VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI- 590 018

DSA PROJECT ON

SoundSync – Song Recommendation System

Submitted by

AADITYA V - 1RN23CY001
BHARGAVI GANGOOR - 1RN23CY010
SANJAY N - 1RN24CY049

Under the guidance of

Dr. Kiran P
Professor & Head, Dept. of CSE (CY)

Mrs. Likitha. R
Asst. Prof., Dept. of CSE (CY)

Department of Computer Science & Engineering (Cyber Security) RNS

INSTITUTE OF TECHNOLOGY

Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098 Ph:

(080)28611880, 28611881 URL: www.rnsit.ac.in

2024 – 25

Introduction:

SoundSync is a song recommendation system that leverages Data Structures and Algorithms (DSA) to categorize and manage songs across different genres and languages. This system is built using Binary Search Trees (BST) for efficient song management, Doubly Linked Lists (DLL) to organize languages within genres, and provides functionalities such as song addition, genre-based song recommendation, and URL opening for songs. The project demonstrates the application of these data structures and algorithms in creating an efficient and scalable music recommendation system. SoundSync provides a hands-on approach to understanding how DSA can be used in practical applications related to song and media management.

Problem Definition:

The main challenge addressed by this project is the efficient organization, retrieval, and recommendation of songs within a vast and growing collection The system aims to:

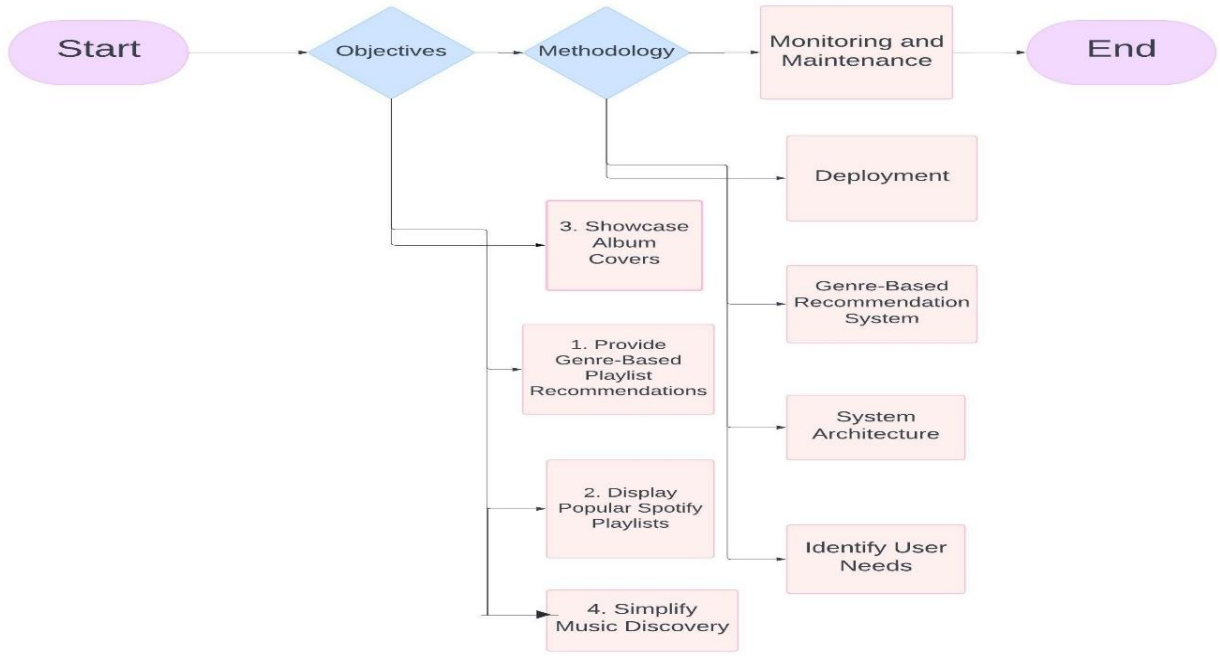
- Categorize songs based on their genre and language, allowing easy retrieval.
- Recommend songs to users based on the genre they are interested in.
- Provide a simple and effective interface for adding, recommending, and opening songs.
- Existing methods of song management often lack the scalability and flexibility needed for large datasets. This project proposes a system that uses DSA concepts such as BSTs and DLLs to overcome these limitations.

