MICROPROJECT

Title : ***MUSIC LIBRARY MANAGEMENT***
Course name: Programming in C
Date:

Submitted to : Prof. Smitha Jacob,
Dept of Computer Science & Engineering
Submitted by: Evlin Sara Johny
Roll no: 31

INTRODUCTION

Problem Statement: Create a music library managing system to manage music collections, by creating a system to manage a personal music library, including adding, deleting, and categorizing music tracks by genre, artist, and album.

Objective: To develop a music library management system in C that allows users to add, delete, search, and display songs in their library.

Brief Overview Of the Project :

- **Adding Songs:** Implement the `addSong` function to append a new song to the library array.
- **Deleting Songs:** Implement the `deleteSong` function to remove a song by shifting the subsequent elements.
- **Searching Songs:** Implement the `searchSong` function to iterate through the library and display matching songs.
- **Displaying Library:** Implement the `displayLibrary` function to print all songs' details.

SYSTEM REQUIREMENTS

Hardware Requirements:

- **Processor:** Intel Core i3 or equivalent
- **RAM:** 2 GB or higher
- **Storage:** 50 MB of free space for the application and data
- **Display:** Monitor with at least 800x600 resolution
- **Input Devices:** Keyboard and mouse

Software Requirements:

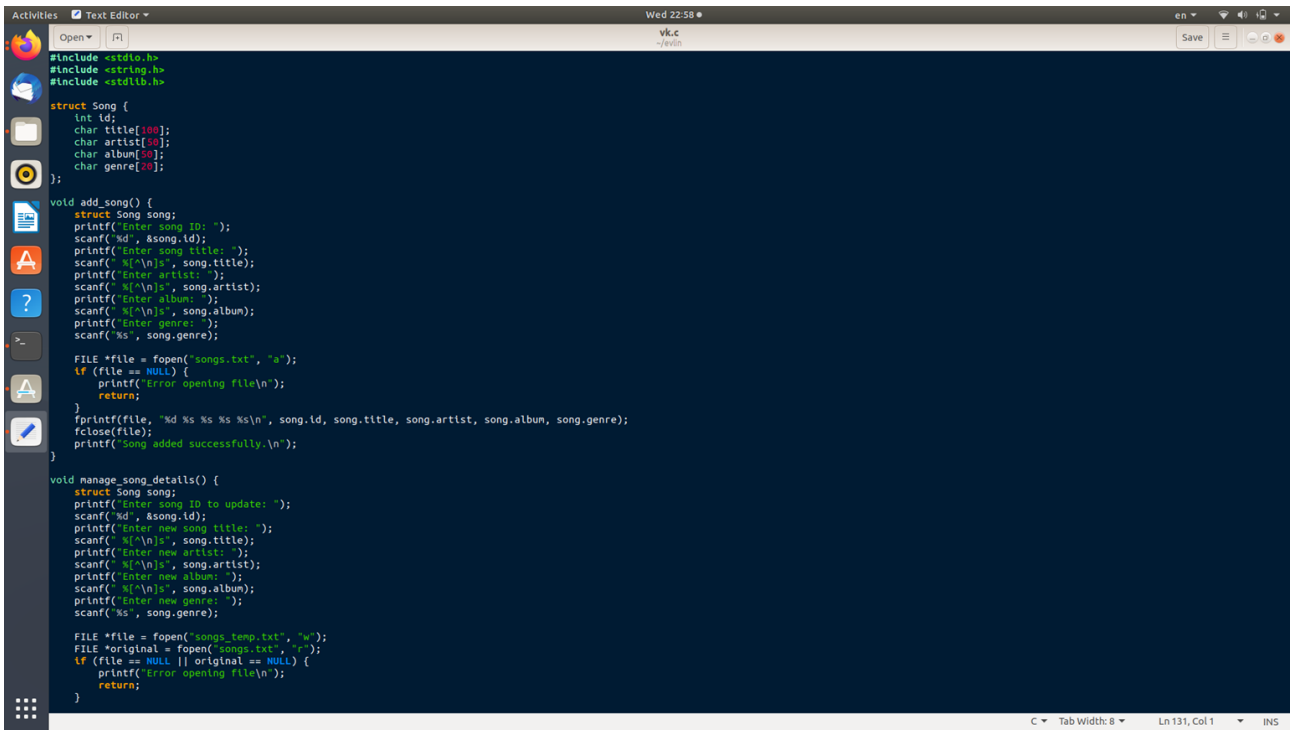
- **Operating System:** Windows, macOS, or Linux
- **Compiler:** GCC (GNU Compiler Collection) or any compatible C compiler
- **Text Editor/IDE:** Any text editor or IDE that supports C programming (e.g., Visual Studio Code, Code::Blocks, Eclipse)
- **Libraries:** Standard C library (stdlib.h, stdio.h, string.h)

