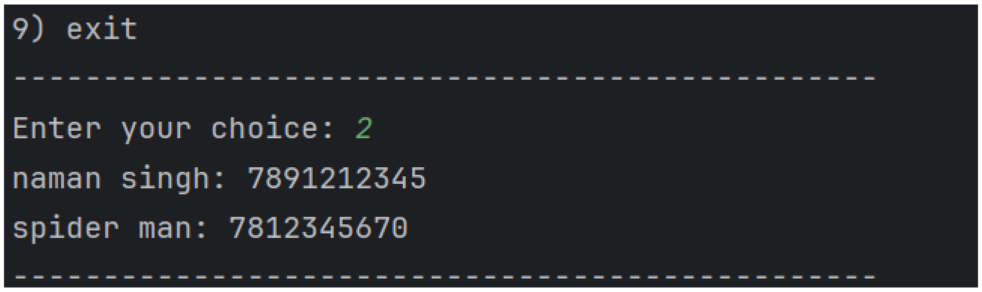
- **Key Functions:**
    - `add_contacts()`: Gets input from the user and validates the length of the phone number.
    - `get_contacts()`: Uses the name as a "key" to find the phone number in the dictionary.
    - `additional_details()`: Creates a list of details (DOB, Email, etc.) and saves it in the ad dictionary.
- **The Menu:** I used a `while` loop that prints the options 1-9. Inside the loop, `if-elif` statements check what number the user typed and call the correct function.

## 8. Screenshots / Results

1) 

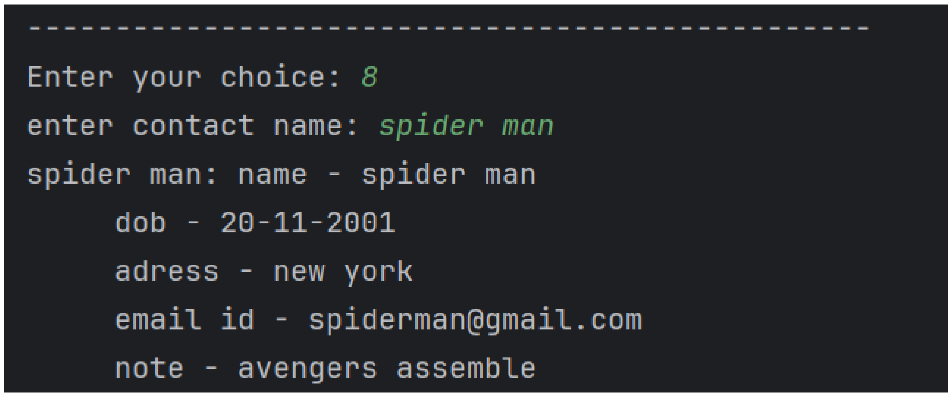
```
YOUR CONTACTS
1) add a new contact
2) display all contacts
3) delete a contact
4) update a contact
5) get contact details
6) get contact for a name
7) add additional details for a contact
8) additional details for a contact
9) exit
------------------------------------------------
Enter your choice: 1
Enter First Name: naman
Enter Last Name: singh
Enter Phone Number: 7891212345
Contact added
add additional details? (y/n): y
Enter the contact name: naman singh
enter dob :09-03-2006
enter address: gurgram
enter email: naman@gmail.com
enter a note: NA
```

2) Dispaly all contacts



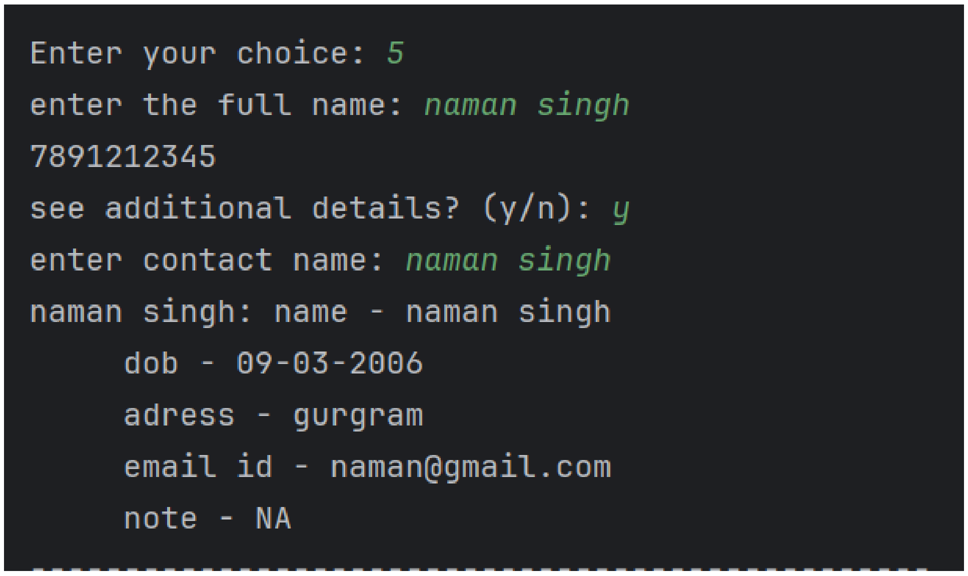
```
9) exit
-----------------------------------------------
Enter your choice: 2
naman singh: 7891212345
spider man: 7812345670
-----------------------------------------------
```