Objectives:

The key objectives of the SoundSync project are:

- Efficient Song Management: Implement a system that organizes songs in a scalable and efficient way using BSTs and DLLs.
- Genre and Language Categorization: Categorize songs by their genres and languages to enable targeted recommendations.
- Song Recommendation System: Enable users to recommend songs based on the genre they choose.
- Practical Understanding of DSA: Use this project as a learning tool to understand how DSA principles, such as BSTs, DLLs, and hashing, can be applied in real-world applications like music recommendations.

Flowchart:



DSA Concepts Used:

The project utilizes several core data structures and algorithms to achieve its functionality:

- Binary Search Trees (BST): Used to store and manage songs in each language. The BST structure ensures efficient searching, adding, and deleting of songs.
- Doubly Linked Lists (DLL): Employed to manage genres and languages. Each genre points to a linked list of languages, and each language points to a list of songs.
- Queue Management: A simple queue system manages transactions (such as adding songs) in an orderly fashion.

Working:

- The SoundSync system operates using the following structure:

    - Genres are managed using a Doubly Linked List (DLL), with each node representing a specific genre.
    - Each Genre has a list of Languages, also implemented as a DLL, where each language node contains a list of songs.
    - Songs are stored in a Binary Search Tree (BST) for each language, ensuring efficient insertion and retrieval based on the song name.
    - The system allows the user to: Add new genres, languages, and songs.
    - Recommend songs from a specified genre.
    - Open the song's URL for listening.

- The primary functionality of the system is organized as follows:

    - Genre and Language Management: Users can add new genres and languages under those genres.
    - Song Management: Users can add songs to specific languages, and songs are stored in a BST, ensuring efficient searching and management.
    - Song Recommendations: The system can recommend songs based on a selected genre, listing all songs in that genre and language.

Projected Outcomes:

The expected outcomes of the SoundSync project are:

- Functional Song Management System: A working system where songs can be categorized and efficiently managed based on their genre and language.
- Song Recommendations: A recommendation feature that allows users to receive suggestions based on the genre they select.
- Improved Understanding of DSA: A deeper understanding of how DSA can be applied to real-world applications, specifically in the context of music data management.
- Data Integrity and Immutability: The system ensures data integrity by using efficient data structures to prevent data loss and ensure consistency across the application.

The SoundSync system provides a comprehensive learning experience, offering insights into how data structures can be used to create real-world applications in media management.

CODE:

soundsync.c > ...

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

// Song structure as a Binary Search Tree (BST)
struct songlistss {
    char song[100];
    char link[200];
    struct songlistss* left;
    struct songlistss* right;
};

// Language structure as a DLL
struct languages {
    char lang[20];
    struct songlistss* artist;       //pointer referring to songlistsss strcuture.....we add song by inserting language
    struct languages* lelink;
    struct languages* rilink;
};

// Genre structure as a DLL
struct genress {
    char genre[20];
    struct languages* language;      //pointer referring to languages strcuture.....we add language by inserting genre
    struct genress* llink;
    struct genress* rlink;
};

// Type definitions
```

```c
// Type definitions
typedef struct languages* Lang;
typedef struct genress* Genre;
typedef struct songlistss* SL;

// Function declarations
Genre addgenre(Genre head, char* g);
Genre addlang(Genre head, char* l,char* g);
SL addsong(SL root, char* s, char* link);
Genre addsongtog(Genre head, char* g, char* l, char* s, char* link);
void recommend(Genre head, char* g);
void display(Genre head);
void open_url(const char* s);

// Add a new genre
Genre addgenre(Genre head, char* g) {
    Genre temp = (Genre)malloc(sizeof(struct genress));
    strcpy(temp->genre, g);
    temp->language = NULL;
    temp->rlink = NULL;
    temp->llink = NULL;

    if (head == NULL) {
        return temp; // First genre
    }

    Genre cur = head;
    while (cur->rlink != NULL) {
        if (strcmp(cur->genre, g) == 0) { // Check if genre exists
            printf("Genre '%s' already exists.\n", g);
```

```
            free(temp);
            return head;
        }
        cur = cur->rlink;
    }
     // after traversing all nodes,checks for the genre in the last node
    if (strcmp(cur->genre, g) == 0) {
        printf("Genre '%s' already exists.\n", g);
        free(temp);
        return head;
    }

    cur->rlink = temp;
    temp->llink = cur;
    return head;


