PROJECT REPORT

Course Code: CSEG1032

Course Title: Programming in C

Project Title: Contact Management System

Student Name: Prince Raj Rawat

SAP ID: 590022507

Semester: 1st

---

1. ABSTRACT

This project implements a menu-driven Contact Management System using the C programming language. The program allows the user to add, view, search, and delete contact records stored persistently in a file. It demonstrates the use of structures, file handling, and modular programming techniques to build a scalable and maintainable software system.

2. OBJECTIVE

The objective of this project is to apply fundamental and intermediate concepts of the C language to design a working application. It aims to help store contact information efficiently.

• Use structures to represent complex data.
• Employ functions and modular design for code organization.
• Perform file operations for persistent data storage.

3. PROBLEM DEFINITION

Almost every people require simple tools to maintain contact . Manually handling these records is error-prone and inefficient. This project provides a basic command-line system for managing contact information such as Name, Phone Number, and Email Address in a text file.

4. SYSTEM DESIGN AND ALGORITHM

The system is modular, consisting of multiple source files:

• main.c controls program flow and user interaction.

• call.c handles CRUD (Create, Read, Update/Delete) operations.

5. IMPLEMENTATION DETAILS

Key Language Features Used:

• Structures to represent contact info.

• File Handling functions (fopen, fwrite, fread, fclose) to store and retrieve data.

• Functions for modularity and clarity.

Example Code Snippet

6. OUTPUT (Sample)

Menu:

```
1.1.Add Contact
  2. View Contacts
  3. Search Contact
  4. Delete Contact
  5. Exit
2.INPUT : 1
```

Sample Output

```
Enter Name: John Doe
Enter Phone: 9876543210
Enter Email: john@example.com
Contact added successfully!
```

7. CONCLUSION

The Contact Management System demonstrates the practical application of C programming concepts such as structures, file handling, and modular design. It provides a simple yet effective solution for managing contact records in small organizations.

8. FUTURE ENHANCEMENTS

• Add Update Contact feature.

• add grouping in contact like family, friends , classmate etc.

• Create a GUI-based interface for better usability.

9. APPENDIX:

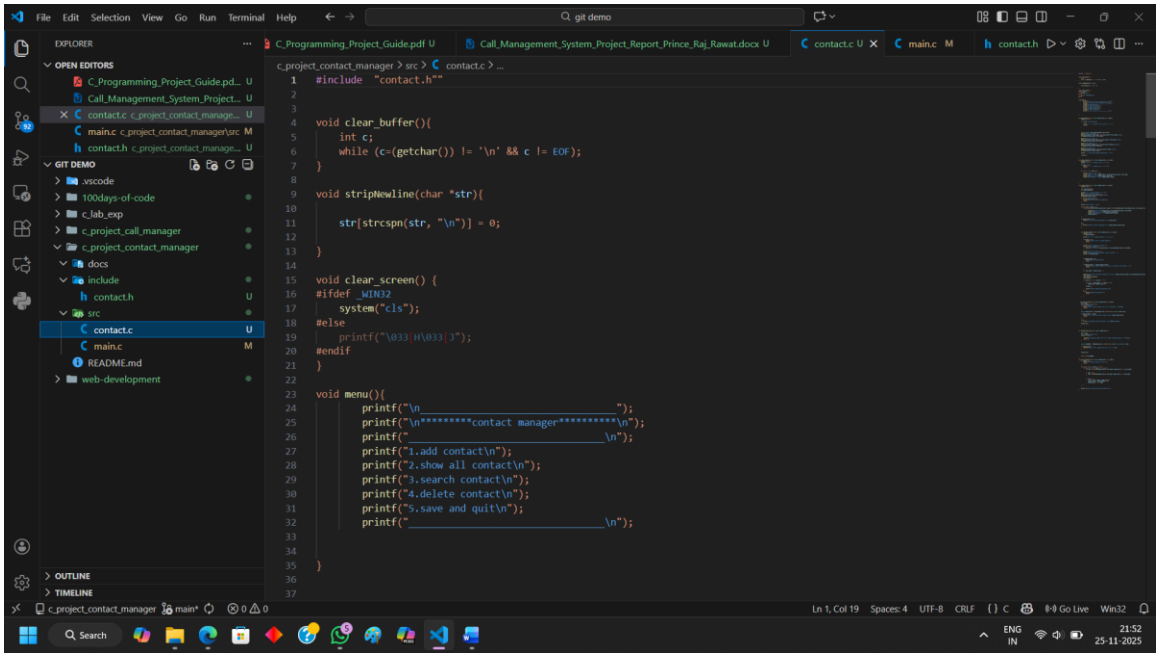
main.c (Provided)

c_project_contact_manager > src > main.c > ...

