LAPORAN TAHAP 3: TUGAS BESAR STRUKTUR DATA

‘Music Player’ Console Application

By “ngantuk ya bram nama kelompoknya”
Zaky Muhammad Fauzi 103052400064
Brama Hartoyo 103052400030

LATAR BELAKANG

Mata kuliah Struktur Data memiliki tujuan utama untuk memberikan pemahaman mendalam mengenai berbagai jenis penyimpanan dan pengelolaan data, seperti Linked List, Stack, Queue, Tree, dan Graph. Teori-teori ini tidak hanya bersifat konseptual, tetapi diharapkan mampu diterapkan oleh mahasiswa untuk menyelesaikan permasalahan nyata dalam pengembangan perangkat lunak.

Salah satu studi kasus yang relevan dengan pengelolaan data yang kompleks adalah aplikasi pemutar musik modern. Layanan seperti Spotify atau Apple Music menuntut pengelolaan koleksi lagu dalam skala besar, pengaturan playlist yang fleksibel, antrian pemutaran yang dinamis, serta kemampuan rekomendasi lagu. Oleh karena itu, dalam tugas besar ini, kami diminta untuk membangun sebuah replika aplikasi pemutar musik berbasis konsol.

Aplikasi ini akan mensimulasikan peran Admin (pengelola data) dan User (pengguna fitur), di mana efisiensi penyimpanan dan akses data menjadi kunci utama. Melalui proyek ini, kami akan melatih kemampuan analisis dalam memilih struktur data yang paling kompatibel dengan kebutuhan fitur, seperti penggunaan Multi Linked List untuk relasi data atau Queue untuk antrian lagu.

ANALISIS MASALAH

Berdasarkan deskripsi tugas, kami mengidentifikasi beberapa tantangan dan kebutuhan sistem yang harus diselesaikan:

1. Manajemen data lagu yang kompleks aplikasi harus mampu menyimpan data lagu dengan atribut seperti Judul, Artis, Genre, Album, dan Tahun menggunakan tipe data bentukan (record). Tantangannya adalah bagaimana menyimpan data ini agar mudah ditambah, diubah, dan dihapus oleh Admin, serta dicari dengan cepat oleh User.
2. Relasi antara Library Utama dan Playlist User terdapat ketergantungan data yang ketat. Jika Admin mengubah atau menghapus data lagu di library utama, perubahan tersebut harus tercermin secara otomatis di semua playlist milik User yang memuat lagu tersebut. Ini memerlukan struktur data yang memungkinkan referensi atau pointer yang efisien tanpa menduplikasi data lagu secara fisik yang akan memboroskan memori.
3. Logika pemutaran lagu (Next/Prev) Fitur navigasi lagu memiliki logika percabangan:
    a. Jika sedang dalam mode Playlist, lagu selanjutnya mengikuti urutan playlist.
    b. Jika sedang dalam mode Library (luar playlist), lagu selanjutnya haruslah lagu yang "mirip" (berdasarkan prioritas kesamaan Artis atau Genre). Hal ini menuntut penggunaan algoritma pencarian atau struktur data relasional (seperti Graph atau Tree) untuk menentukan kemiripan lagu.

4. Peran ganda (Admin vs User) sistem harus membedakan akses. Admin memiliki hak mutlak terhadap database lagu, sementara User fokus pada manipulasi playlist pribadi dan pemutaran lagu.

PERENCANAAN FITUR

Untuk menyelesaikan masalah di atas, hasil diskusi dengan asisten praktikum MRR, kelompok kami menyediakan fitur sebagai berikut.

1. Struktur data utama

Fitur	Struktur Data yang Dipilih
Penyimpanan Lagu	Doubly Linked List
Data Playlist User	Multi Linked List
Antrian Lagu	Queue
Riwayat Lagu	Stack
Rekomendasi Lagu Mirip	Graph/Pencarian Array

2. Rancangan tipe data (record/struct)
    a. ID Lagu (integer).
    b. Judul (string).
    c. Penyanyi (string).
    d. Genre (string).
    e. Tahun (integer).
3. Skenario algoritma “mirip” (similarity logic)
   Untuk skenario algoritma lagu yang serupa, lagu yang dipilih dengan mengecek dari prioritas sebagai berikut.
    a. Prioritas 1: Lagu dengan nama penyanyi yang sama.
    b. Prioritas 2: Jika tidak ada penyanyi sama, cari lagu dengan genre yang sama.
    c. Fallback: Jika tidak ada keduanya, putar lagu secara acak/urutan ID.
4. Alat pengembangan
   Kami akan menggunakan Github sebagai tempat penyimpanan kode, dokumentasi, dan kolaborasi versi (version control).