DESIGN AND DEVELOPMENT

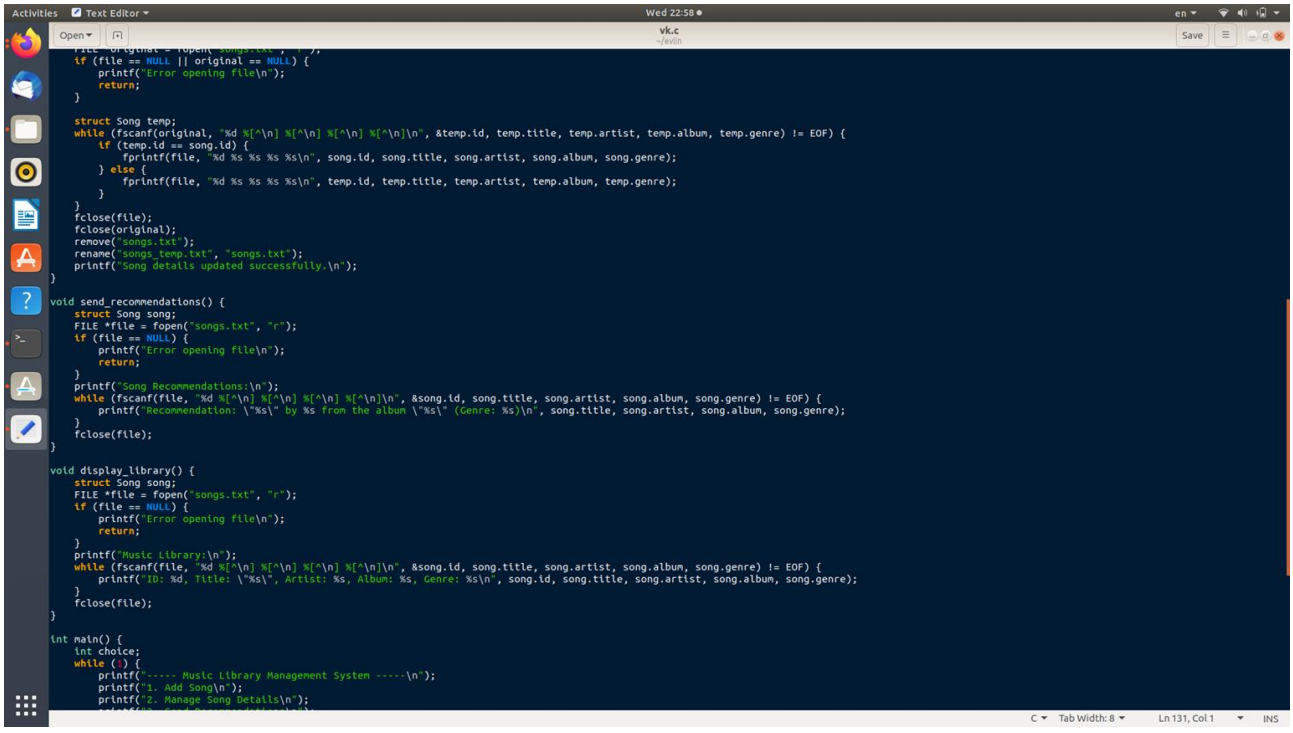
PSEUDOCODE

Step 1: START
Step 2: Defines a struct Song to store song details.
Step 3: Displays a menu and handles user input.
Step 4:Calls the appropriate functions based on user choice.
Step 5: Prompts the user to enter song details.
Step 6:Appends the song details to the songs.txt file.
Step 7:Prompts the user to enter the song ID to update.
Step 8:Reads the new details from the user.
Step 9:Writes all songs to a temporary file, updating the details for the specified song
Step 10:Replaces the original file with the updated temporary file.
Step 11:Reads all songs from the `songs.txt` file
Step 12:Prints all song details.
Step 13: STOP