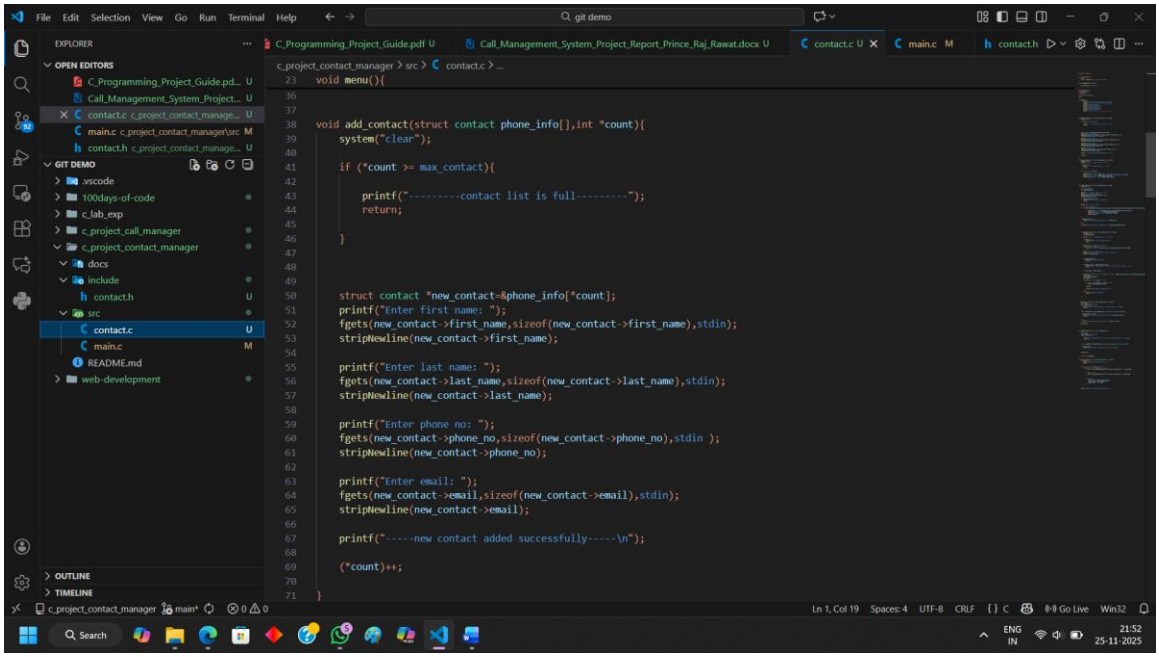
```c
#include "contact.h"


int main(){

    struct contact phone_info[max_contact];
    int count=0;
    int opt=0;

    printf("starting contact manager..\n");

    count = upload_from_file(phone_info);
    printf("Loaded %d contacts from file.\n", count);



    while(opt != 5)
    {
        menu();

        printf("\nenter :");
        scanf("%d",&opt);
        clear_buffer();

        switch(opt){

            case 1 :
                {clear_screen();
                add_contact(phone_info, &count);
                sort_contacts(phone_info, count);
                break;}
            case 2 :
                {clear_screen();
                show_contacts(phone_info, count);
                break;}
             case 3 :
                {clear_screen();
```

```c
int main(){
    while(opt != 5)
        switch(opt){
            case 1 :
                break;}
            case 2 :
                {clear_screen();
                show_contacts(phone_info, count);
                break;}
             case 3 :
                {clear_screen();
                find_contact(phone_info, count);
                break;}
            case 4 :
                {clear_screen();
                delete_contact(phone_info, &count);
                break;}
            case 5 :
                {save_in_file(phone_info, count);
                break;}
            default:
                {printf("invalid choice please choose no. from 1-5.\n");}
        }


    }



    return 0;

}
```

