

# Front End Engineering-II

Project Report

Semester-IV (Batch-2022)

Music Player using HTML/CSS/ReactJS



**Supervised By:**

Dr. Baljit kaur

**Submitted By:**

Jatin Sihag 2210990456

Kajal 2210990469

Jatin Goel 2210990457

Kashish 2210990494

**Department of Computer Science and Engineering  
Chitkara University Institute of Engineering & Technology,  
Chitkara University, Punjab**

## Abstract :

This project focuses on the development of a modern music player application using React.js, a popular JavaScript library for building user interfaces. The music player aims to provide an intuitive and seamless user experience, incorporating essential features such as playback control, playlist management, and real-time audio visualization. The application leverages the component-based architecture of React to ensure modularity and reusability of code, enhancing maintainability and scalability.

Key functionalities include play, pause, skip, and shuffle, as well as volume control and track progress indication. The integration of a state management system like Redux ensures efficient handling of application state, improving performance and responsiveness.

Additionally, the project incorporates external APIs for fetching song metadata, album art, and lyrics, enriching the user's experience with dynamic content. Overall, this project demonstrates the potential of React.js in creating interactive and user-friendly web applications, while addressing common challenges in music player development such as state management, asynchronous data fetching, and media handling.

It addresses common challenges in music player development, such as state management, asynchronous data fetching, and media handling, while delivering a rich set of features that cater to modern user needs. This music player serves as a comprehensive example of how contemporary web technologies can be used to build sophisticated, high-performance applications.

## Table of Contents

S.No.	Title(s)	Page No.(s)
1	Introduction	3-5
2	Problem Definition /requirements	6-7
3	Proposed Design/ Methodology	8-10
4	Results	11-14
5	References	15

## **Introduction :**

In the digital age, music consumption has dramatically shifted from physical media to digital formats, leading to the proliferation of various music streaming and playback applications. Amidst this transformation, providing users with an intuitive, responsive, and feature-rich music player has become essential. This project, a music player application developed using React.js, aims to meet these demands by offering a modern solution that blends functionality with an excellent user experience.

React.js, a widely adopted JavaScript library developed by Facebook, is known for its efficiency in building user interfaces, especially for single-page applications where reactivity and performance are crucial. Its component-based architecture allows developers to create encapsulated components that manage their own state, which can then be composed to build complex UIs. This modularity not only enhances code reusability but also simplifies maintenance and scalability, making it an ideal choice for developing a music player application. Additionally, the music player supports offline functionality, allowing users to download and play songs without an internet connection, and social sharing features that enable users to share their favorite tracks and playlists via social media platforms.

This project not only demonstrates the capabilities of React.js in building high-performance web applications but also addresses the common challenges in developing a music player, such as managing asynchronous data fetching, handling media playback, and ensuring a responsive and intuitive user interface. Through this music player application, we explore the intersection of modern web technologies and user-centric design, delivering a robust solution for contemporary music consumption.

### **1.1 Background**

The evolution of music consumption has dramatically shifted in recent years, moving from traditional physical media to digital formats and streaming services. This shift has necessitated the development of sophisticated music player applications that can provide seamless, high-quality playback and an enriched user experience.

The rise of web technologies has further expanded the possibilities for developing such applications, with React.js emerging as a powerful tool for building dynamic and responsive user interfaces. React.js, developed by Facebook, offers a component-based architecture that enhances the development of scalable and maintainable web applications, making it an ideal choice for creating a modern music player.

## 1.2 Objectives

The primary objective of this project is to develop a comprehensive music player application using React.js that delivers a smooth and intuitive user experience. The application aims to:

**Provide Core Playback Features:** Implement basic playback functionalities such as play, pause, skip, and shuffle.

**Enhance User Interaction:** Include advanced features like volume control, track progress indication, repeat modes, and real-time audio visualization.

**Utilize Efficient State Management:** Employ Redux for effective state management to ensure smooth performance and responsiveness.

**Integrate Dynamic Content:** Fetch song metadata, album art, and lyrics using external APIs to enrich the user's experience.

**Ensure Accessibility and Responsiveness:** Design a user interface that is both simple and accessible, with responsive design principles to support various devices and screen sizes.

**Offer Customization and Offline Capabilities:** Provide options for users to customize their experience and support offline functionality for music playback without an internet connection.

**Enable Social Features:** Allow users to share their favorite tracks and playlists on social media platforms.

## 1.3 Significance

This project holds significant value in demonstrating the potential of React.js in building interactive and user-friendly web applications. By addressing common challenges in music player development, such as state management, asynchronous data fetching, and media handling, the project highlights best practices and innovative solutions in web development. The application not only serves as a practical tool for music consumption but also as a learning resource for developers interested in leveraging modern web technologies.