IMPLEMENTASI ADT

Berdasarkan analisis masalah dan perencanaan fitur, struktur data yang dipilih adalah sebagai berikut:

1. Doubly Linked List (DLL) sebagai penyimpanan library lagu karena mendukung penambahan, penghapusan, dan update data di posisi mana pun dan dua arah yang memudahkan pencarian next/prev efisien untuk data berurutan (library lagu). Struktur record lagu:
    a. ID Lagu (integer).
    b. Judul (string).
    c. Penyanyi (string).
    d. Genre (string).
    e. Tahun (integer).

2. Multi Linked List (MLL) sebagai playlist user karena playlist tidak menduplikasi data lagu dan playlist hanya menyimpan pointer ke node DLL (library), jika data lagu di library berubah maka playlist ikut berubah otomatis dan jika lagu dihapus admin maka referensi playlist menjadi aman dihapus. Struktur playlistnya:
    a. song (adrLagu).
    b. next (adrPlay).
3. Queue antrian lagu untuk simulasi tambahkan ke antrian pemutaran berikutnya. Mendukung enqueue (tambah ke belakang) dan dequeue (ambil dari depan).
4. Stack riwayat lagu untuk menyimpan lagu yang sudah diputar dengan fitur kembali ke lagu sebelumnya (undo play) dengan operasi push (tambah ke top) dan pop (kembali/undo).
5. Rekomendasi lagu berdasar aturan kemiripan untuk fitur Next di luar playlist dengan memperhatikan prioritas dari penyanyi yang sama atau genre yang sama dan fallback (lagu pertama di library).

IMPLEMENTASI FITUR

Berikut fitur lengkap yang telah selesai dibuat berdasarkan kode yang sudah disusun:

1. Fitur Peran Admin
    a. Menambah lagu baru ke library. Lagu disimpan sebagai node baru DLL.
    b. Menampilkan semua lagu yang tersimpan di library dengan melakukan traversal DLL dari first hingga last menggunakan prosedur showLibrary.
    c. Menyediakan pencarian lagu berdasarkan ID lagu, Judul lagu, Artis, atau Genre.
    d. Mengubah data lagu dengan mengupdate record pada node DLL seperti judul, artis, genre, dan tahun. Karena playlist menyimpan pointer ke DLL maka playlist akan ikut berubah otomatis.
    e. Menghapus lagu dari library dengan menghapus node DLL menggunakan prosedur deleteLagu sehingga node playlist yang menunjuk ke lagu itu juga terhapus agar tidak ada pointer menggantung (dangling pointer).
    f. Mengurutkan library berdasarkan ID lagu secara ascending menggunakan algoritma bubble sort yang diterapkan langsung pada node DLL melalui prosedur sortLibraryById.
2. Fitur Peran User:
    a. Melihat seluruh daftar lagu yang tersedia di library dengan DLL.
    b. Mencari lagu berdasarkan ID dan hasil pencarian disimpan sebagai current song yang siap diputar.
    c. Memutar lagu secara simulatif dan informasi lagu (judul, artis, genre, tahun) ditampilkan ke layar. Lagu yang diputar otomatis dimasukkan ke stack (riwayat lagu) menggunakan prosedur push.
    d. Menghentikan pemutaran lagu menggunakan prosedur stopSong, yang mengubah status pemutaran menjadi tidak aktif..
    e. Next lagu jika user tidak berada di playlist akan mencari lagu yang mirip berdasarkan artis, jika tidak ada maka mencari berdasarkan genre, jika tetap tidak ada maka fallback ke lagu pertama.