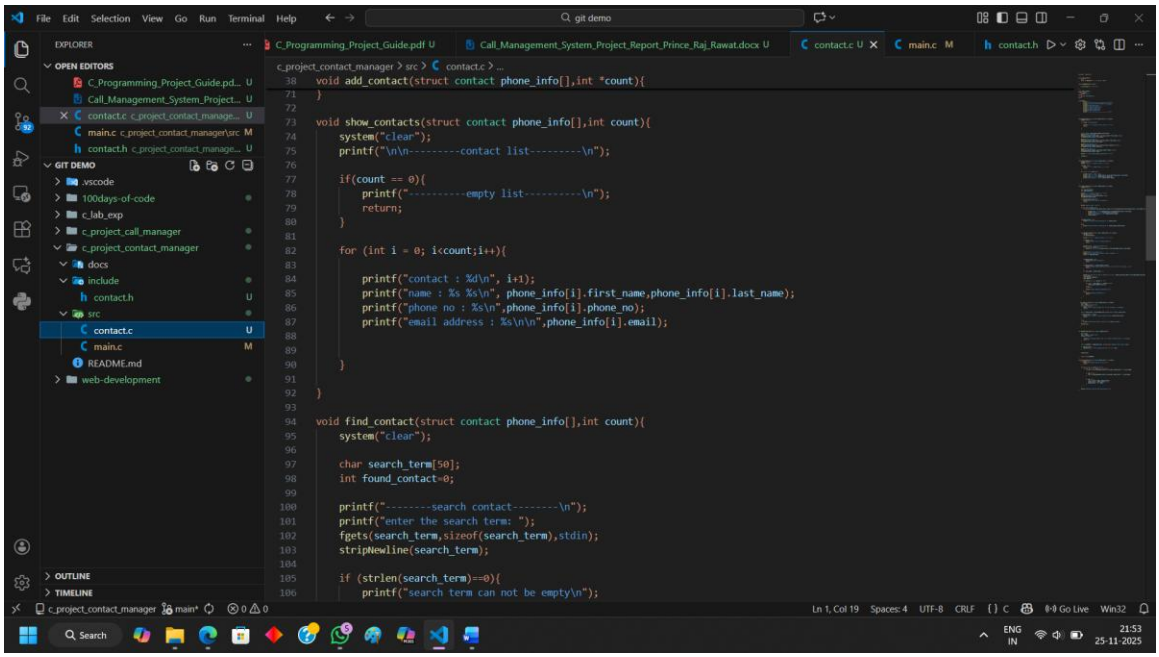
contact.c



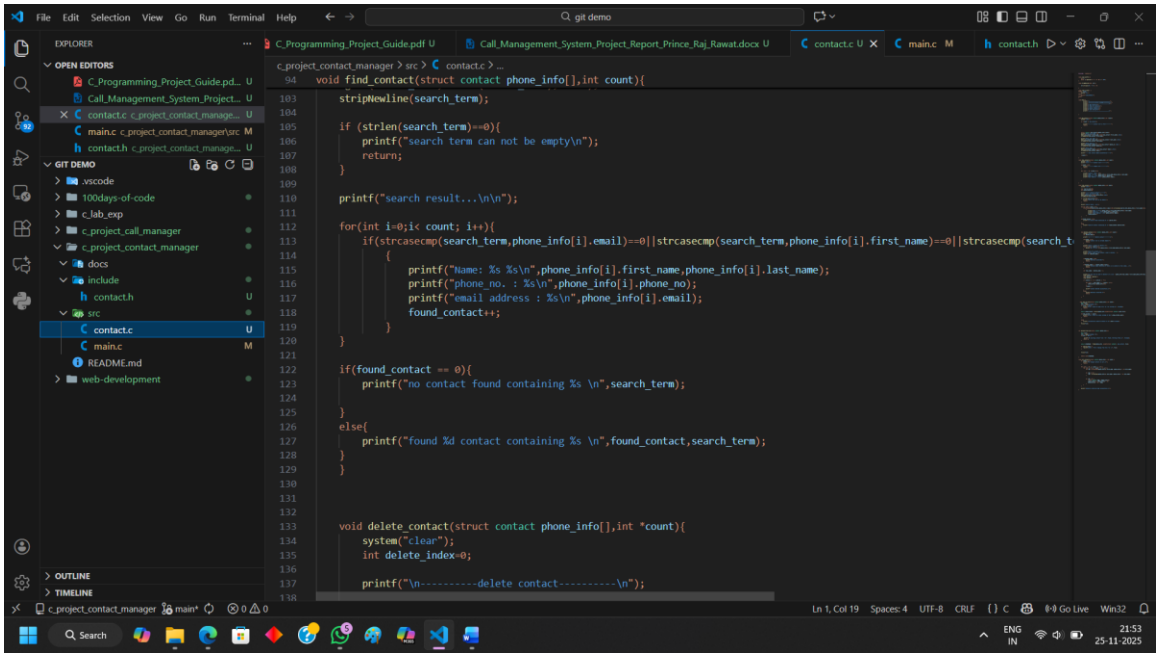
```c
#include  "contact.h""


void clear_buffer(){
    int c;
    while (c=(getchar()) != '\n' && c != EOF);
}

void stripNewline(char *str){

    str[strcspn(str, "\n")] = 0;

}

void clear_screen() {
#ifdef _WIN32
    system("cls");
#else
    printf("\033[H\033[J");
#endif
}

void menu(){
        printf("\n_____________________________");
        printf("\n*********contact manager**********\n");
        printf("_____________________________\n");
        printf("1.add contact\n");
        printf("2.show all contact\n");
        printf("3.search contact\n");
        printf("4.delete contact\n");
        printf("5.save and quit\n");
        printf("_____________________________\n");


}
```