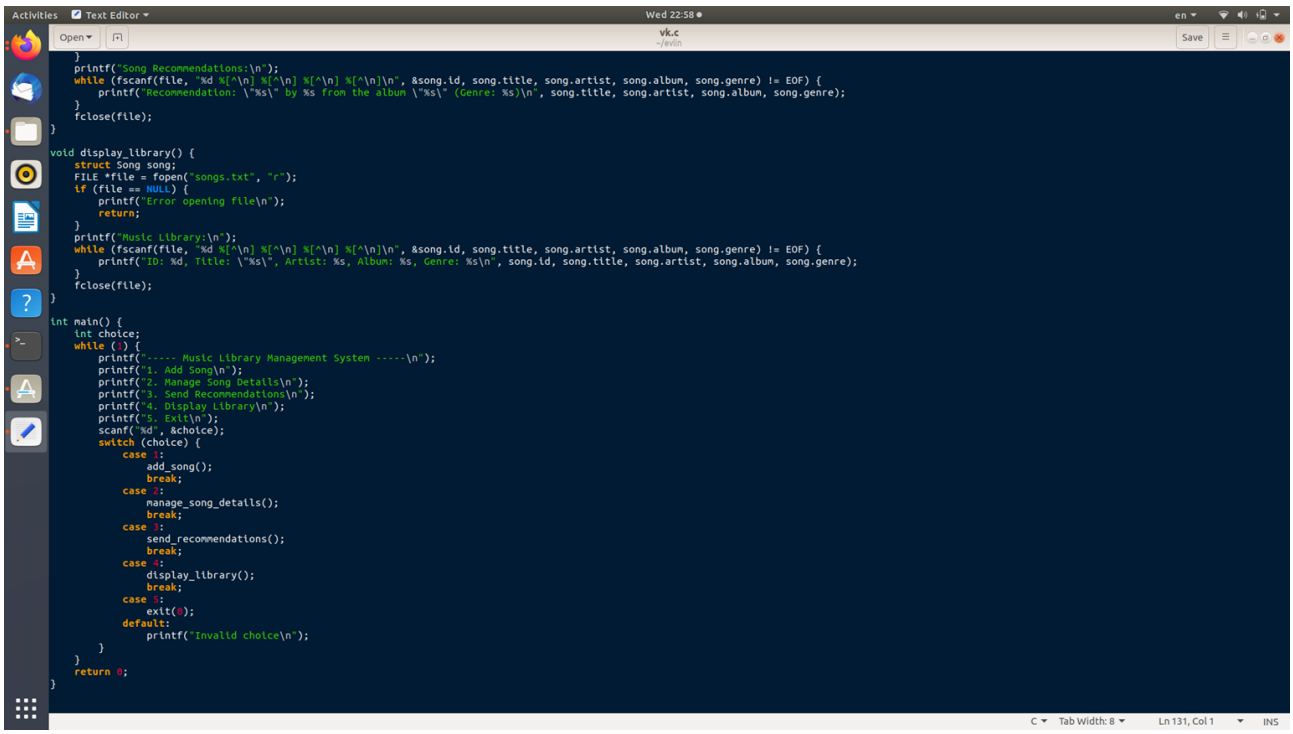
TESTING and RESULTS

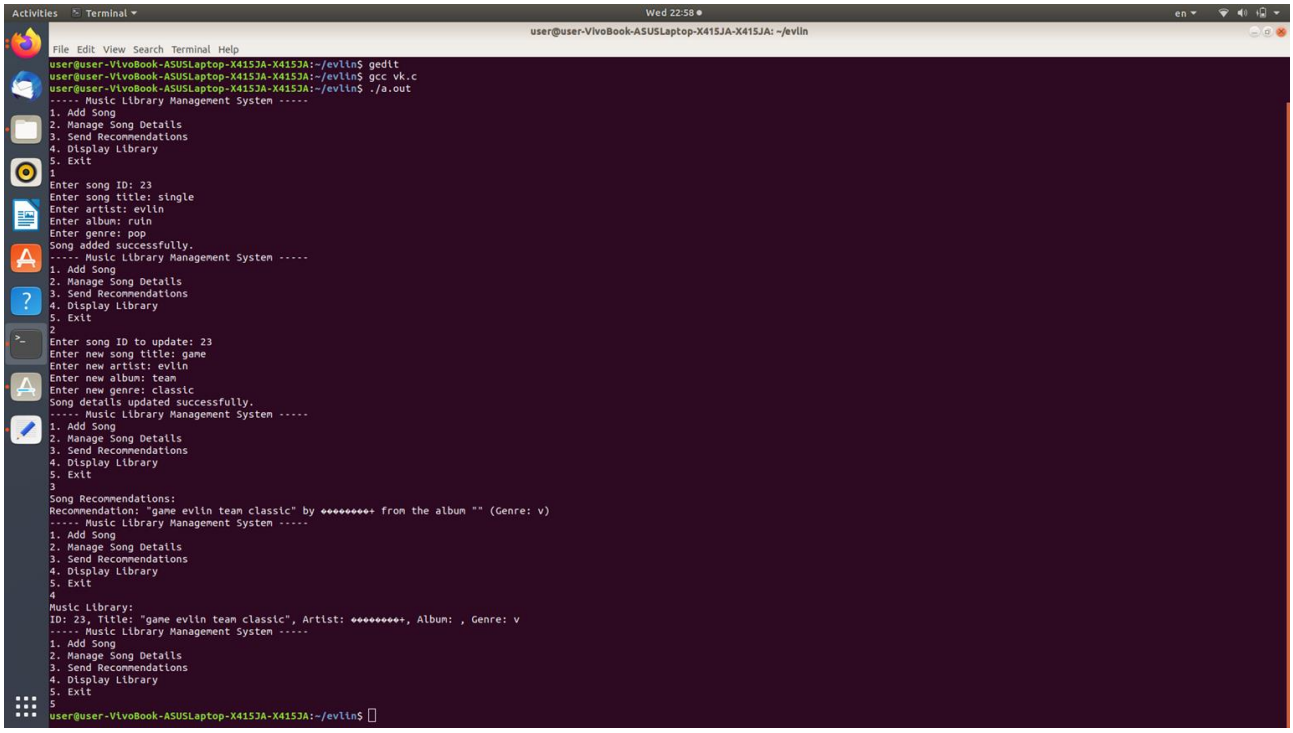



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Song {
    int id;
    char title[100];
    char artist[50];
    char album[50];
    char genre[20];
};

void add_song() {
    struct Song song;
    printf("Enter song ID: ");
    scanf("%d", &song.id);
    printf("Enter song title: ");
    scanf(" %[^\n]s", song.title);
    printf("Enter artist: ");
    scanf(" %[^\n]s", song.artist);
    printf("Enter album: ");
    scanf(" %[^\n]s", song.album);
    printf("Enter genre: ");
    scanf("%s", song.genre);

    FILE *file = fopen("songs.txt", "a");
    if (file == NULL) {
        printf("Error opening file\n");
        return;
    }
    fprintf(file, "%d %s %s %s %s\n", song.id, song.title, song.artist, song.album, song.genre);
    fclose(file);
    printf("Song added successfully.\n");
}

void manage_song_details() {
    struct Song song;
    printf("Enter song ID to update: ");
    scanf("%d", &song.id);
    printf("Enter new song title: ");
    scanf(" %[^\n]s", song.title);
    printf("Enter new artist: ");
    scanf(" %[^\n]s", song.artist);
    printf("Enter new album: ");
    scanf(" %[^\n]s", song.album);
    printf("Enter new genre: ");
    scanf("%s", song.genre);

    FILE *file = fopen("songs_temp.txt", "w");