The significance of this music player project extends to various stakeholders:

**For Users:** It offers a high-quality, customizable music listening experience with advanced features and an engaging interface.

For Developers: It provides a comprehensive example of building a complex web application using React.js, showcasing techniques for managing state, handling asynchronous operations, and creating responsive UIs.

For the Tech Community: It contributes to the ongoing discussion and development of web-based media applications, highlighting the capabilities and advantages of using modern JavaScript libraries like React.

## **1.4 Conclusion**

The development of a modern music player application using React.js underscores the transformative potential of web technologies in creating responsive, feature-rich, and user-friendly interfaces. Throughout this project, we have demonstrated how React.js component-based architecture facilitates the creation of scalable and maintainable applications, essential for handling the complexities of media playback and user interaction.

Key functionalities such as core playback controls, advanced audio features, real-time visualization, and dynamic content integration have been successfully implemented. The use of Redux for state management has ensured that the application remains performant and responsive, effectively handling the various states and interactions typical in a music player.

This project not only provides a practical tool for music consumption but also serves as a valuable example for developers looking to harness the capabilities of React.js. It addresses common challenges in web application development, such as asynchronous data handling and efficient state management, offering insights and solutions that can be applied to a wide range of projects.

In summary, the music player application exemplifies how modern web technologies can be leveraged to create sophisticated, high-performance applications. It highlights the importance of user-centric design and the effective use of state management and external integrations. This project contributes to the broader field of web development by showcasing

# **Problem Definition and Requirements :**

## **1.1 Problem Statement**

In today's digital age, listening to music has become a daily activity for many people. However, existing music player apps often lack certain features or are not user-friendly enough. Therefore, the goal is to develop a new music player using React.js that overcomes these limitations and provides an enjoyable listening experience for users.

## **1.2 Software Requirements**

- \* React.js: This library helps create the main structure and interface of the music player.
- \* Redux: It manages the data flow and ensures smooth communication between different parts of the app.
- \* React Router: This helps users navigate through different sections of the music player seamlessly.
- \* Axios or Fetch API: These tools fetch data about songs, artists, and albums from external sources.
- \* Web Audio API: It handles audio playback, ensuring a smooth and high-quality listening experience.
- \* CSS/SASS: Styling tools to make the music player visually appealing and responsive.
- \* Testing Libraries: Tools like Jest and React Testing Library to ensure the app works correctly.

## **1.3 Development Tools:**

- \* Code Editor: A program like VS Code to write and edit the code.
- \* Version Control: Using Git to keep track of changes made to the codebase.
- \* Build Tools: Tools like Webpack and Babel to bundle and compile the code.
- \* Deployment Platforms: Services like Netlify or Vercel to deploy the music player online.

## **1.4 External Integrations:**

- Music Metadata API: Integration with services like Spotify API to fetch details about songs, artists, and albums
- Lyrics API: Integration with services providing song lyrics for users who want to sing along.

## **1.5 Hardware Requirements**

\* Computer: A decently powerful computer with enough memory and processing power to handle development tasks.

\* Internet Connection: Necessary for fetching data from external APIs and deploying the app online.

\* Testing Devices: Various devices like computers, smartphones, and tablets to test the music player's compatibility and responsiveness across different platforms.

## **1.6 Data Sets**

\* Music Metadata: Information about songs, artists, albums, and genres fetched from external music databases.

Album Art: High-quality images of album covers obtained from the same music databases.

Song Lyrics: Text data containing lyrics for songs fetched from online lyric databases.

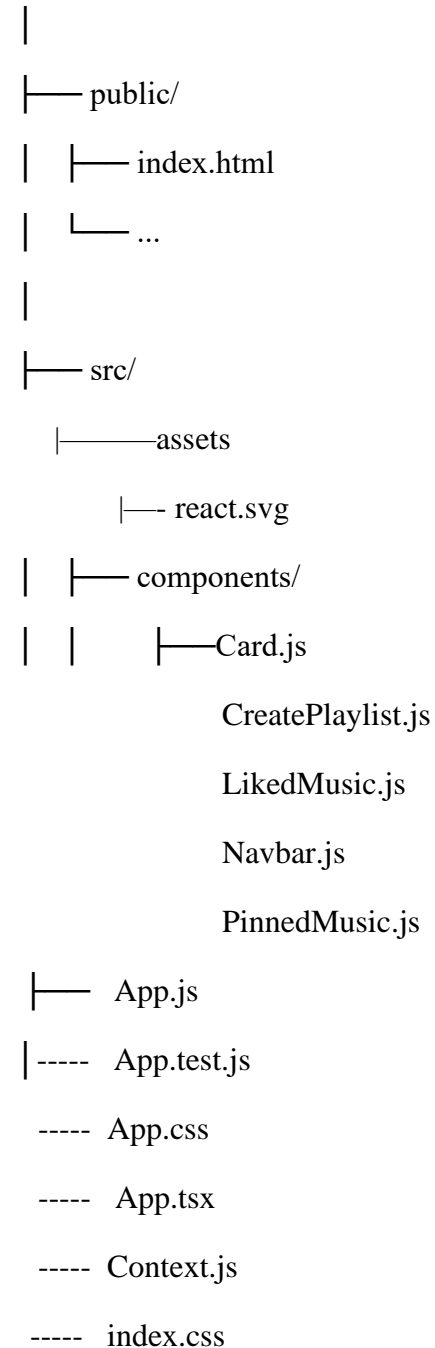
By fulfilling these requirements, we aim to develop a music player that not only meets users' expectations but also exceeds them by providing a smooth, feature-rich, and enjoyable music listening experience.