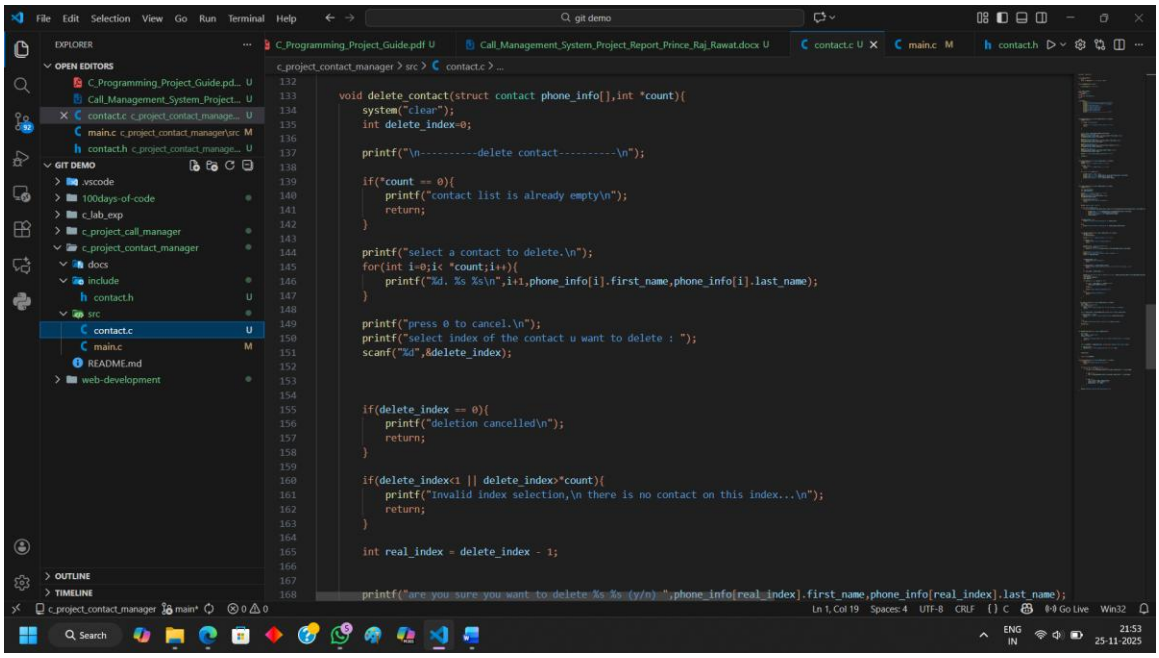
```c
void menu(){


void add_contact(struct contact phone_info[],int *count){
    system("clear");

    if (*count >= max_contact){

        printf("---------contact list is full---------");
        return;

    }



    struct contact *new_contact=&phone_info[*count];
    printf("Enter first name: ");
    fgets(new_contact->first_name,sizeof(new_contact->first_name),stdin);
    stripNewline(new_contact->first_name);

    printf("Enter last name: ");
    fgets(new_contact->last_name,sizeof(new_contact->last_name),stdin);
    stripNewline(new_contact->last_name);

    printf("Enter phone no: ");
    fgets(new_contact->phone_no,sizeof(new_contact->phone_no),stdin );
    stripNewline(new_contact->phone_no);

    printf("Enter email: ");
    fgets(new_contact->email,sizeof(new_contact->email),stdin);
    stripNewline(new_contact->email);

    printf("-----new contact added successfully-----\n");

    (*count)++;

}
```