// Add a new language
Genre addlang(Genre head, char* g, char* l) {
    Genre cur = head;// Find the genre
    while (cur != NULL && strcmp(cur->genre, g) != 0) {
        cur = cur->rlink;
    }
    if (cur == NULL) {
        printf("Genre '%s' not found.\n", g);
        return head;
    }
    Lang langHead = cur->language;// Acts like Head node of Lang DLL
    Lang temp = (Lang)malloc(sizeof(struct languages));
    strcpy(temp->lang, l);
    temp->artist = NULL;
    temp->lelink = NULL;
    temp->rilink = NULL;
    if (langHead == NULL) {
        cur->language = temp; // First language
        return head;
    }
    Lang langCur = langHead;
    while (langCur->rilink != NULL) {
        if (strcmp(langCur->lang, l) == 0) { // Language already exists
            printf("Language '%s' already exists in genre '%s'.\n", l, g);
            free(temp); // Avoid memory leak
            return head;}
        langCur = langCur->rilink;
    }
    if (strcmp(langCur->lang, l) == 0) {
        printf("Language '%s' already exists in genre '%s'.\n", l, g);
        free(temp);
        return head;
    }
    langCur->rilink = temp;
    temp->lelink = langCur;
    return head;
}
```

```
SL addsong(SL root, char* s, char* link) {
    if (root == NULL) {
        SL temp = (SL)malloc(sizeof(struct songlistss));
        strcpy(temp->song, s);
        strcpy(temp->link, link);
        temp->left = NULL;
        temp->right = NULL;
        return temp; }
    if (strcmp(s, root->song) < 0) {
        root->left = addsong(root->left, s, link);
    } else if (strcmp(s, root->song) > 0) {
        root->right = addsong(root->right, s, link);
    } else {
        printf("Song '%s' already exists.\n", s);}
    return root;
}
// Add a song to a genre and language
Genre addsongtog(Genre head, char* g, char* l, char* s, char* link) {
    Genre cur = head;
    while (cur != NULL && strcmp(cur->genre, g) != 0) {
        cur = cur->rlink; }
    if (cur == NULL) {
        printf("Genre '%s' not found.\n", g);
        return head; }
    Lang langCur = cur->language;
    while (langCur != NULL && strcmp(langCur->lang, l) != 0) {
        langCur = langCur->rilink; }
    if (langCur == NULL) {
        printf("Language '%s' not found in genre '%s'.\n", l, g);
        return head;
    }
    langCur->artist = addsong(langCur->artist, s, link);
    return head;
}
```

```c
void recommend(Genre head, char* g) {
    Genre cur = head;
    while (cur != NULL && strcmp(cur->genre, g) != 0) {
        cur = cur->rlink;
    }
    if (cur != NULL) {
        printf("\nSongs recommended in genre '%s':\n", g);
        Lang langc = cur->language;
        while (langc != NULL) {
            printf("Language: %s\n", langc->lang);
            SL songc = langc->artist;
            while (songc != NULL) {
                printf("\tSong: %s - Link: %s\n", songc->song, songc->link);
                songc = songc->right;
            }
            langc = langc->rilink;
        }
    } else {
        printf("Genre '%s' not found.\n", g);
    }
}

// Display the collection
void display(Genre head) {
    if (head == NULL) {
        printf("No genres available.\n");
        return;
    }

