S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

<u>Project Report</u>

On

“ ***<u>Contact Management System in ‘C' Language</u>*** ”

Submitted By

Shivani Dhakate

(CS23109)

Guided By:-

Prof.Roshni Talmale

S. B. JAIN INSTITUTE OF TECHNOLOGY,
MANAGEMENT AND RESEARCH, NAGPUR
(An Autonomous Institute, Affiliated to RTMNU, Nagpur)
FIRST YEAR ENGINEERING DEPARTMENT
"Emerge as a leading Institute for developing competent and creative Professionals"

Index

Aim:

Create a simple program in C for managing a list of contacts, using concepts like structs, arrays, and file I/O, to practice and demonstrate C programming skills.

Objectives:

The objectives of implementing contact management in C could include:

1. **Data Structure Practice**: Use structs and arrays to organize and store contact information efficiently.

2. **File Handling**: Learn how to read from and write to files to store contacts persistently.

3**. User Interaction**: Implement a simple user interface for adding contacts and displaying the contact list.

4. **Error Handling**: Manage potential errors, such as a full contact list or incorrect user input.

5. **Functionality**: Provide basic operations like adding contacts, displaying contacts, and exiting the program.

6**. Code Organization**: Practice writing well-structured, readable code with clear separation of concerns.

7. **Testing and Debugging**: Test the program with various scenarios and debug any issues that arise.

8. **Learning Outcome:** Gain a better understanding of C programming concepts and improve programming skills.

Major C aspects used in the project:

In the contact management project in C, several key aspects of the language are used:

1. Structures: The `struct` keyword is used to define a structure to hold contact information (name, phone number, email).

2. Arrays: An array of structures is used to store multiple contacts.

3. File I/O: File I/O functions (`fopen`, `fclose`, `fwrite`, `fread`) are used to save contacts to a file and load contacts from a file.

4. User Input/Output: The `printf` and `scanf` functions are used for user input and output, providing an interactive interface for adding and displaying contacts.

5. Loops: A `do-while` loop is used to continuously display a menu and process user input until the user chooses to exit.

6. Conditional Statements: A `switch` statement is used to perform different actions based on the user's choice in the menu.

7. Functions: Functions are used to modularize the code, with separate functions for adding contacts, displaying contacts, and the main menu loop.

8. Header Files: Standard C header files (`stdio.h`, `string.h`) are included for standard I/O and string manipulation functions.

9. Error Handling: Basic error handling is implemented to check if the contact list is full before adding a new contact.

10. Comments: Comments are used to document the code and explain the purpose of each function and section of code.

Overall, the project demonstrates how to use these fundamental aspects of C to create a simple contact management system.

Introduction:

In today's interconnected world, efficient management of contacts is crucial for individuals and organizations alike. Whether it's maintaining a directory of business contacts, managing personal connections, or organizing customer information, a reliable Contact Management System (CMS) streamlines communication and enhances productivity. In this project, we embark on the development of a Contact Management System using the C programming language.

The Contact Management System will serve as a robust platform for storing, retrieving, updating, and deleting contact information. It will offer functionalities such as adding new contacts, searching for specific contacts based on various criteria, and modifying existing contact details. Additionally, the system will prioritize data security and integrity, ensuring that sensitive information remains protected.

Key Objectives:

1. **Efficient Data Organization:** The CMS will employ data structures and algorithms to efficiently organize contact information, enabling quick retrieval and manipulation of data.

2. **User-Friendly Interface**: A user-friendly interface will be developed to facilitate easy interaction with the system. Users will be able to perform operations intuitively, enhancing usability.

3. **Search and Filter Functionality:** The system will incorporate robust search and filter functionalities, allowing users to locate specific contacts based on criteria such as name, organization, or contact details.

4. **Data Security:** Implementation of secure data handling mechanisms will be a priority to safeguard sensitive contact information against unauthorized access or manipulation.

5. **Scalability:** The CMS will be designed with scalability in mind, allowing it to accommodate a growing database of contacts without compromising performance.

6. **Error Handling:** Comprehensive error handling mechanisms will be implemented to detect and handle errors effectively, ensuring the stability and reliability of the system.