# Proposed Design / Methodology :

## 1.1 Schematic Diagram

File Structure

music-player-app/





```
    index.js
    initialize.js
    logo.svg
    main.tsx
    reportWebVitals.js
    setupTests.js
    vite-env.d.ts
|  └── package.json
|  └── README.md
|  └── tsconfig.json
    tsconfig.node.json
    package-lock.json
```

## 1.2 Methodology

- State Management with Redux:

Redux is used to manage the application's state, ensuring consistency and predictability across different components.

- Integration with External APIs:

Axios or Fetch API is employed to fetch data about songs, albums, and artists from external music metadata APIs like Spotify API.

- Audio Playback with Web Audio API:

Web Audio API is utilized for audio processing and playback control, ensuring smooth and high-quality audio playback.

- Component-Based Architecture:

The application is divided into reusable components like Player Controls, Playlist, and Audio Visualizer, making it easy to manage and update.

- Dynamic Routing with React Router:

React Router is used for dynamic navigation between different pages of the music player application, such as the home page, playlist page, and search page.

- Responsive Design:

CSS/SASS is used for styling the application, ensuring responsiveness across various devices and screen sizes.

- Testing with Jest and React Testing Library:

Unit and integration tests are written using Jest and React Testing Library to ensure the application functions as expected and remains bug-free.

### **1.3 Algorithms Used**

- Audio Visualization Algorithm:

Used to generate visual representations of audio data for the AudioVisualizer component.

Example: Fast Fourier Transform (FFT) for frequency analysis.

- Search Algorithm:

Used to search for songs, albums, or artists based on user input.

Example: Binary search for faster search results.

- Playlist Management Algorithm:

Used to manage playlists, including adding, removing, and rearranging songs.

Example: Linked list data structure for efficient playlist manipulation.

## Results:

- Screenshots

### 1.1 Home page



🎵 Jatin-music

Discover music in 30 seconds

## 1.2 Playlist page

Jatin-music

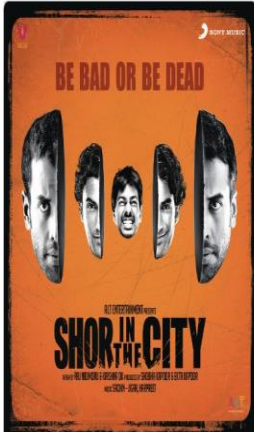
main--loquacious-cheesecake-36a7ae.netlify.app