f. Next song (mode library/rekomendasi lagu yang mirip) jika user tidak sedang berada di playlist, fitur Next akan mencari lagu dengan artis yang sama. Jika tidak ditemukan, mencari lagu dengan genre yang sama. Jika tetap tidak ada, berpindah ke lagu berikutnya di library dengan logika yang diimplementasikan melalui fungsi nextSimilar.
g. Previous song (riwayat lagu) kembali ke lagu sebelumnya menggunakan stack riwayat lagu. Lagu terakhir yang diputar diambil menggunakan operasi pop.
h. Menambahkan lagu ke playlist pribadi. Playlist disimpan sebagai MLL yang hanya menyimpan pointer ke node lagu di DLL, sehingga tidak terjadi duplikasi data.
i. Menampilkan seluruh lagu yang ada dalam playlist melalui MLL menggunakan prosedur showPlaylist.
j. Menghapus lagu tertentu dari playlist dengan menghapus node playlist yang menunjuk ke lagu tersebut menggunakan prosedur removeFromPlaylist.
k. Validasi lagu untuk mengecek apakah sebuah lagu sudah ada di playlist atau belum menggunakan fungsi isInPlaylist, untuk mencegah duplikasi logis.
l. Menambahkan lagu ke antrian pemutaran yang diimplementasikan menggunakan struktur Queue (FIFO) melalui prosedur enqueue.
m. Memutar lagu dari antrian berdasarkan urutan masuk antrian menggunakan prosedur dequeue.
n. Menampilkan seluruh lagu yang ada di antrian beserta urutannya menggunakan prosedur showQueue.
o. Menghapus lagu tertentu dari antrian berdasarkan ID menggunakan prosedur removeFromQueue.
p. Validasi lagu dalam antrian mengecek apakah sebuah lagu sudah ada di antrian menggunakan fungsi isInQueue.

3. Struktur Menu Main program menyediakan dua mode yaitu Admin Mode dan User Mode. Setiap mode memiliki menu masing-masing sesuai fitur.

IMPLEMENTASI KODE

Berikut prosedur dan fungsi yang telah selesai dibuat menjadi pseudocode:

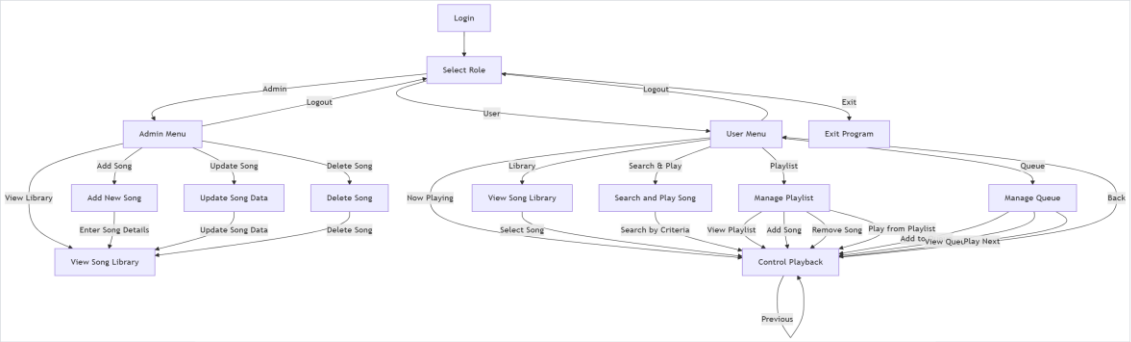
1. Procedure createListLagu(in/out L: ListLagu). IS: membuat list lagu kosong, FS: L.first = NILL dan L.last = NILL.
2. Function allocateLagu(x: Lagu) return adrLagu. IS: mengalokasikan node lagu baru, FS: mengembalikan alamat node lagu p dengan info = x.
3. Procedure insertLagu(in/out L: ListLagu, x: Lagu). IS: menambahkan lagu ke akhir list, FS: lagu x tersimpan sebagai elemen terakhir list.
4. Procedure showLibrary(in L: ListLagu). IS: Menampilkan seluruh lagu di library, FS: Semua data lagu ditampilkan ke layar.
5. Function findLaguByTitle(L: ListLagu, judul : string) return adrLagu. IS: Mencari lagu berdasarkan judul, FS: Mengembalikan alamat lagu jika ditemukan, NILL jika tidak.
6. Function findLaguByArtist(L: ListLagu, artiS: String) return adrLagu. IS: Mencari lagu berdasarkan artis, FS: Mengembalikan alamat lagu atau NILL.
7. Function findLaguByGenre(L: ListLagu, genre: string) return adrLagu. IS: Mencari lagu berdasarkan genre, FS: Mengembalikan alamat lagu atau NILL.