7.**Documentation and Testing:** Thorough documentation will accompany the system, providing insights into its architecture, functionalities, and usage. Rigorous testing procedures will be employed to identify and rectify any bugs or inconsistencies

Major Library and Functions:

In C, there isn't a built-in library specifically for contact management like in some higher-level languages. However, you can use standard libraries and functions to manage contacts. Here are some major libraries and functions commonly used for this purpose:

1. stdio.h: This library provides functions for standard input and output, which are essential for interacting with the user. Functions like printf and scanf are used for this purpose.
2. string.h: This library provides functions for string manipulation, which can be useful for managing contact information. Functions like strcpy, strcat, and strcmp are commonly used.
3. stdlib.h: This library provides functions for memory allocation and other general utilities. Functions like malloc and free can be used for dynamic memory management if needed.
4. File I/O functions: To save contacts to a file or load contacts from a file, you can use functions like fopen, fclose, fwrite, and fread from stdio.h.
5. Data structures: While not a library per se, C programmers often use data structures like arrays, structs, and linked lists to organize and manage contact information efficiently.

Algorithm:

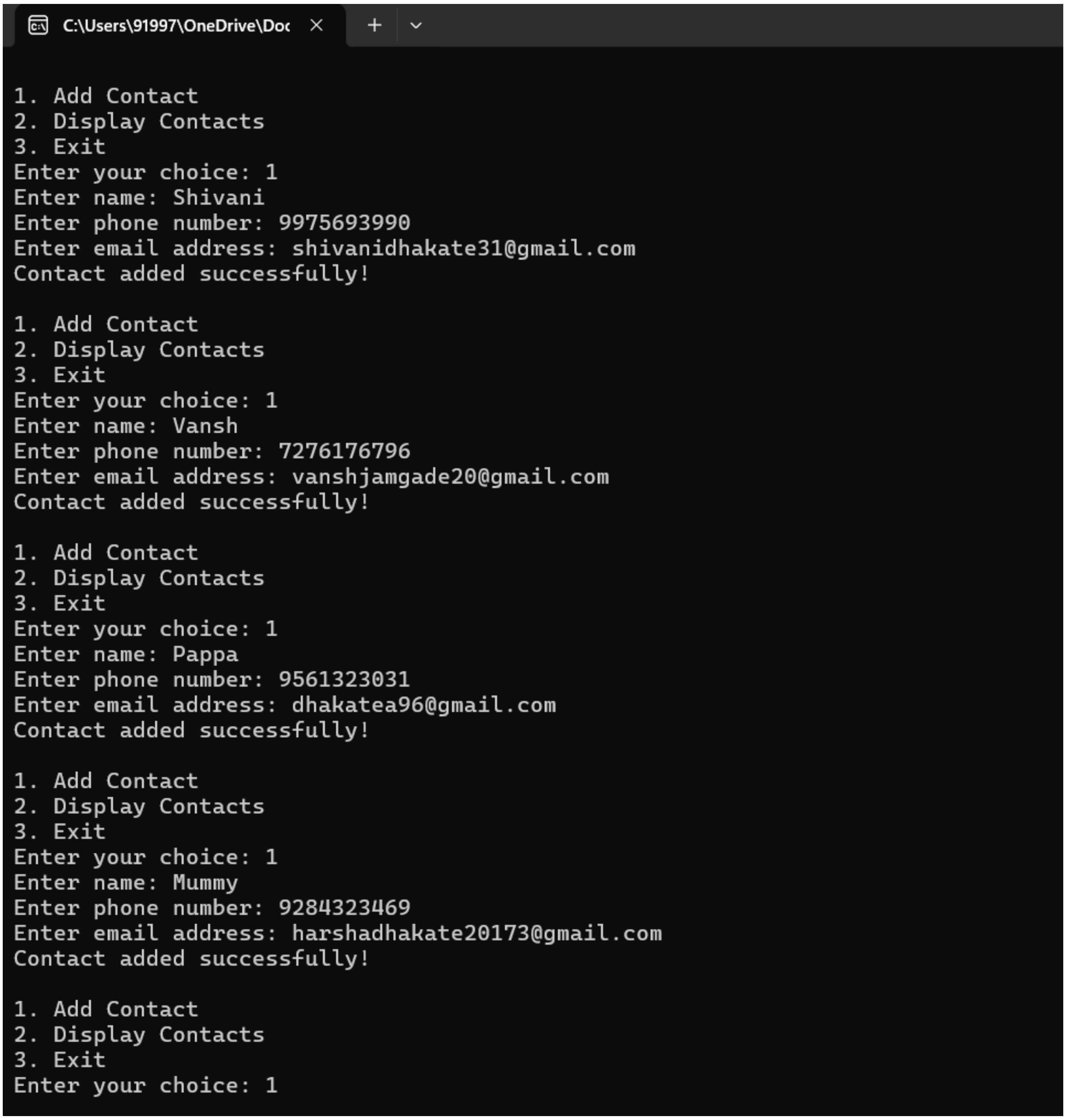
Here is a basic algorithm for the contact management program in C:

