AMOGUS
MUSIC PLAYER

OUTLINE

WHY DATA STRUCTURE?

PROJECT OVERVIEW

ARCHITECTURE (HOW WE MADE IT)

IMPLEMENTATION & FUNCTIONS

IMPOSTOR

THE ROLE OF AI

Q/A

WHY DATA STRUCTURE?

- EFFICIENCY: THEY OPTIMIZE HOW SOFTWARE USES MEMORY AND CPU.
- ORGANIZATION: THEY PROVIDE A LOGICAL WAY TO MANAGE COMPLEX DATA (LIKE A PLAYLIST OF THOUSANDS OF SONGS).
- SCALABILITY: GOOD STRUCTURES ENSURE THE APP RUNS FAST WHETHER IT HAS 10 SONGS OR 10,000.

EMERGENCY MEETING

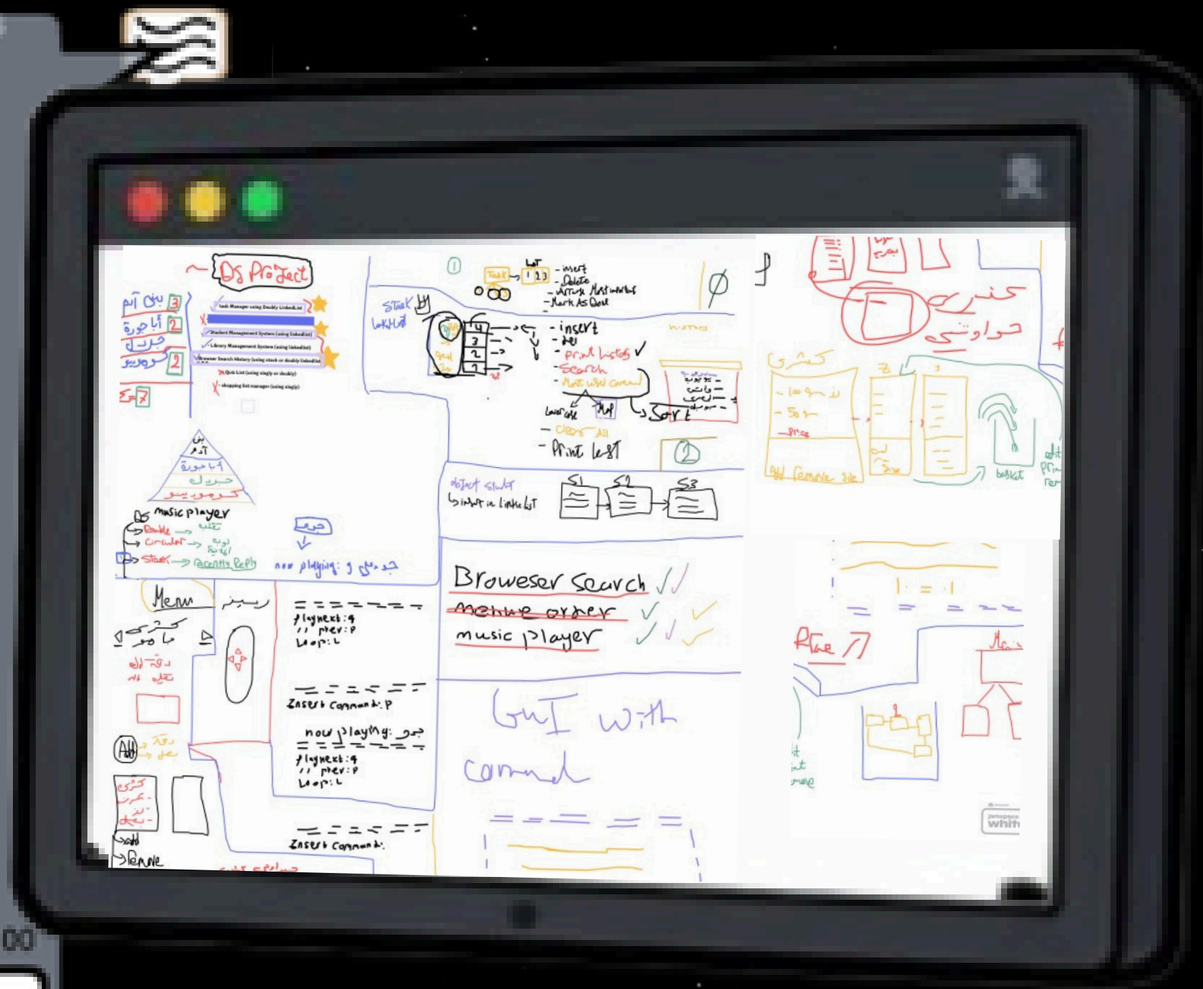
نعمل task managerوخلاص ؟

لا ياعم عايزين حاجة رايقة

تيجوا نطنش البروجكت وخلاص :)

حد يطلع الراجل دا عشان أنا علي اخري

ششش هو musicplayer



نستني رأي الزعيم

PROJECT OVERVIEW

- WE BUILT A MUSIC PLAYER THAT OPERATES ENTIRELY IN THE TERMINAL.
- IT SIMULATES REAL-WORLD APPLICATIONS LIKE SPOTIFY OR APPLE MUSIC BUT STRIPS AWAY THE HEAVY GRAPHICS TO FOCUS ON THE LOGIC BEHIND THE SCENES.
- GOAL: TO IMPLEMENT CORE DATA STRUCTURES IN A PRACTICAL, INTERACTIVE WAY.

