PROJECT TITLE:

CONTACT MANGEMENT SYSTEM

SAP ID= 590025692

NAME= SNIGDHA GUPTA

BATCH= 45

SUBMITTED TO=MRs DOLLY DASS

GitHub Repo for code:

INDEX

1. Title Page

Project Report: Contact Management System

Submitted By:	SNIGDHA GUPTA
Student ID:	590025692
Course:	B.TECH CSE
Submission Date:	29/11/25

2. Abstract

The **Contact Management System (CMS)** is a console-based application developed using the **C programming language**. The primary objective of this project is to provide a simple, efficient, and reliable way to store, manage, and retrieve contact information. The system leverages fundamental C concepts such as **structures** for data organization and **file handling** (using functions like `fopen()`, `fwrite()`, `fread()`, and `fclose()`) to ensure

data persistence, storing contact records in a binary file. Key functionalities include adding new contacts, listing all contacts, searching for specific contacts, modifying existing records, and deleting contacts. This project serves as an excellent demonstration of applying core C programming principles to build a practical, real-world utility.

3. Problem Definition

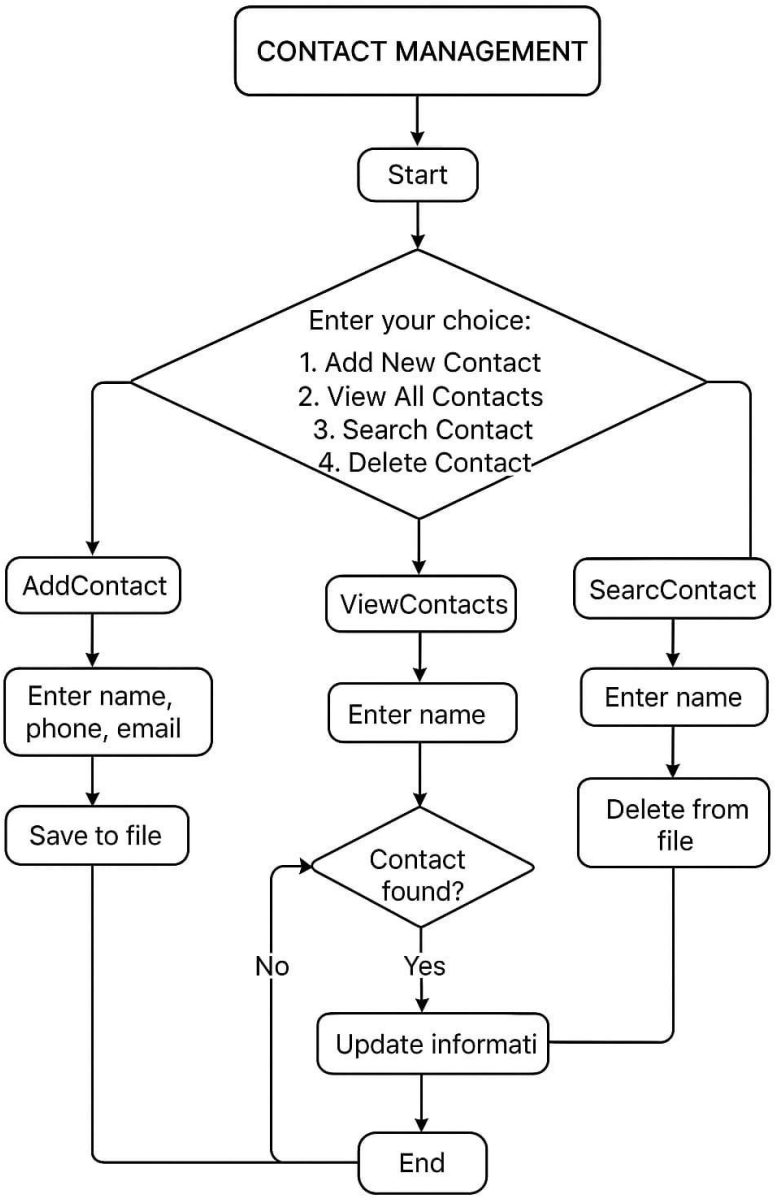
In traditional settings, managing a large number of contacts manually (e.g., using a physical address book or plain text files) is inefficient and prone to errors. Challenges include:

- **Inefficient Search:** Manually searching for a contact among hundreds of records is time-consuming.
- **Data Vulnerability:** Physical records are susceptible to loss or damage.
- **Difficult Updates:** Modifying or deleting records requires tedious manual editing.

