Spotify Music Playlist Manager

Using linked list

Team members

Murali
Purushotham
Mahesh
Poorna
Rajashekar

Introduction

- A music playlist manager allows users to create, modify, and navigate playlists.
- Using a linked list provides dynamic memory allocation and efficient song management.

Problem Statement: Scalable and Personalized Playlists

Challenge

Design a system for personalized music recommendations at scale.

Requirements

- Efficient data storage and retrieval.
- Fast and relevant recommendations.
- Seamless playlist management and sharing.

Context

Spotify has 574M users (Q1 2024) and 8B+ playlists.

Why Linked List?

- • Efficient Insert/Delete operations
- • Dynamic memory allocation
- • No fixed size like arrays
- • Enables seamless navigation through the playlist

Types of Linked Lists

- Singly Linked List - One-directional navigation
- Doubly Linked List - Bi-directional navigation
- Circular Linked List - Looped playlist functionality

Implementation Details

- Each song is represented as a node in the linked list.
- Each node contains:
- Song Name
- Artist Name
- Pointer to Next (and Previous in Doubly Linked List)

Playlist Operations

- • Add Song
- • Remove Song
- • Display Playlist
- • Move to Next/Previous Song
- • Shuffle Playlist

Time Complexity Analysis

- • Add Song: O(1) (if inserting at the end)
- • Remove Song: O(n) (search required)
- • Display Playlist: O(n)
- • Move to Next/Previous: O(1)

Real-world Applications

- • Used in music streaming services like Spotify & Apple Music
- • Helps in efficient playlist organization
- • Supports dynamic playlist modifications

Conclusion & Future Scope

- Linked Lists provide an efficient way to manage playlists dynamically.
- Future Enhancements:
- • Playlist Sorting
- • Integration with Cloud Storage
- • AI-based Song Recommendations

PROGRAM

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

// Structure for a Song
typedef struct Song {
    char title[50];
    char artist[50];
    float duration;
    struct Song* next;
    struct Song* prev;
} Song;

// Head and Tail Pointers
Song* head = NULL;
Song* tail = NULL;
Song* current = NULL;

// Function to create a new song
Song* createSong(char title[], char artist[], float duration) {
    Song* newSong = (Song*)malloc(sizeof(Song));
    strcpy(newSong->title, title);
    strcpy(newSong->artist, artist);
    newSong->duration = duration;
    newSong->next = NULL;
    newSong->prev = NULL;
    return newSong;
}

