Completely-Fun Version 2

Subject: **Data Structures**

Doctor: **Ahmed Seif**

Project: **Music Playlist Generator** with **ChatBot**

Work On

Name: Noran Ahmed **ID:** 42410001

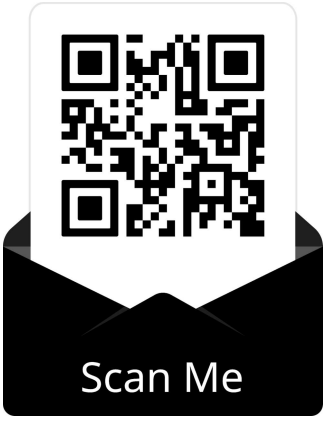
Name: Yousef Mohamed **ID:** 42410515

Name: Kerolos Nady **ID:** 42410542

Name: Shrouk Khalaf **ID:** 42410575

Name: Philopateer Waheed **ID:** 42410086

Name: Sohila Mahmoud **ID:** 42410561



Class Conclusion: Song

The Song struct encapsulates track metadata (title, artist, genre, duration, YouTube link). It guarantees a non-negative duration, provides a MM:SS formatter, and a compact display method. It participates in the playlist as a doubly-linked element via next/prev pointers.

- Fields: title, artist, genre, duration, youtubelink; links to neighbors via next/prev.
- Validation: duration coerced to ≥ 0 in the constructor.
- Utility: getformattedDuration() formats seconds into MM:SS; display() prints a readable card.

Class Conclusion: PlaylistManager

PlaylistManager manages a doubly-linked list of Song nodes (head/tail) with add/remove/display/save operations. It validates YouTube links (must contain youtube.com or youtu.be), logs actions with timestamps, and exports to CSV.

- Core operations: Addsong(), removeLastSong(), displayAll(), size(), getBack().
- Validation: isValidyoutubelink(url) ensures popular YouTube domains are present before accepting a song.
- Persistence: saveToExcel() writes playlist rows and an Activity Log section to name.csv.
- Activity tracking: DoublyLinkedList stores timestamped actions via logActivity().

Template Conclusion: DoublyLinkedList

A minimal, generic doubly-linked list used for the activity log. It supports push_back, basic iteration via begin(), emptiness and size queries, and properly releases nodes in the destructor.

- Members: head, tail, count; Node holds data with next/prev.
- Operations: push_back(value), begin(), empty(), size().

Class Conclusions: ResponseSystem & ActionBot

ResponseSystem provides keyword-based replies; ActionBot extends it with actionable commands that trigger playlist and data-structure operations. It handles user intents like displaying the playlist, adding/removing songs, undoing, queue playback, and searching.

- Rule set: greetings (hello/hi/hey), status, help, bye; plus command keywords.
- Commands: display playlist, play next, add song, remove last, undo remove, search song.
- Integration: calls PlaylistManager, queue, stack, and BST utilities to execute actions.

Class Conclusion: ChatBot

A simple conversational loop that delegates message interpretation to a ResponseSystem. It remains active until the user types ‘bye’, printing any non-empty responses returned by the

underlying system.

- Flow: prints prompt, reads input, checks for exit token, delegates to getResponse().
- Behavior: silent for command-handling paths where output is already produced by invoked operations.

Conclusions: Supporting Data Structures

- Undo Stack: Captures the last removed song for one-step restoration via pushUndo()/popUndo().
- Play Queue: Buffers upcoming tracks; playNext() dequeues and prints the now-playing title.
- Binary Search Tree: Insert and search by song title (insertBST/searchBST). Built as songs are added to accelerate lookups.

Conclusion: Program Flow & Menu

The application exposes a menu-driven interface for playlist management and a conversational chatbot. The main loop processes user choices, updates the BST index on song addition, supports undo, queue playback, and saves the playlist (CSV).

- Menu: 1 Add Song · 2 Remove Last · 3 Display · 4 Play Next · 5 Search · 6 Undo · 7 Chatbot · 8 Save (CSV) · 0 Exit.
- Global objects: myPlaylist (PlaylistManager), bstRoot (BST index).