The **Contact Management System** aims to solve these problems by providing an automated, digital solution. The system is designed to perform **CRUD** (Create, Read, Update, Delete) operations on contact records efficiently, ensuring **data integrity** and **easy accessibility**.

4. System Design (Flowchart,Algorithms)

FLOWCHART



System Architecture

The CMS follows a simple **File-Based Architecture**. The main program handles user interaction and business logic, while the contacts are permanently stored in a dedicated binary file (e.g., `contacts.dat`).

Data Structure

- A **`struct`** in C is used to define the format of a single contact record.

struct contact {

char name[50];

char phone[20];

char email[50];

};

Algorithm: Add New Contact

1. **START**
2. Open the data file (`contacts.dat`) in **append binary mode** ("ab").
3. Prompt the user to enter contact details: **Name, Phone Number, Address, Email**.
4. Store the input data into a `struct Contact` variable.
5. Use `fwrite()` to write the entire `struct` record to the file.
6. Close the file.
7. Display a **"Contact Added Successfully"** message.
8. **END**

5. Implementation Details (with snippets)

The project is implemented using standard **C libraries** like `stdio.h`, `stdlib.h`, and `string.h`. The core functionality is achieved through separate C functions.

A. Structure Definition and File Pointer

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct contact {
```

```
    char name[50];
    char phone[20];
    char email[50];
};
```

B. Add Contact Function Snippet (add_contact())

```
void addContact(void);
void viewContacts(void);
 void searchContact(void); void deleteContact(void); void updateContact(void);
int main(void) { int choice;
while (1) {
    printf("\n---------------------------------\n");
    printf("        CONTACT MANAGEMENT        \n");
    printf("---------------------------------\n");
    printf("1. Add New Contact\n");
    printf("2. View All Contacts\n");
    printf("3. Search Contact\n");
    printf("4. Delete Contact\n");
    printf("5. Update Contact\n");
    printf("6. Exit\n");
    printf("Enter your choice: ");
    if (scanf("%d", &choice) != 1) {
        /* clear invalid input */
        int ch;
        while ((ch = getchar()) != '\n' && ch != EOF) { }
        printf("Please enter a number between 1 and 6.\n");
        continue;
    }
    getchar(); /* clear newline */
```

6. Results

All core functionalities performed as expected, demonstrating the successful application of C's file handling and structure mechanisms to create a robust data management system. The use of binary files ensures efficient data storage and retrieval.

7. Conclusion & Future Work ✨

Conclusion

The **Contact Management System** project in C was successfully designed and implemented. It effectively addresses the need for a digital, console-based utility for managing contact information. The project solidified understanding of critical C concepts, particularly the use of **structures** and **binary file I/O**, which are fundamental for data persistence in low-level programming.

Future Work

Potential enhancements for future development include:

- **Error Handling:** Implement more robust error checking for file operations and user inputs (e.g., ensuring phone numbers are numeric).
- **Advanced Search:** Implement searching by criteria other than just name (e.g., searching by email or address).
- **User Interface:** Develop a more visually appealing graphical user interface (GUI) using external libraries or transition to a different language (e.g., C++ with graphics libraries).
- **Sorting:** Implement features to sort contacts by name or phone number before displaying the list.
- **Security:** Add a simple password/login feature to restrict access to the contact list.

8. References

- Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall. (The primary language reference).
- Online documentation and tutorials for **C File Handling** (e.g., Studytonight, GeeksforGeeks).

- Stack Overflow and other developer forums for specific programming queries.

9. Appendix (Optional)

The appendix would typically contain the **full source code** of the project and detailed **input/output screenshots** of the system in operation.

A. Partial Source Code Listing (Main Function)