Paused

Jatin-music

Saibo

Search




**Saibo**

Artist: Sachin-Jigar

Release date: 2011-03-28

0:00 / 0:29

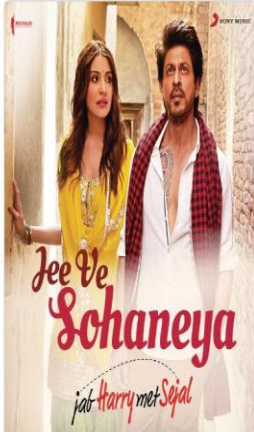


**Saibo - Lofi Flip**

Artist: VIBIE

Release date: 2021-06-18

0:00 / 0:29




**Saibo (From "Shor in the City")**

Artist: Various Artists

Release date: 2017-08-03

0:00 / 0:29



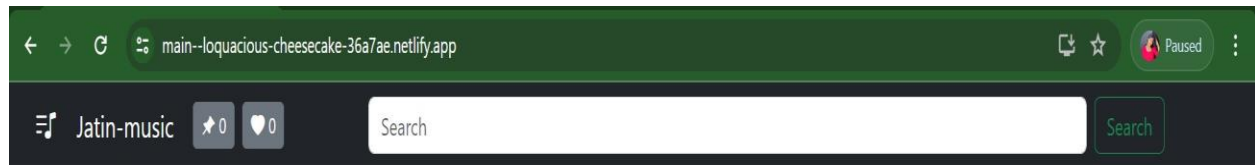
**Saibo - Instrumental**

Artist: Rishi Kumar

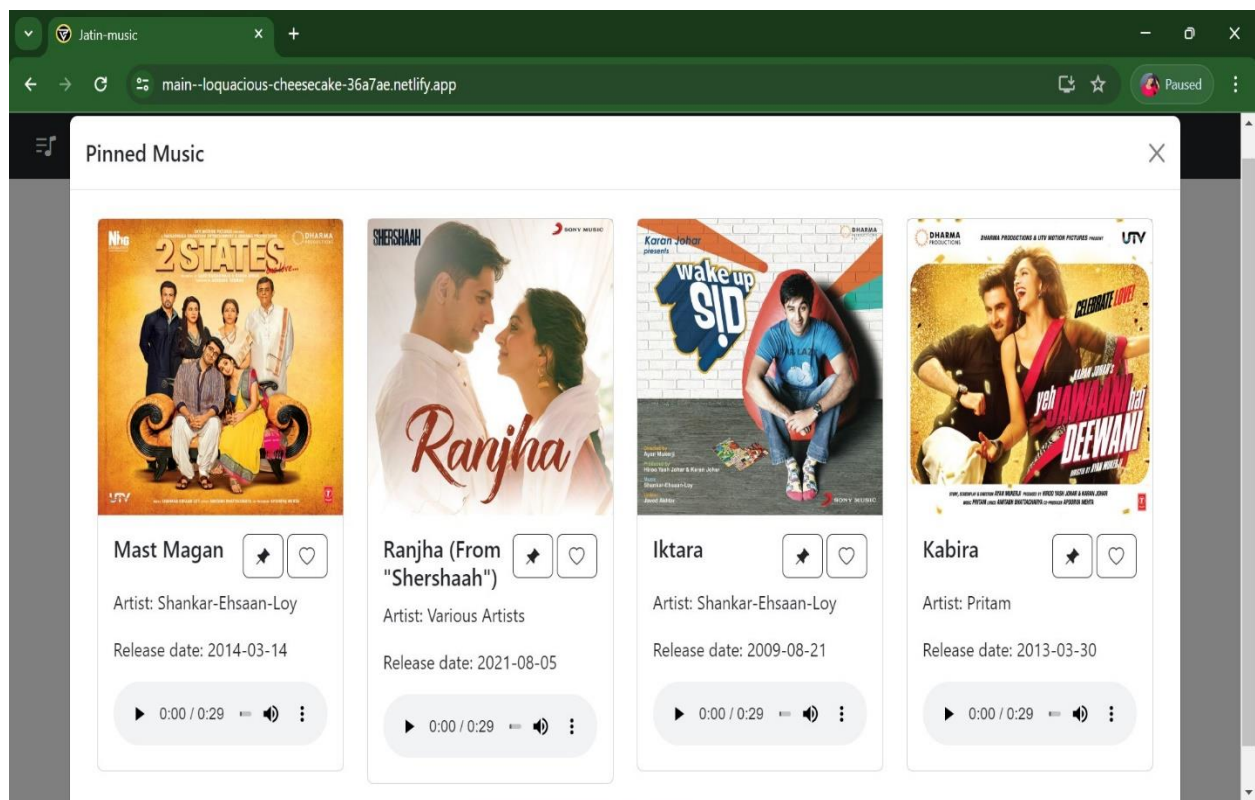
Release date: 2023-03-30

0:00 / 0:29

### 1.3 Search page



### 1.4 Pinned Music




## 1.5 Liked songs

Jatin-music

main--loquacious-cheesecake-36a7ae.netlify.app

Your Liked Music ❤️

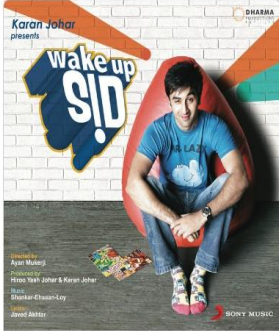


**Dead and Cold**

Artist: SadBoyProlific

Release date: 2018-11-10

0:00 / 0:29




**Iktara**

Artist: Shankar-Ehsaan-Loy

Release date: 2009-08-21

0:00 / 0:29

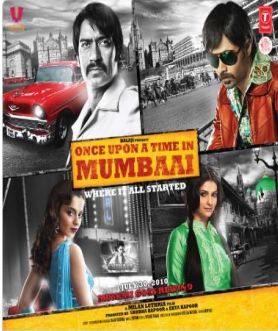


**COSAS QUE NO TE DIJE**

Artist: SAIKO

Release date: 2021-09-17

0:00 / 0:29



**Pee Loon**

Artist: Pritam

Release date: 2010-06-29

0:00 / 0:29

## References :

<https://github.com/topics/react-music-player>

<https://blog.logrocket.com/building-audio-player-react/>

<https://dev.to/aviyel/building-a-music-player-application-in-react-js-3ngd>