Web-Based Music Player Project Report

1. Introduction:

This project involved the development of a web-based music player application using React.js & Tailwind CSS. The goal was to create a user-friendly and interactive music player capable of playing audio files, managing playlist, and providing standard music player controls.

This report details the project's requirements, implementation, features, challenges, and future improvements.

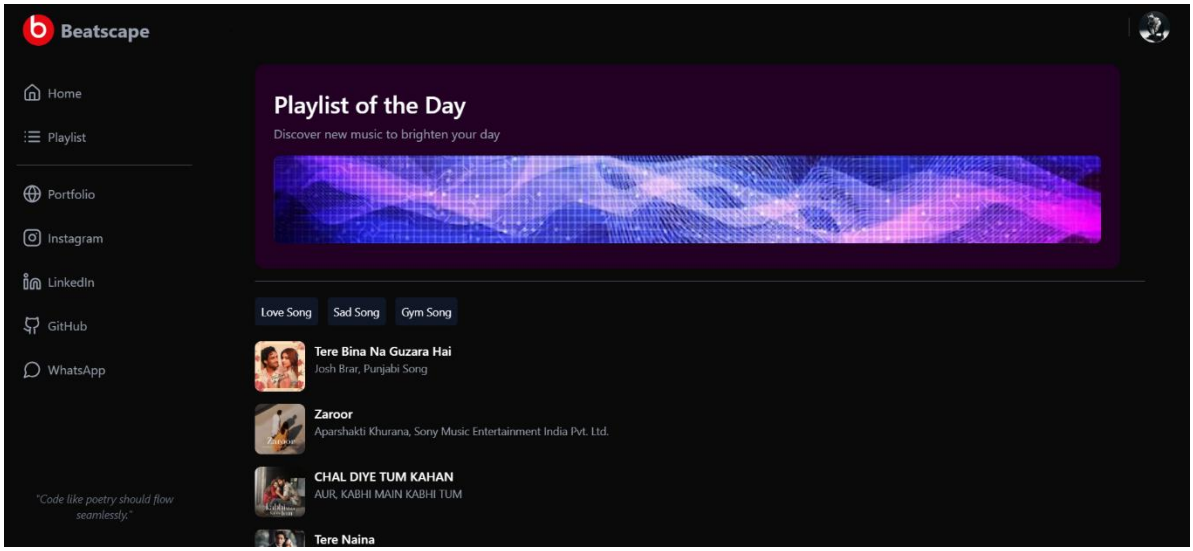
2. Problem Statement/Overview:

The primary objective was to build a functional and aesthetically pleasing music player within a web browser environment. This involved handling audio playback, designing an intuitive user interface, and managing user interactions. A key challenge was creating a smooth user experience while working within the limitations of browser-based audio playback.

3. Project Requirements:

The project had the following core requirements:

- **User Interface:** A clean and intuitive UI with controls for play/pause, volume adjustment, and a progress bar.
- **Audio Playback:** The ability to play audio files using HTML5's <audio> element.
- **Playlist Management (Stretch Goal):** Implementing a system to create and manage playlists.

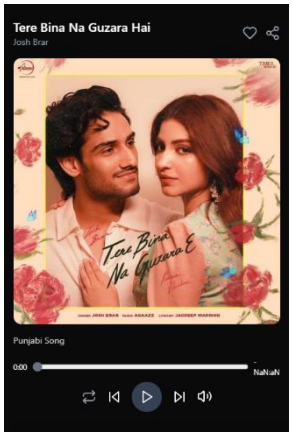


4. Features:

The implemented music player includes the following features:

- **Play/Pause Control:** A button toggles between play and pause states, controlling the audio playback.

  How to use: Click the button to start or stop the music.



- **Volume Adjustment:** A slider allows the user to adjust the volume of the audio.

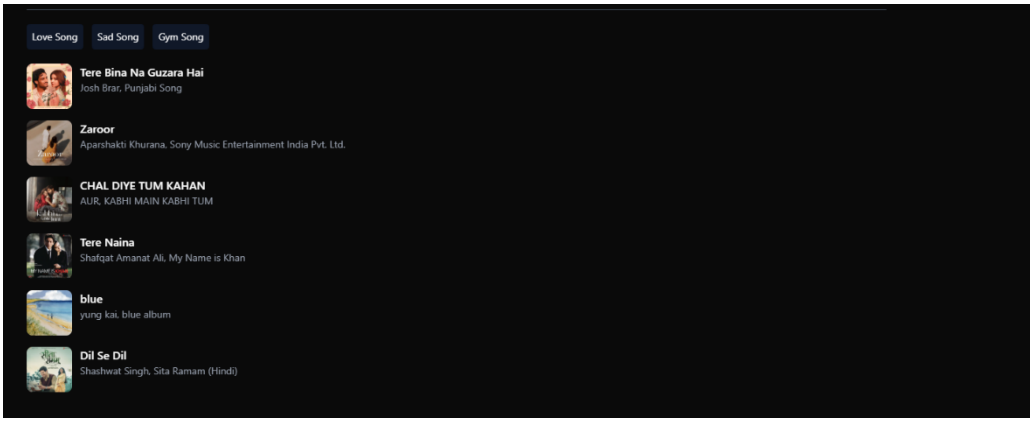
  How to use: Drag the slider to the left to decrease the volume, and to the right to increase it.

- **Progress Bar:** A visual representation of the current playback position within the audio track. The progress bar is also interactive, allowing users to seek to different parts of the song.