// Add song to the playlist
void addSong(char title[], char artist[], float duration) {
    Song* newSong = createSong(title, artist, duration);
    if (head == NULL) {
        head = newSong;
        tail = newSong;
        current = newSong;
    } else {
        tail->next = newSong;
        newSong->prev = tail;
        tail = newSong;
    }
    printf("Added: %s by %s\n", title, artist);
}
```

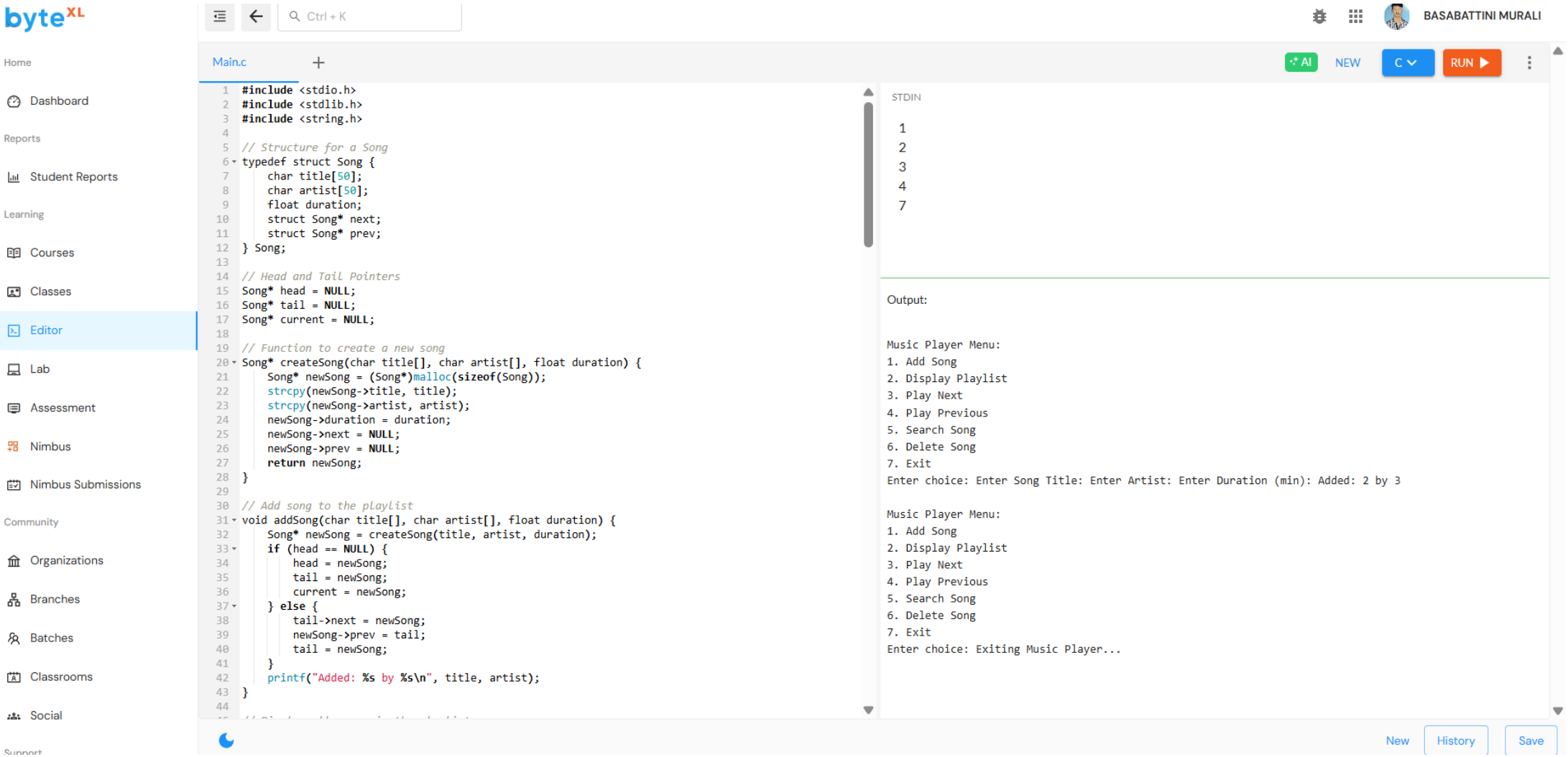
```c
// Display all songs in the playlist
void displayPlaylist() {
    Song* temp = head;
    if (temp == NULL) {
        printf("Playlist is empty!\n");
        return;
    }
    printf("\n--- Playlist ---\n");
    while (temp != NULL) {
        printf("%s - %s (%.2f min)\n", temp->title, temp->artist, temp->duration);
        temp = temp->next;
    }
    printf("----------------\n");
}

// Play next song
void playNext() {
    if (current == NULL || current->next == NULL) {
        printf("End of Playlist!\n");
    } else {
        current = current->next;
        printf("Now Playing: %s by %s\n", current->title, current->artist);
    }
}

// Play previous song
void playPrev() {
    if (current == NULL || current->prev == NULL) {
        printf("Start of Playlist!\n");
    } else {
        current = current->prev;
        printf("Now Playing: %s by %s\n", current->title, current->artist);
    }
}

// Search for a song
void searchSong(char title[]) {
    Song* temp = head;
    while (temp != NULL) {
        if (strcmp(temp->title, title) == 0) {
            printf("Found: %s by %s\n", temp->title, temp->artist);
            return;
        }
```

```
            return;
        }
        temp = temp->next;
    }
    printf("Song not found!\n");
}

// Delete a song from the playlist
void deleteSong(char title[]) {
    Song* temp = head;
    while (temp != NULL) {
        if (strcmp(temp->title, title) == 0) {
            if (temp->prev) temp->prev->next = temp->next;
            if (temp->next) temp->next->prev = temp->prev;
            if (temp == head) head = temp->next;
            if (temp == tail) tail = temp->prev;
            if (temp == current) current = temp->next ? temp->next : temp->prev;
            free(temp);
            printf("Deleted: %s\n", title);
            return;
        }
        temp = temp->next;
    }
    printf("Song not found!\n");
}