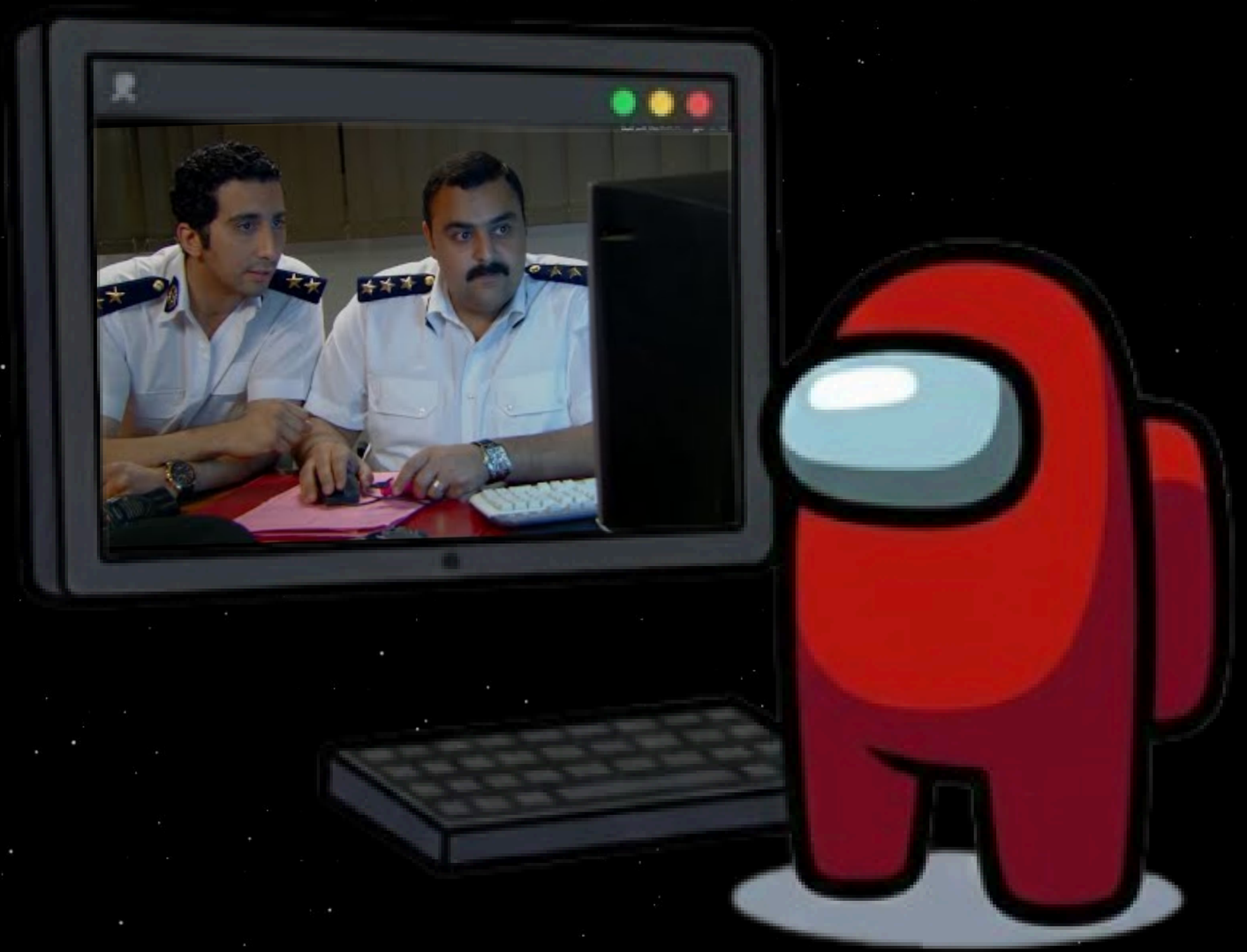
ARCHITECTURE (HOW WE MADE IT)

DOUBLY LINKED LIST (THE PLAYLIST)

- Used to store the songs.
- Why? Unlike arrays, it allows us to add or remove songs dynamically without shifting memory.
- Why Doubly? It allows us to navigate Forward (Next Song) and Backward (Previous Song) easily.