  How to use: Click or drag the thumb on the progress bar to jump to a specific time in the song.

- **Playlist Display (Partial Implementation):** A list of audio tracks is displayed, allowing the user to select which song to play

  How to use: Click on a song title in the list to start playing that track. Features like adding/removing songs from the playlist may or may not be complete.



5. Tech Stack:

- **Frontend:** React.js & Tailwind

  For structuring the web page and its elements, including the <audio> element and UI controls.

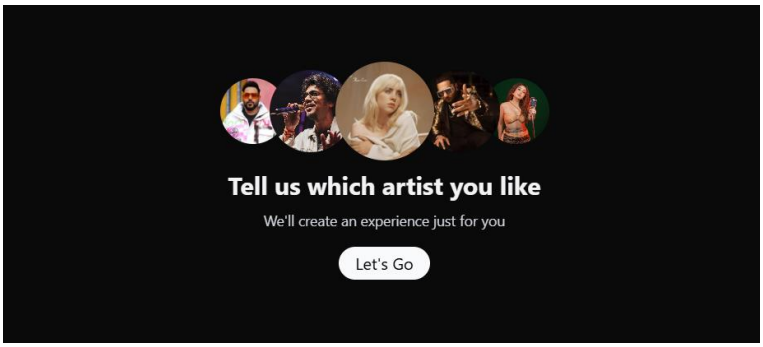
- **JavaScript:**

  For implementing the interactive functionality, such as controlling audio playback, updating the progress bar, and handling user events.

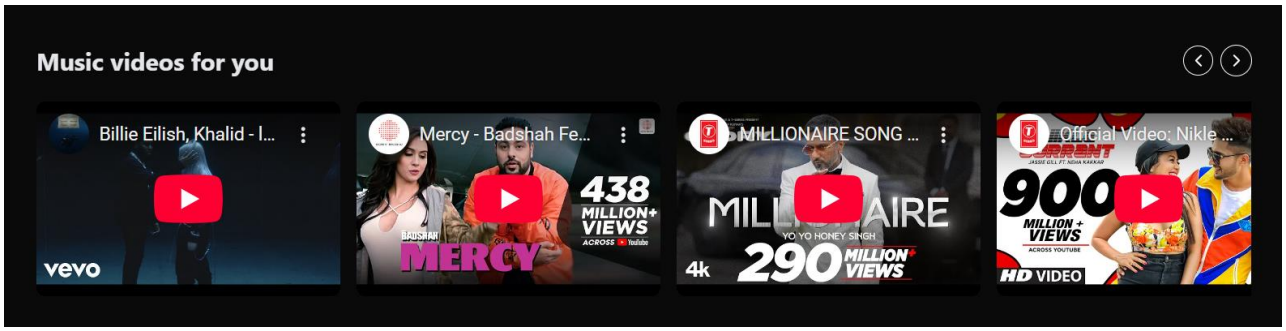
6. Evaluation Criteria:

- **Functionality:** The music player successfully plays audio files, manages basic playlists, and controls audio properly.
- **User Interface:** The UI is clean, intuitive, and responsive. Controls are easily accessible and understandable.
- **Code Quality:** The code is well-organized, commented, and follows best practices. Modularization is used where appropriate.
- **User Experience:** The music player provides a smooth and enjoyable user experience. Playback is seamless, and controls are responsive.
- **Creativity:** Beyond the core requirements, I added two features to enhance the user experience and make my music player stand out.

  First, I implemented a voting system, allowing users to vote for their favourite singers. This adds an element of community engagement and friendly competition.



Second, I integrated video, enabling users to not only listen to the audio but also experience the song's music video directly within the webpage. This provides a richer, more immersive experience and caters to users who enjoy the visual aspect of music. These additions demonstrate a proactive approach to enhancing the project beyond the basic requirements and showcase a deeper understanding of user engagement.



7. Challenges and Solutions:

- **Cross-Browser Compatibility:** Ensuring consistent behaviour of the <audio> element across different browsers was a challenge. This was addressed by using feature detection and providing fallback solutions where necessary.
- **Smooth Progress Bar Updates:** Continuously updating the progress bar smoothly while the audio plays required efficient JavaScript code. This was achieved by using requestAnimationFrame for optimized animation.
- **Playlist Management Complexity:** Implementing a full-featured playlist management system with features like adding, removing, and reordering songs proved to be more complex than initially anticipated.

8. How to Use (Guide):

1. Open https://beatscapehub.netlify.app/ in a web browser.
2. The music player interface will load.
3. Click on a song title in the playlist to select it.
4. Use the play/pause button to control playback.
5. Drag the volume slider to adjust the volume.
6. Click or drag the thumb on the progress bar to seek within the song.

9. Links:

- **GitHub Repository:** https://github.com/Ansari-Furkan-26/Unified-Mentor/
- **Project Link:** https://beatscapehub.netlify.app/
- **Portfolio:** https://frontendgenie.netlify.app/
- **LinkedIn:** https://www.linkedin.com/in/furqanansari/

10. Conclusion:

This project provided valuable experience in building a web-based application using React.js, Tailwind CSS, and JavaScript. While some features, like full playlist management, the core functionality of playing audio files and providing standard controls was successfully implemented. The project highlighted the challenges of cross-browser compatibility and the importance of efficient JavaScript coding for smooth user interactions. Future improvements would focus on completing the playlist management system, adding more advanced features (e.g., visualizations, equalizer), and further refining the user interface and experience.