1. Include the necessary header files (`stdio.h` and `string.h`).
2. Define a constant `MAX_CONTACTS` to limit the number of contacts.
3. Define a structure `Contact` to hold contact information (name, phone, email).
4. Define an array `contacts` of type `Contact` to store multiple contacts.
5. Define a variable `numContacts` to keep track of the number of contacts added.
6. Implement the `addContact` function:
   a. Check if the number of contacts is less than `MAX_CONTACTS`.
   b. If yes, prompt the user to enter name, phone number, and email for the new contact.
   c. Store the new contact in the `contacts` array at the index `numContacts`.
   d. Increment `numContacts` and display a success message.
   e. If the contact list is full, display a message indicating that the contact list is full.
7. Implement the `displayContacts` function:
   a. Print a header indicating that the contacts are being displayed.
   b. Iterate through the `contacts` array and print each contact's details (name, phone, email) along with its index.
8. Implement the `main` function:
   a. Declare variables `choice` to store user input for menu choice.
   b. Use a `do-while` loop to display a menu and process user input until the user chooses to exit.
   c. Inside the loop, use a `switch` statement to perform different actions based on the user's choice:
      - If the user chooses to add a contact, call the `addContact` function.
      - If the user chooses to display contacts, call the `displayContacts` function.
      - If the user chooses to exit, display a message and exit the loop.
      - If the user enters an invalid choice, display a message indicating that the choice is invalid.
9. Compile and run the program.