    Genre cur = head;
    while (cur != NULL) {
        printf("\nGenre: %s\n", cur->genre);
        Lang langc = cur->language;
        while (langc != NULL) {
            printf("\tLanguage: %s\n", langc->lang);
            SL songc = langc->artist;
            while (songc != NULL) {
```

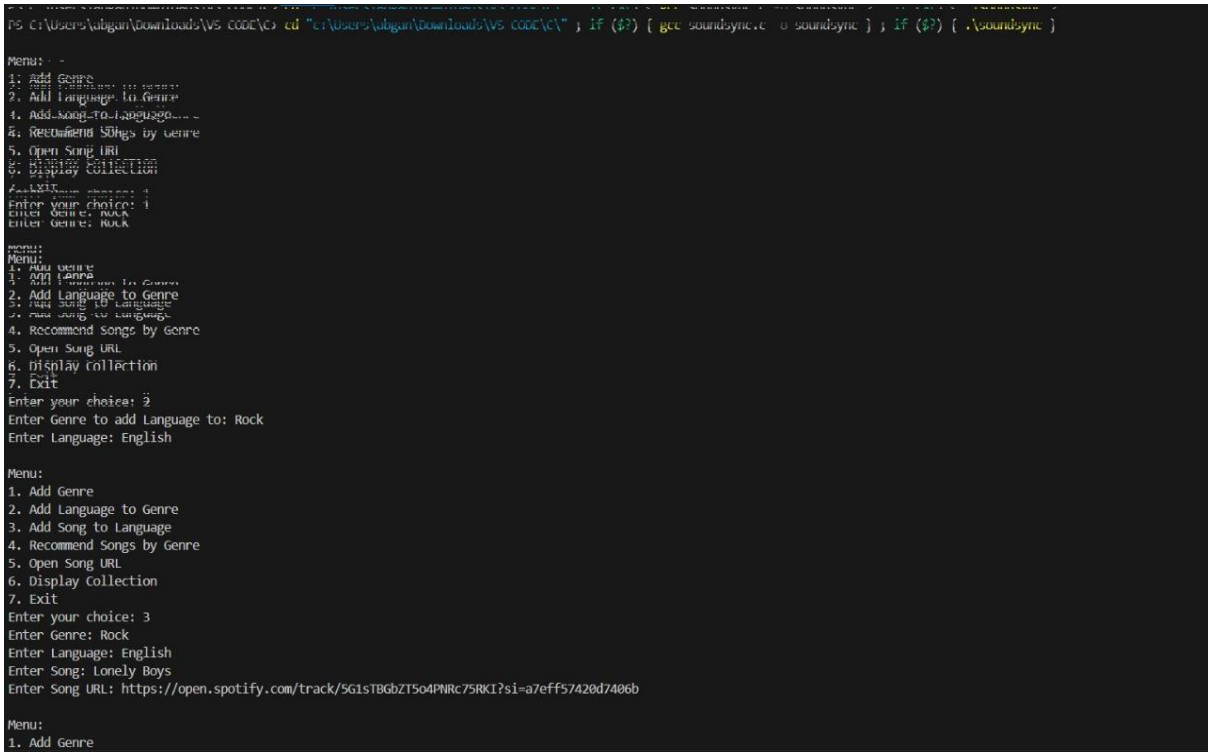
```c
            while (songc != NULL) {
                songc = songc->right;
            }
            langc = langc->rilink;
        }
        cur = cur->rlink;
    }
}

// Open a song URL
void open_url(const char* url) {
    char command[256];
    snprintf(command, sizeof(command), "start %s", url); // Format command for system call
    system(command); // Execute command
}

// Main function
int main() {
    Genre gHead = NULL;
    int choice;
    char genre[20], language[20], song[100], link[200];

    while (1) {
        printf("\nMenu:\n");
        printf("1. Add Genre\n");
        printf("2. Add Language to Genre\n");
        printf("3. Add Song to Language\n");
        printf("4. Recommend Songs by Genre\n");
        printf("5. Open Song URL\n");
        printf("6. Display Collection\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();
```

OUTPUT:




```
Menu:
1. Add Genre
2. Add Language to Genre
3. Add Song to Language
4. Recommend Songs by Genre
5. Open Song URL
6. Display Collection
7. Exit
Enter your choice: 4
Enter Genre to recommend songs: Rock

Songs recommended in genre 'Rock':
Language: English
        Song: Lonely Boys - Link: https://open.spotify.com/track/5G1sTBGbZT5o4PNRc75RKI?si=a7eff57420d7406b

Menu:
1. Add Genre
2. Add Language to Genre
3. Add Song to Language
4. Recommend Songs by Genre
5. Open Song URL
6. Display Collection
7. Exit
Enter your choice: 6

Genre: Rock
        Language: English
                Song: Lonely Boys - Link: https://open.spotify.com/track/5G1sTBGbZT5o4PNRc75RKI?si=a7eff57420d7406b

Menu:
1. Add Genre
2. Add Language to Genre
3. Add Song to Language
4. Recommend Songs by Genre
5. Open Song URL
6. Display Collection
7. Exit
Enter your choice: 5
Enter Song URL to open: https://open.spotify.com/track/5G1sTBGbZT5o4PNRc75RKI?si=a7eff57420d7406b
```