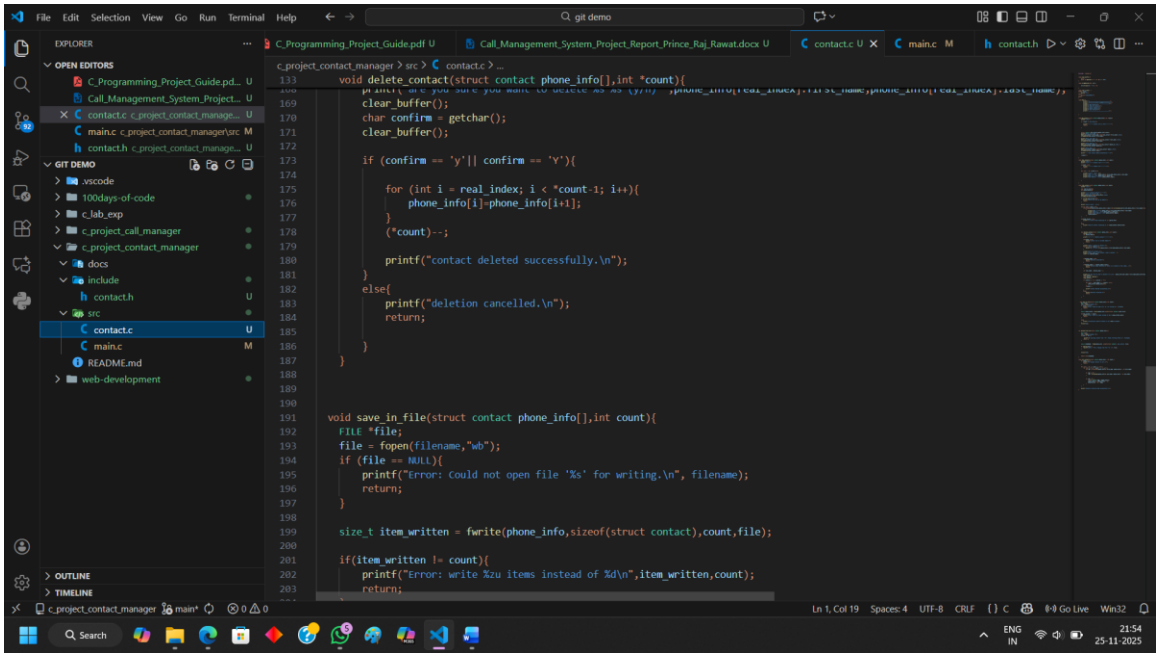
```
void add_contact(struct contact phone_info[],int *count){
}

void show_contacts(struct contact phone_info[],int count){
    system("clear");
    printf("\n\n---------contact list---------\n");

    if(count == 0){
        printf("----------empty list----------\n");
        return;
    }

    for (int i = 0; i<count;i++){

        printf("contact : %d\n", i+1);
        printf("name : %s %s\n", phone_info[i].first_name,phone_info[i].last_name);
        printf("phone no : %s\n",phone_info[i].phone_no);
        printf("email address : %s\n\n",phone_info[i].email);


    }
}

void find_contact(struct contact phone_info[],int count){
    system("clear");

    char search_term[50];
    int found_contact=0;

    printf("--------search contact--------\n");
    printf("enter the search term: ");
    fgets(search_term,sizeof(search_term),stdin);
    stripNewline(search_term);

    if (strlen(search_term)==0){
        printf("search term can not be empty\n");
```