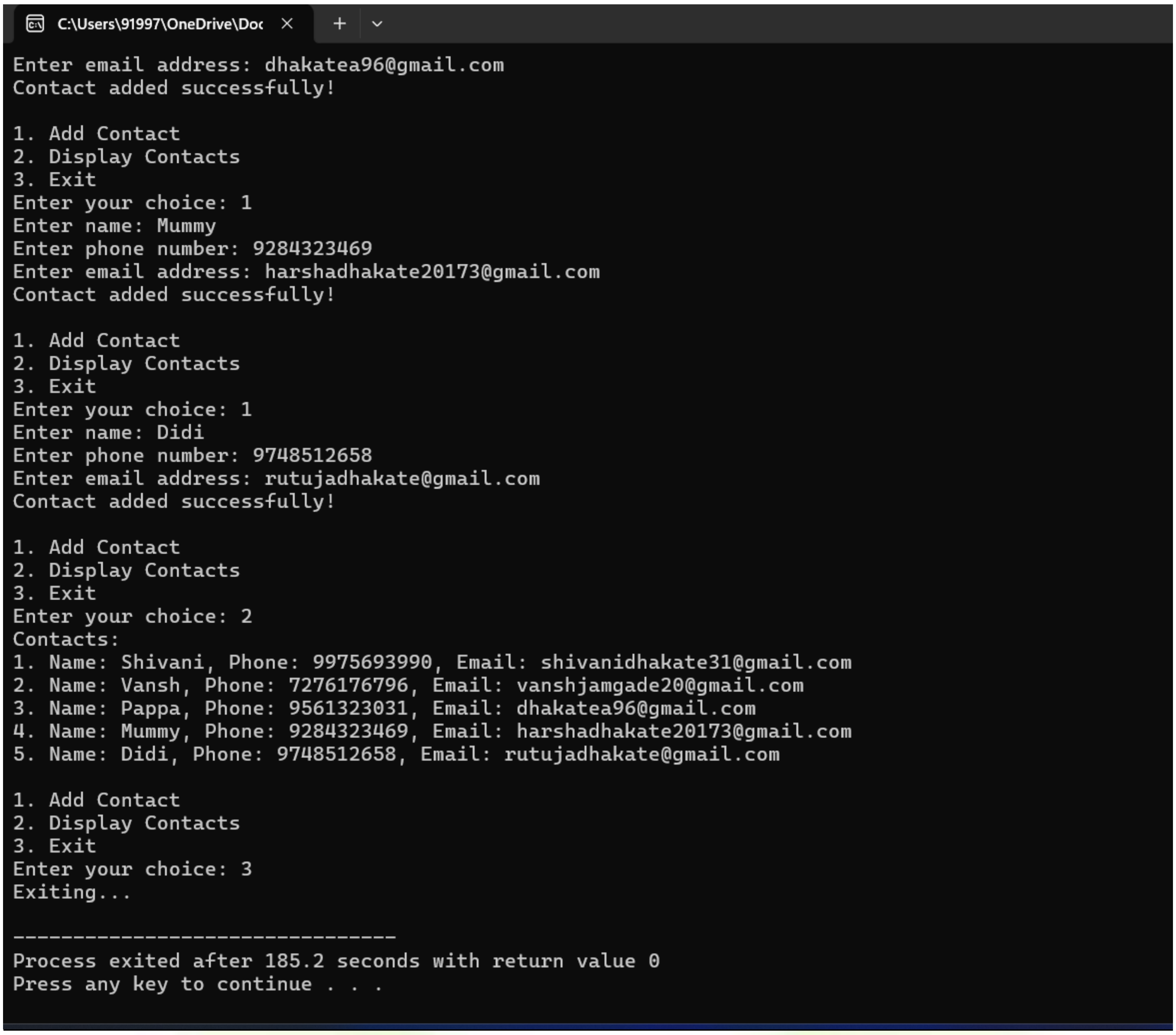
CODE:

```
#include <stdio.h>
#include <string.h>

#define MAX_CONTACTS 100

struct Contact {
    char name[50];
    char phone[20];
    char email[50];
};

struct Contact contacts[MAX_CONTACTS];
int numContacts = 0;

void addContact() {
    if (numContacts < MAX_CONTACTS) {
        struct Contact newContact;
        printf("Enter name: ");
        scanf("%s", newContact.name);
        printf("Enter phone number: ");
        scanf("%s", newContact.phone);
        printf("Enter email address: ");
        scanf("%s", newContact.email);

        contacts[numContacts] = newContact;
        numContacts++;
        printf("Contact added successfully!\n");
    } else {
        printf("Contact list is full!\n");
    }
}

void displayContacts() {
    printf("Contacts:\n");
    for (int i = 0; i < numContacts; i++) {
        printf("%d. Name: %s, Phone: %s, Email: %s\n", i+1, contacts[i].name,
contacts[i].phone, contacts[i].email);
    }
}

int main() {
    int choice;
    do {
        printf("\n1. Add Contact\n2. Display Contacts\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addContact();
                break;
            case 2:
                displayContacts();
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    } while (choice != 3);

    return 0;
}
```

Result & Discussion (Output):