    FILE *original = fopen("songs.txt", "r");
    if (file == NULL || original == NULL) {
        printf("Error opening file\n");
        return;
    }
```




```c
    if (file == NULL || original == NULL) {
        printf("Error opening file\n");
        return;
    }

    struct Song temp;
    while (fscanf(original, "%d %[^\n] %[^\n] %[^\n] %[^\n]\n", &temp.id, temp.title, temp.artist, temp.album, temp.genre) != EOF) {
        if (temp.id == song.id) {
            fprintf(file, "%d %s %s %s %s\n", song.id, song.title, song.artist, song.album, song.genre);
        } else {
            fprintf(file, "%d %s %s %s %s\n", temp.id, temp.title, temp.artist, temp.album, temp.genre);
        }
    }
    fclose(file);
    fclose(original);
    remove("songs.txt");
    rename("songs_temp.txt", "songs.txt");
    printf("Song details updated successfully.\n");
}

void send_recommendations() {
    struct Song song;
    FILE *file = fopen("songs.txt", "r");
    if (file == NULL) {
        printf("Error opening file\n");
        return;
    }
    printf("Song Recommendations:\n");
    while (fscanf(file, "%d %[^\n] %[^\n] %[^\n] %[^\n]\n", &song.id, song.title, song.artist, song.album, song.genre) != EOF) {
        printf("Recommendation: \"%s\" by %s from the album \"%s\" (Genre: %s)\n", song.title, song.artist, song.album, song.genre);
    }
    fclose(file);
}