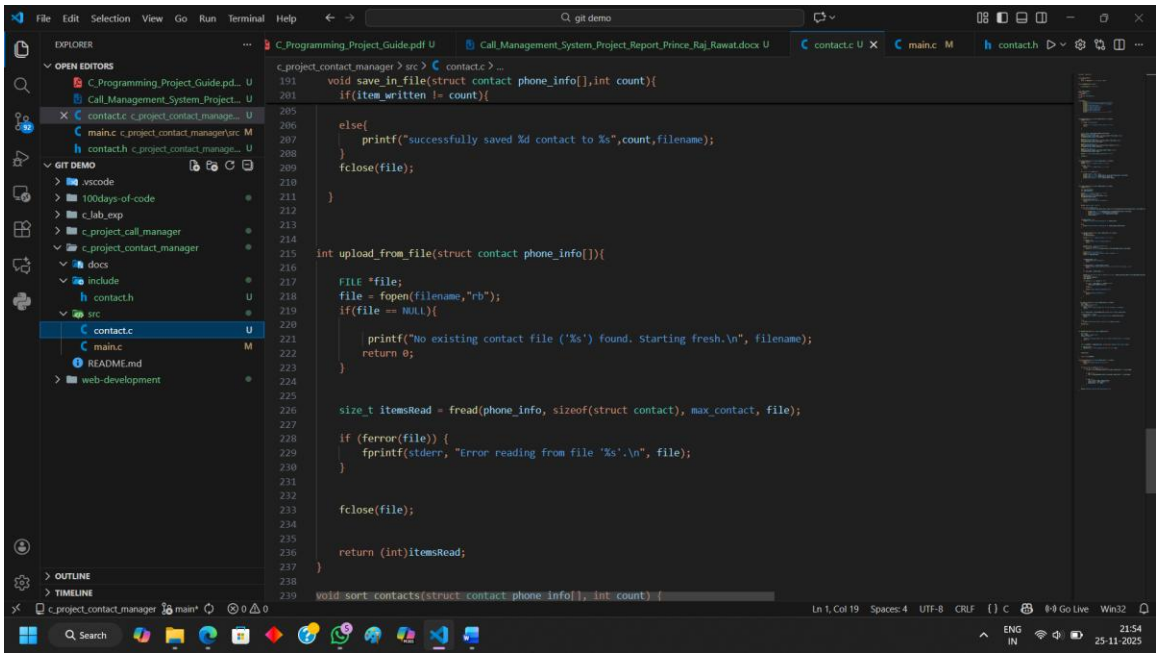
```
void find_contact(struct contact phone_info[],int count){
    stripNewline(search_term);

    if (strlen(search_term)==0){
        printf("search term can not be empty\n");
        return;
    }

    printf("search result...\n\n");

    for(int i=0;i< count; i++){
        if(strcasecmp(search_term,phone_info[i].email)==0||strcasecmp(search_term,phone_info[i].first_name)==0||strcasecmp(search_t
            {
                printf("Name: %s %s\n",phone_info[i].first_name,phone_info[i].last_name);
                printf("phone_no. : %s\n",phone_info[i].phone_no);
                printf("email address : %s\n",phone_info[i].email);
                found_contact++;
            }
    }

    if(found_contact == 0){
        printf("no contact found containing %s \n",search_term);

    }
    else{
        printf("found %d contact containing %s \n",found_contact,search_term);
    }
    }



    void delete_contact(struct contact phone_info[],int *count){
        system("clear");
        int delete_index=0;

        printf("\n----------delete contact----------\n");
```