8. Procedure sortLibraryById(in/out L: ListLagu). IS: Mengurutkan lagu berdasarkan ID secara ascending, FS: List lagu terurut berdasarkan ID.
9. Function findLaguById(L: ListLagu, id: integer) return adrLagu. IS: Mencari lagu berdasarkan ID, FS: Mengembalikan alamat lagu atau NILL.
10. Procedure updateLagu(in/out p: adrLagu, in baru : Lagu). IS: Mengubah data lagu, FS: Data lagu pada node p terupdate.
11. Procedure deleteLagu(in/out L: ListLagu, id: integer). IS: Menghapus lagu dari library, FS: Lagu dengan ID tertentu terhapus dari list.
12. Procedure createPlaylist(in/out P: Playlist). IS: Membuat playlist kosong, FS: P.first = NILL dan P.last = NILL.
13. Procedure addToPlaylist(in/out P: Playlist, song: adrLagu). IS: Menambahkan lagu ke playlist, FS: Lagu tersimpan di akhir playlist.
14. Procedure removeFromPlaylist(in/out P: Playlist, id: integer). IS: Menghapus lagu dari playlist, FS: Lagu dengan ID tertentu terhapus dari playlist.
15. Function isInPlaylist(P: Playlist, id: integer) return boolean. IS: Mengecek keberadaan lagu di playlist, FS: true jika ada, false jika tidak.
16. Procedure showPlaylist(in P: Playlist). IS: Menampilkan seluruh lagu dalam playlist, FS: Isi playlist ditampilkan ke layar.
17. Procedure removeFromAllPlaylists(in/out P: Playlist, in id: integer). IS: Menghapus seluruh lagu dengan ID tertentu dari playlist, FS: Tidak ada lagu dengan ID tersebut di dalam playlist.
18. Procedure createStack(in/out S: Stack). IS: Membuat stack kosong, FS: S.top = NILL
19. Procedure push(in/out S: Stack, x: adrLagu). IS: Menambahkan lagu ke stack, FS: Lagu berada di puncak stack.
20. Function pop(in/out S: Stack) return adrLagu. IS: Mengambil lagu teratas dari stack, FS: Lagu teratas terambil dan stack berkurang.
21. Procedure clearSongFromStacks(in/out S: Stack, in id: integer). IS: Menghapus lagu tertentu dari riwayat pemutaran (stack), FS: Stack tidak mengandung lagu dengan ID tersebut.
22. Procedure createQueue(in/out Q: Queue). IS: Membuat queue kosong, FS: Q.head = NILL dan Q.tail = NILL.
23. Procedure enqueue(in/out Q: Queue, x: adrLagu). IS: Menambahkan lagu ke antrian, FS: Lagu berada di belakang queue.
24. Function dequeue(in/out Q: Queue) return adrLagu. IS: Mengambil lagu dari depan antrian, FS: Lagu terdepan terambil dari queue.
25. Function isInQueue(Q: Queue, id: integer) return boolean. IS: Mengecek lagu di antrian, FS: true jika ada, false jika tidak.
26. Procedure showQueue(in Q: Queue). IS: Menampilkan seluruh isi antrian lagu, FS: Seluruh lagu dalam queue ditampilkan ke layar.
27. Procedure removeFromQueue(in/out Q: Queue, in id: integer). IS: Menghapus lagu tertentu dari antrian, FS: Lagu dengan ID tersebut tidak ada di queue.
28. Procedure playSong(song: adrLagu, in/out S: Stack). IS: Memutar lagu dan menyimpan ke history, FS: Lagu diputar dan masuk ke stack.
29. Procedure stopSong(in/out isPlaying : boolean). IS: Menghentikan pemutaran lagu, FS: Status pemutaran menjadi tidak aktif.