STACK (THE HISTORY)

- Used to track the "Recently Played" songs.
- Why? It follows the LIFO (Last In, First Out) principle. When you hit the "Back" button, it pops the last song played.



IMPLEMENTATION & FUNCTIONS

PLAYLIST MANAGEMENT

- addSong(Node* head, Song s): Adds a new track to the end of the list.
- isLooping(): This function is crucial for controlling the continuous playback behavior of the music player.
- searchSong(string name): Traverses the list to find a specific track.

IMPLEMENTATION & FUNCTIONS

PLAYBACK CONTROLS

- **playNext()**: Moves the current pointer to **current->next.**
- **playPrev()**: Moves the current pointer to **current->prev.**

IMPLEMENTATION & FUNCTIONS

STACK IMPLEMENTATION

- **push(Song s)** : Adds a song to the history.

Logic:

1. Create a new Node with the song data.
2. Link this new node to the current top of the stack (newNode->next = top).
3. Update the top pointer to be this new node.

- **pop()**: Goes back to the previous song.

Logic:

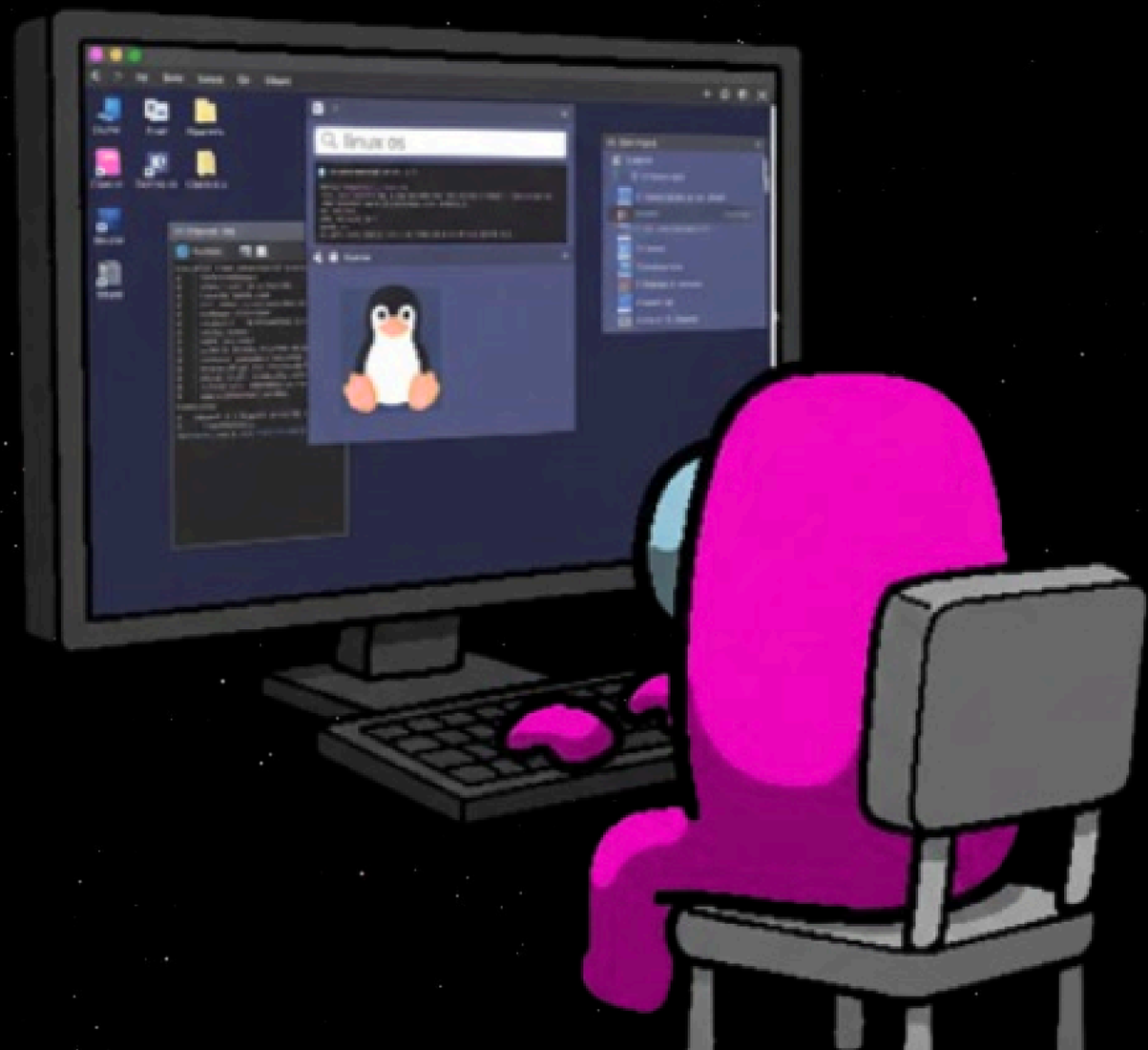
1. Safety Check: First, check isEmpty() to ensure we don't crash.
2. Store Data: Temporarily save the data of the current top node.
3. Move Pointer: Move the top pointer down one level (top = top->next).
4. Cleanup: Delete the old top node to free memory.