3) Check additional details for a saved contact



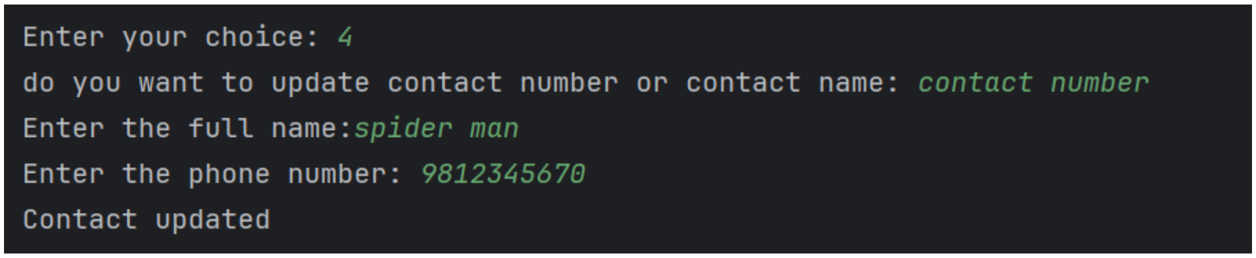
```
------------------------------------------------
Enter your choice: 8
enter contact name: spider man
spider man: name - spider man
     dob - 20-11-2001
     adress - new york
     email id - spiderman@gmail.com
     note - avengers assemble
```

4) check phone number for a contact name



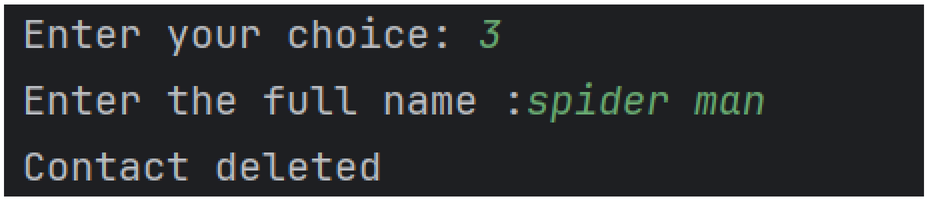
```
Enter your choice: 5
enter the full name: naman singh
7891212345
see additional details? (y/n): y
enter contact name: naman singh
naman singh: name - naman singh
     dob - 09-03-2006
     adress - gurgram
     email id - naman@gmail.com
     note - NA
------------------------------------------------
```

5) Update a contact number

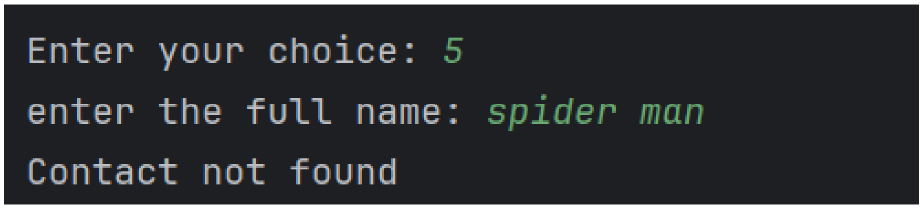


```
Enter your choice: 4
do you want to update contact number or contact name: contact number
Enter the full name:spider man
Enter the phone number: 9812345670
Contact updated
```

6) Deleting a contact

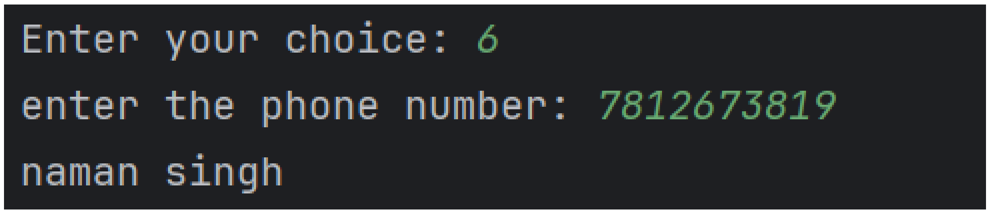


```
Enter your choice: 3
Enter the full name :spider man
Contact deleted
```



```
Enter your choice: 5
enter the full name: spider man
Contact not found
```

7) Find the name associated with a contact number



```
Enter your choice: 6
enter the phone number: 7812673819
naman singh
```

## 10. Challenges Faced

- **Handling Errors:** Python is a case sensitive language, and all the user's inputs given must be typed precisely. So care must be taken when searching for a specific contact to get the desired result
- **Syncing Data:** I have two separate dictionaries (one for numbers, one for details). I had to make sure that if I added details, the contact actually existed in the main list first.

## 11. Learnings & Key Takeaways

Working on this project helped me understand several things:

- **Logic Building:** I got much better at using `if-else` conditions and loops to control the flow of a program.
- **Data Structures:** I really understood how Dictionaries work and why Key-Value pairs are useful for linking data (like a Name to a Number).
- **Modular Code:** Breaking the code into small functions (like add vs `delete`) made it much easier to write and fix errors compared to writing one huge block of code.

## 12. Future Enhancements

If I were to work on this project further, I would add:

1. **File Storage:** I would use text files or a CSV file to save the contacts so they don't get deleted when the program stops.
2. **Better Search:** Right now, you have to type the exact full name. I would like to make it so you can just type "John" and find "John Doe".

## 14. References

1. Vityarthi course
2. Python libraries