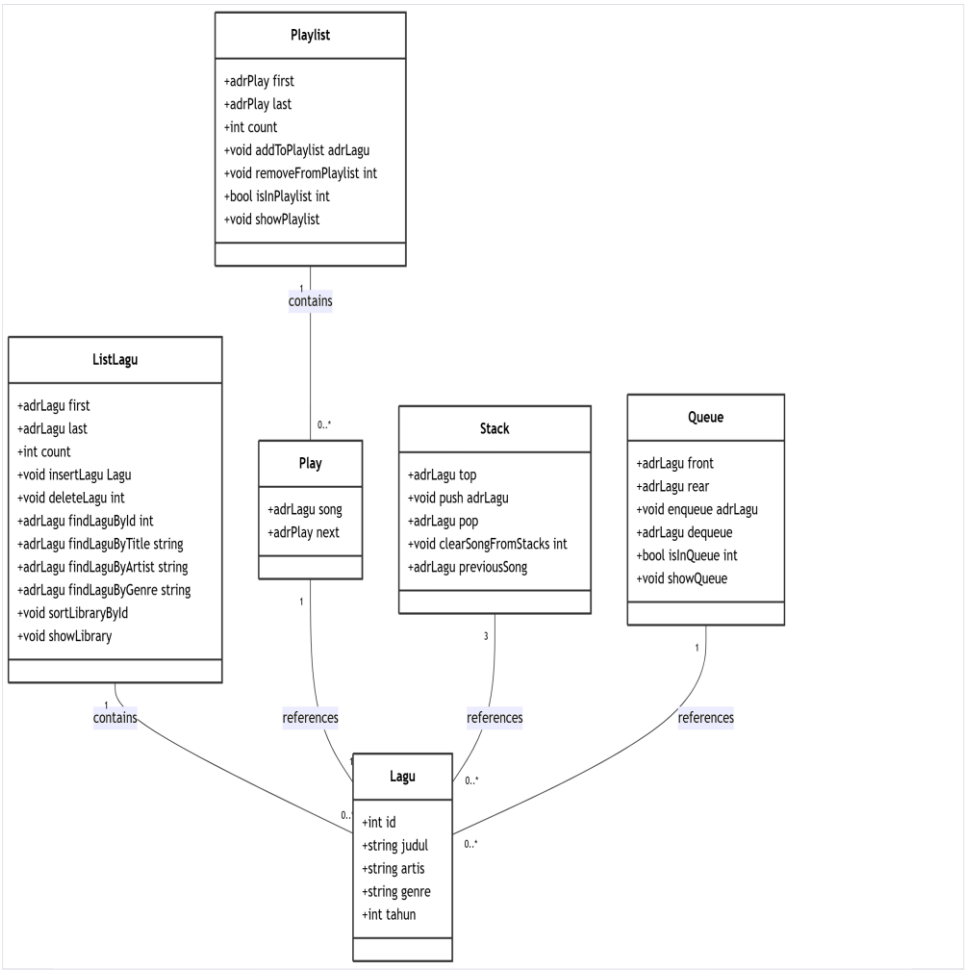
30. Function nextSimilar(L: ListLagu, current: adrLagu) return adrLagu. IS: Mencari lagu mirip berdasarkan artis atau genre, FS: Lagu mirip ditemukan atau NILL.
31. Function previousSong(in/out S: Stack) return adrLagu. IS: Kembali ke lagu sebelumnya, FS: Lagu sebelumnya dikembalikan.

PROGRAM FLOWCHART

User Journey



Class Diagram



LAPORAN TAHAP 3: TUGAS BESAR STRUKTUR DATA

KENDALA DAN SOLUSI

Kendala	Solusi
Kendala playlist harus mengikuti update library jika lagu diubah pada library, playlist tidak boleh menyimpan kopi data.	Solusinya dengan playlist menyimpan pointer (adrLagu) ke node DLL maka playlist langsung berubah otomatis tanpa perlu diperbarui.
Kendala penghapusan lagu mengakibatkan pointer menggantung jika Admin menghapus lagu dari DLL, playlist yang menyimpan pointer tersebut menjadi invalid.	Solusinya saat penghapusan lagu dilakukan, playlist dicari node yang menunjuk ke lagu tersebut node playlist ikut dihapus menghindari dangling pointer.

LAMPIRAN

Link GITHUB