Bytexl's guided project

Final Project report

Name of the educator	**Shubham D**
Project title	**C Music Player Demo Frame**
Tools / platforms used	**Code::block**

About the Project:

The ***C Music Player Demo Frame*** project is an educational tool developed to help users understand and practice user-define Data Structures and Linked List and their algorithms in an interactive way. This tool allows users to create a visual representation of a Linked List nodes and structures, then observe the execution of algorithms like create, insert, update, delete in real-time. This visual approach enhances comprehension by demonstrating how algorithms traverse nodes and memory address, calculate paths, and explore relationships within the pointers and memory area. Primarily, this project is targeted at students and individuals looking to improve their understanding of data structures and algorithms, making it a strong addition for interview preparation and practical learning. The user-friendly interface ensures that even those with a basic understanding of user-defined data structures can easily navigate and experiment with these algorithms, reinforcing their learning experience.

System Requirements:

- **Software Requirements:**
  - Code::block(IDE)
- **Hardware Requirements:**
  - Minimum 4GB RAM
  - Processor: Intel i3 or equivalent (dual-core or higher recommended)
  - Storage: At least 500MB of free disk space
  - Display: Standard monitor (1920x1080 recommended for visual clarity)

Functional Requirements:

- Ability to create and add new song in the playlist and change the song to next and previous and also delete the playlist interactively to form a custom playlist.
- Support for multiple data structure with real-time visualization.
- Display algorithm-specific data (like traversing/display and next/prev).
- Clear and customizable visualization to highlight the algorithm's progress.

User Interface Requirements:

- Use command prompt as an UI where users can easily create, add, display, next, previous, delete nodes/songs.
- Clear labeling of nodes and structures with customizable attributes (like Title, Album, Year).
- Real-time feedback on each algorithm's process, such as displaying accurate update to the user.

Inputs and Outputs:

- **Inputs:**
  - User-defined data structures, dynamic data structure (Linked List).
  - User selection visualize the output.
- **Outputs:**
  - Music Player Demo Frame, showing the Main menu appearance and real-time processing of data structures.
  - Double Linked List and structures are used together.
  - Pre-defined libraries, and function are used specifically to gain the accurate output and result the actual instruction if invalid input provided.

List of Subsystems:

1. **Double Linked List:** Handles the creation, insertion, traversing, deletion of new songs.
2. **User-define Data Structures:** Structure manages to form a structure and store the record and the details of the inputted data in particular manner.
3. **Visualization Subsystem:** Displays the Main menu, Song directory, Song playlist, current song details, and the next and previous song execution.
4. **User Interface Subsystem:** Provides the frontend through command prompt by which users interact with it easily.

Other Applications Relevant to Your Project:

This project can be used in educational contexts to teach data structures and algorithms interactively. It's also valuable for interview preparation, as understanding **user-defined a**nd **dynamic** data structures is critical in tech roles. The project could be adapted for different flavors, such as adding more member in the structure like creating complex data structures for specialized applications like game development or visual Effects.

Designing of Test Cases:

1. **Node/Song Creation Test:** Ensure that nodes/songs can be created successfully without errors before adding into playlist.
2. **Node/Songs Addition Test:** Ensure that nodes/songs can be added successfully in the playlist without errors and the node/song is available in the collection.
3. **Next/Previous Song/NodeText:** All the songs added successfully in the playlist and they are organized in the manner so they can easily provide the next or previous song trace.
4. **Deletion of Song/Node Test:** Verify the Song/Node and then remove/delete it from the playlist structure.
5. **User Interface Test:** Check the appearance of the Instruction like Menu and their other option, submenu and make sure they are readable and visible.

Future Work:

Future improvements could include adding more algorithms and libraries to develop it for Visual interface dynamic insertion of data though, Like SDL2 Library.

References:

1. General Reference from the You tube and the website
2. Basic information is already included in the academics.

Reflection on the Project Creation:

During this project, I faced technical challenges related to inbuild libraries and their uses. Understanding how to effectively I can use and manage real-time use as per required. Careful integration of Structure and Doubly Linked List data structure to organize the data and the collection of songs. My academic background, especially in data structures, helped me design efficient ways to handle algorithm traversal within large data structures and perform many more operations. This project also emphasized the importance of implementing and testing of data structures in real-time, as the interactive nature of the project required extensive validation of each process.

The project gave me a unique opportunity to deepen my practical knowledge of Data Structure and Algorithms, particularly in how they function dynamically and interactively. While my existing knowledge was beneficial, familiarity with event-driven programming and advanced visualization libraries would have further streamlined the development process.