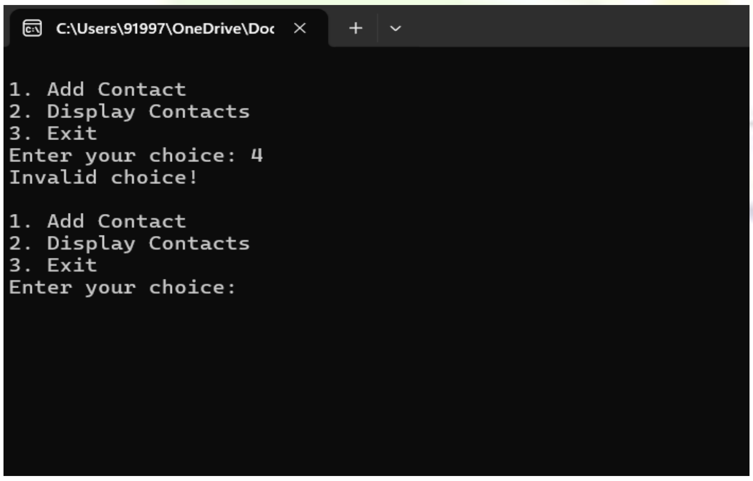


```
1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
Enter name: Shivani
Enter phone number: 9975693990
Enter email address: shivanidhakate31@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
Enter name: Vansh
Enter phone number: 7276176796
Enter email address: vanshjamgade20@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
Enter name: Pappa
Enter phone number: 9561323031
Enter email address: dhakatea96@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
Enter name: Mummy
Enter phone number: 9284323469
Enter email address: harshadhakate20173@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
```



```
Enter email address: dhakatea96@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
Enter name: Mummy
Enter phone number: 9284323469
Enter email address: harshadhakate20173@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 1
Enter name: Didi
Enter phone number: 9748512658
Enter email address: rutujadhakate@gmail.com
Contact added successfully!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 2
Contacts:
1. Name: Shivani, Phone: 9975693990, Email: shivanidhakate31@gmail.com
2. Name: Vansh, Phone: 7276176796, Email: vanshjamgade20@gmail.com
3. Name: Pappa, Phone: 9561323031, Email: dhakatea96@gmail.com
4. Name: Mummy, Phone: 9284323469, Email: harshadhakate20173@gmail.com
5. Name: Didi, Phone: 9748512658, Email: rutujadhakate@gmail.com

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 3
Exiting...

--------------------------------
Process exited after 185.2 seconds with return value 0
Press any key to continue . . .
```

Selecting wrong option number



```
1. Add Contact
2. Display Contacts
3. Exit
Enter your choice: 4
Invalid choice!

1. Add Contact
2. Display Contacts
3. Exit
Enter your choice:
```

Explanation of Functions:

Functions used in the contact management program:

1. `void addContact()`:

- This function is used to add a new contact to the `contacts` array.

- It first checks if the `numContacts` is less than `MAX_CONTACTS` to ensure there is space for a new contact.

- If there is space, it prompts the user to enter the name, phone number, and email address for the new contact using `scanf`.

- It then creates a new `Contact` struct with the entered information and adds it to the `contacts` array at index `numContacts`.

- Finally, it increments `numContacts` to indicate that a new contact has been added.

2. `void displayContacts()`:

- This function is used to display all the contacts stored in the `contacts` array.

- It prints a header "Contacts:" to indicate that the list of contacts is being displayed.

- It then iterates through the `contacts` array using a `for` loop and prints the details (name, phone number, email) of each contact using `printf`.

3. `int main()`:

- The `main` function is the entry point of the program.

- It contains a `do-while` loop that displays a menu of options (Add Contact, Display Contacts, Exit) and processes the user's choice.

- Inside the loop, it uses a `switch` statement to call the appropriate function based on the user's choice:

- If the user chooses to add a contact (choice 1), it calls the `addContact` function.

- If the user chooses to display contacts (choice 2), it calls the `displayContacts` function.

- If the user chooses to exit (choice 3), it prints a message and exits the loop.

- If the user enters an invalid choice, it prints a message indicating that the choice is invalid.

- The loop continues until the user chooses to exit by entering 3.

These functions work together to create a simple contact management system that allows users to add contacts, display the list of contacts, and exit the program.

Conclusion:

In conclusion, contact management in C can be implemented using standard libraries and functions such as `stdio.h` for input/output, `string.h` for string manipulation, and data structures like arrays or structs to store contact information. While C does not have built-in libraries specifically for contact management, you can create a simple contact management system using these standard libraries and functions.

References:

1. www.wikipedia.org/
2. www.Google.com
3. www.beginnersbook.com
4. https://youtube.com

Signature
Prof. Roshni Talmale
Project Guide & Course Coordinator
B.Tech CSE
Sem: II (2023-2024)