void display_library() {
    struct Song song;
    FILE *file = fopen("songs.txt", "r");
    if (file == NULL) {
        printf("Error opening file\n");
        return;
    }
    printf("Music Library:\n");
    while (fscanf(file, "%d %[^\n] %[^\n] %[^\n] %[^\n]\n", &song.id, song.title, song.artist, song.album, song.genre) != EOF) {
        printf("ID: %d, Title: \"%s\", Artist: %s, Album: %s, Genre: %s\n", song.id, song.title, song.artist, song.album, song.genre);
    }
    fclose(file);
}

int main() {
    int choice;
    while (1) {
        printf("----- Music Library Management System -----\n");
        printf("1. Add Song\n");
        printf("2. Manage Song Details\n");
```


Activities
Text Editor
Wed 22:58
en
Open
vk.c
~/evlin
Save
}
printf("Song Recommendations:\n");
while (fscanf(file, "%d %[^\n] %[^\n] %[^\n] %[^\n]\n", &song.id, song.title, song.artist, song.album, song.genre) != EOF) {
printf("Recommendation: \"%s\" by %s from the album \"%s\" (Genre: %s)\n", song.title, song.artist, song.album, song.genre);
}
fclose(file);
}
void display_library() {
struct Song song;
FILE *file = fopen("songs.txt", "r");
if (file == NULL) {
printf("Error opening file\n");
return;
}
printf("Music Library:\n");
while (fscanf(file, "%d %[^\n] %[^\n] %[^\n] %[^\n]\n", &song.id, song.title, song.artist, song.album, song.genre) != EOF) {
printf("ID: %d, Title: \"%s\", Artist: %s, Album: %s, Genre: %s\n", song.id, song.title, song.artist, song.album, song.genre);
}
fclose(file);
}
int main() {
int choice;
while (1) {
printf("----- Music Library Management System -----\n");
printf("1. Add Song\n");
printf("2. Manage Song Details\n");
printf("3. Send Recommendations\n");
printf("4. Display Library\n");
printf("5. Exit\n");
scanf("%d", &choice);
switch (choice) {
case 1:
add_song();
break;
case 2:
manage_song_details();
break;
case 3:
send_recommendations();
break;
case 4:
display_library();
break;
case 5:
exit(0);
default:
printf("Invalid choice\n");
}
}
return 0;
}
C
Tab Width: 8
Ln 131, Col 1
INS


Activities
Terminal
Wed 22:58
en
user@user-VivoBook-ASUSLaptop-X415JA-X415JA: ~/evlin
File Edit View Search Terminal Help
user@user-VivoBook-ASUSLaptop-X415JA-X415JA:~/evlin$ gedit
user@user-VivoBook-ASUSLaptop-X415JA-X415JA:~/evlin$ gcc vk.c
user@user-VivoBook-ASUSLaptop-X415JA-X415JA:~/evlin$ ./a.out
----- Music Library Management System -----
1. Add Song
2. Manage Song Details
3. Send Recommendations
4. Display Library
5. Exit
1
Enter song ID: 23
Enter song title: single
Enter artist: evlin
Enter album: ruin
Enter genre: pop
Song added successfully.
----- Music Library Management System -----
1. Add Song
2. Manage Song Details
3. Send Recommendations
4. Display Library
5. Exit
2
Enter song ID to update: 23
Enter new song title: game
Enter new artist: evlin
Enter new album: team
Enter new genre: classic
Song details updated successfully.
----- Music Library Management System -----
1. Add Song
2. Manage Song Details
3. Send Recommendations
4. Display Library
5. Exit
3
Song Recommendations:
Recommendation: "game evlin team classic" by ♦♦♦♦♦♦♦♦+ from the album "" (Genre: v)
----- Music Library Management System -----
1. Add Song
2. Manage Song Details
3. Send Recommendations
4. Display Library
5. Exit
4
Music Library:
ID: 23, Title: "game evlin team classic", Artist: ♦♦♦♦♦♦♦♦+, Album: , Genre: v
----- Music Library Management System -----
1. Add Song
2. Manage Song Details
3. Send Recommendations
4. Display Library
5. Exit
5
user@user-VivoBook-ASUSLaptop-X415JA-X415JA:~/evlin$

CONCLUSION

The Music Library Management System is a comprehensive tool for managing a collection of songs. It provides essential features for adding, updating, and displaying songs, ensuring data persistence and user-friendly interactions. By following the structured program logic and leveraging file handling in C, the system offers a robust solution for music library management.

Future Enhancements for the Music Library Management System

- Graphical User Interface (GUI)
- Search Functionality
- Sorting Options
- Playlist Management
- Import/Export Functionality
- Statistics and Analytics
- Localization and Internationalization