// Main Menu
void menu() {
    printf("\nMusic Player Menu:\n");
    printf("1. Add Song\n");
    printf("2. Display Playlist\n");
    printf("3. Play Next\n");
    printf("4. Play Previous\n");
    printf("5. Search Song\n");
    printf("6. Delete Song\n");
    printf("7. Exit\n");
}

int main() {
    int choice;
    char title[50], artist[50];
    float duration;

    while (1) {
```

```
    while (1) {
        menu();
        printf("Enter choice: ");
        scanf("%d", &choice);
        getchar();  // To consume newline character

        switch (choice) {
            case 1:
                printf("Enter Song Title: ");
                fgets(title, 50, stdin);
                title[strcspn(title, "\n")] = 0;
                printf("Enter Artist: ");
                fgets(artist, 50, stdin);
                artist[strcspn(artist, "\n")] = 0;
                printf("Enter Duration (min): ");
                scanf("%f", &duration);
                addSong(title, artist, duration);
                break;

            case 2:
                displayPlaylist();
                break;

            case 3:
                playNext();
                break;

            case 4:
                playPrev();
                break;

            case 5:
                printf("Enter Song Title to Search: ");
                fgets(title, 50, stdin);
                title[strcspn(title, "\n")] = 0;
                searchSong(title);
                break;

            case 6:
                printf("Enter Song Title to Delete: ");
                fgets(title, 50, stdin);
                title[strcspn(title, "\n")] = 0;
                deleteSong(title);
```

```
            printf("Enter Artist: ");
            fgets(artist, 50, stdin);
            artist[strcspn(artist, "\n")] = 0;
            printf("Enter Duration (min): ");
            scanf("%f", &duration);
            addSong(title, artist, duration);
            break;

        case 2:
            displayPlaylist();
            break;

        case 3:
            playNext();
            break;

        case 4:
            playPrev();
            break;

        case 5:
            printf("Enter Song Title to Search: ");
            fgets(title, 50, stdin);
            title[strcspn(title, "\n")] = 0;
            searchSong(title);
            break;

        case 6:
            printf("Enter Song Title to Delete: ");
            fgets(title, 50, stdin);
            title[strcspn(title, "\n")] = 0;
            deleteSong(title);
            break;

        case 7:
            printf("Exiting Music Player...\n");
            exit(0);

        default:
            printf("Invalid choice! Try again.\n");
    }
  }
  return 0;
}
```

output


byteXL
Ctrl + K
BASABATTINI MURALI
Home
Dashboard
Reports
Student Reports
Learning
Courses
Classes
Editor
Lab
Assessment
Nimbus
Nimbus Submissions
Community
Organizations
Branches
Batches
Classrooms
Social
Main.c
AI
NEW
C
RUN
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Structure for a Song
typedef struct Song {
char title[50];
char artist[50];
float duration;
struct Song* next;
struct Song* prev;
} Song;
// Head and Tail Pointers
Song* head = NULL;
Song* tail = NULL;
Song* current = NULL;
// Function to create a new song
Song* createSong(char title[], char artist[], float duration) {
Song* newSong = (Song*)malloc(sizeof(Song));
strcpy(newSong->title, title);
strcpy(newSong->artist, artist);
newSong->duration = duration;
newSong->next = NULL;
newSong->prev = NULL;
return newSong;
}
// Add song to the playlist
void addSong(char title[], char artist[], float duration) {
Song* newSong = createSong(title, artist, duration);
if (head == NULL) {
head = newSong;
tail = newSong;
current = newSong;
} else {
tail->next = newSong;
newSong->prev = tail;
tail = newSong;
}
printf("Added: %s by %s\n", title, artist);
}
STDIN
1
2
3
4
7
Output:
Music Player Menu:
1. Add Song
2. Display Playlist
3. Play Next
4. Play Previous
5. Search Song
6. Delete Song
7. Exit
Enter choice: Enter Song Title: Enter Artist: Enter Duration (min): Added: 2 by 3
Music Player Menu:
1. Add Song
2. Display Playlist
3. Play Next
4. Play Previous
5. Search Song
6. Delete Song
7. Exit
Enter choice: Exiting Music Player...
New
History
Save

THANK YOU ...