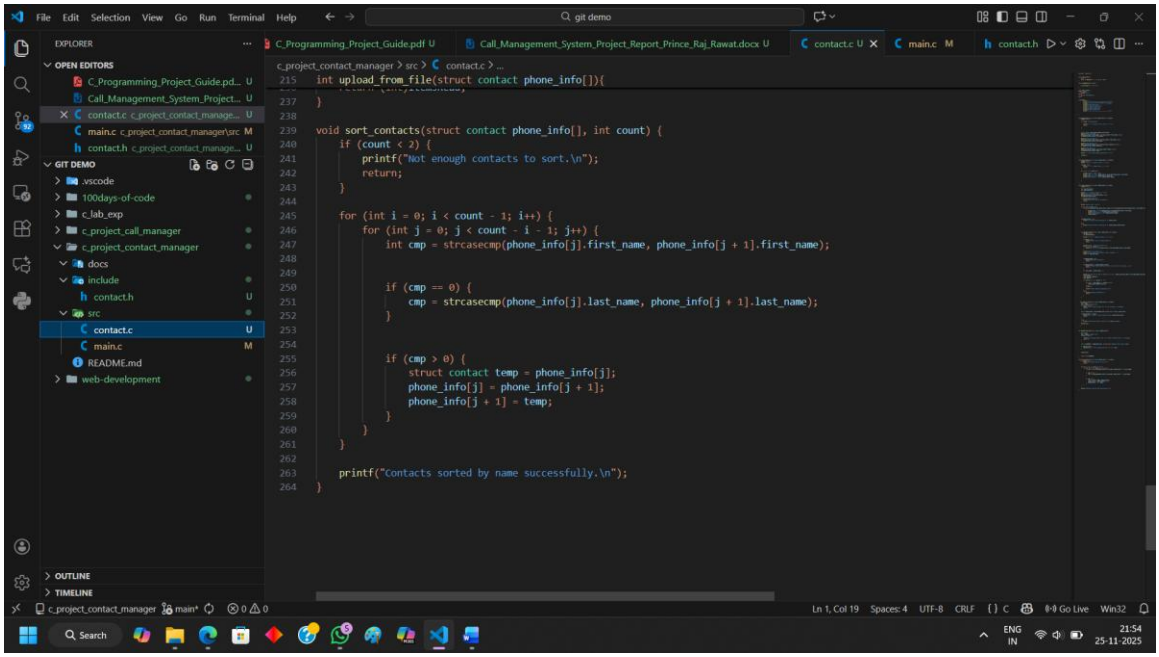
```c
void delete_contact(struct contact phone_info[],int *count){
    system("clear");
    int delete_index=0;

    printf("\n---------delete contact---------\n");

    if(*count == 0){
        printf("contact list is already empty\n");
        return;
    }

    printf("select a contact to delete.\n");
    for(int i=0;i< *count;i++){
        printf("%d. %s %s\n",i+1,phone_info[i].first_name,phone_info[i].last_name);
    }

    printf("press 0 to cancel.\n");
    printf("select index of the contact u want to delete : ");
    scanf("%d",&delete_index);



    if(delete_index == 0){
        printf("deletion cancelled\n");
        return;
    }

    if(delete_index<1 || delete_index>*count){
        printf("invalid index selection,\n there is no contact on this index...\n");
        return;
    }

    int real_index = delete_index - 1;

    printf("are you sure you want to delete %s %s (y/n) ",phone_info[real_index].first_name,phone_info[real_index].last_name);
```



```c
    void delete_contact(struct contact phone_info[],int *count){
        clear_buffer();
        char confirm = getchar();
        clear_buffer();

        if (confirm == 'y'|| confirm == 'Y'){

            for (int i = real_index; i < *count-1; i++){
                phone_info[i]=phone_info[i+1];
            }
            (*count)--;

            printf("contact deleted successfully.\n");
        }
        else{
            printf("deletion cancelled.\n");
            return;

        }
    }



void save_in_file(struct contact phone_info[],int count){
  FILE *file;
  file = fopen(filename,"wb");
  if (file == NULL){
      printf("Error: Could not open file '%s' for writing.\n", filename);
      return;
  }

  size_t item_written = fwrite(phone_info,sizeof(struct contact),count,file);

  if(item_written != count){
      printf("Error: write %zu items instead of %d\n",item_written,count);
      return;
```



```
void save_in_file(struct contact phone_info[],int count){
  if(item_written != count){

    else{
        printf("successfully saved %d contact to %s",count,filename);
    }
    fclose(file);

  }



int upload_from_file(struct contact phone_info[]){

    FILE *file;
    file = fopen(filename,"rb");
    if(file == NULL){

        printf("No existing contact file ('%s') found. Starting fresh.\n", filename);
        return 0;
    }


    size_t itemsRead = fread(phone_info, sizeof(struct contact), max_contact, file);

    if (ferror(file)) {
        fprintf(stderr, "Error reading from file '%s'.\n", file);
    }


    fclose(file);


    return (int)itemsRead;
}
void sort_contacts(struct contact phone_info[], int count) {
```



```
int upload_from_file(struct contact phone_info[]){
}

void sort_contacts(struct contact phone_info[], int count) {
    if (count < 2) {
        printf("Not enough contacts to sort.\n");
        return;
    }

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            int cmp = strcasecmp(phone_info[j].first_name, phone_info[j + 1].first_name);


            if (cmp == 0) {
                cmp = strcasecmp(phone_info[j].last_name, phone_info[j + 1].last_name);
            }


            if (cmp > 0) {
                struct contact temp = phone_info[j];
                phone_info[j] = phone_info[j + 1];
                phone_info[j + 1] = temp;
            }
        }
    }

    printf("Contacts sorted by name successfully.\n");
}
```