- **peek()**: Checks what the previous song was without removing it.
- **Logic:** Simply return top->data.

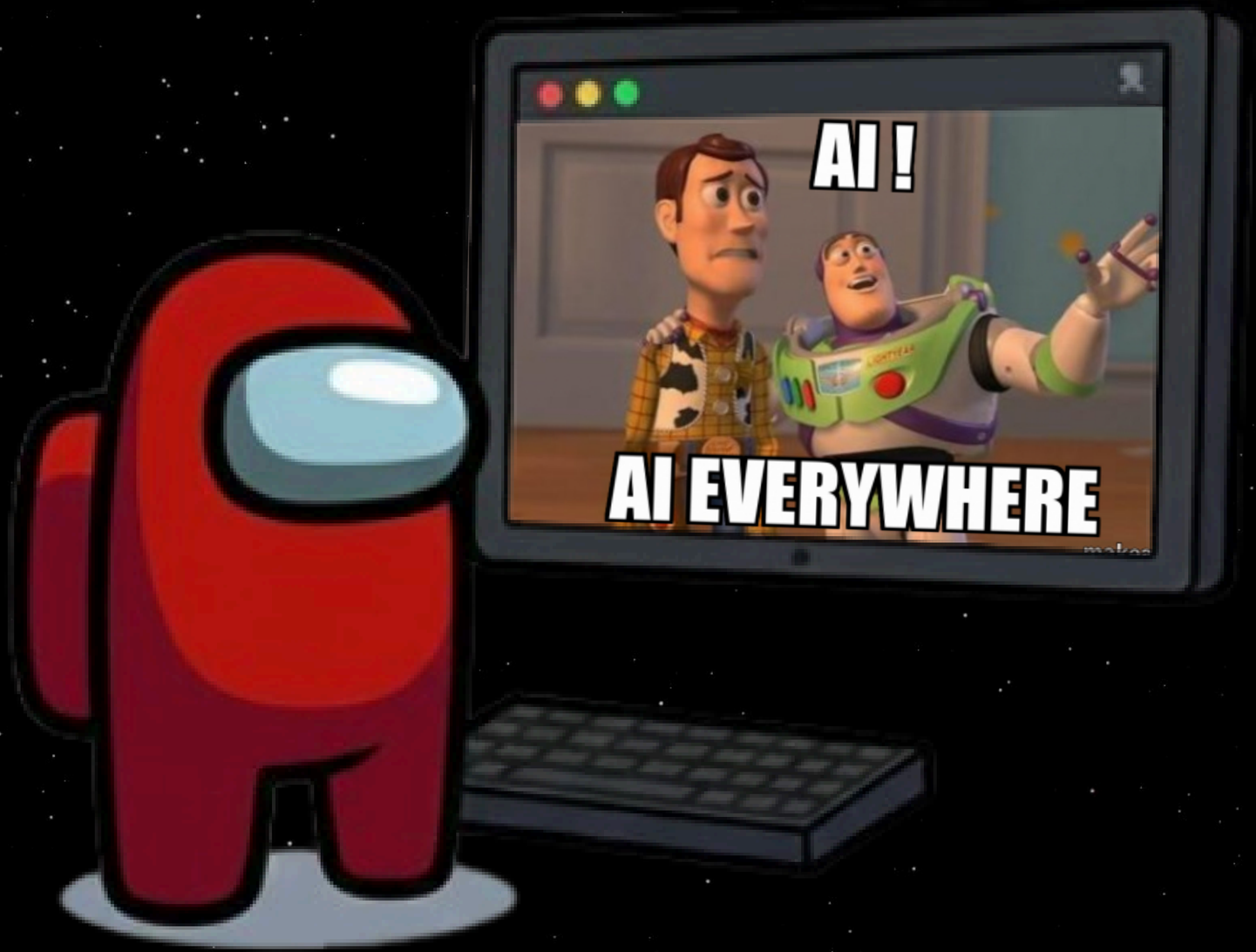
IMPOSTOR

THE TERMINAL GUI

- Instead of simple text input/output, we created an interactive dashboard.
- Features: Displays "Now Playing," the Queue, and simple menu options.
- Method: We used system commands to clear the screen and re-render the list state every time an action occurs, creating the illusion of a static interface.



THE ROLE OF AI



Victory

THANKS

Q/A