**Project Report**

**Name – Swaraj Santosh Pawar**

**Employee ID – 5852**

**Project On Music Player Application**

**Problem Statement**

This project aims to develop a basic music player application that addresses the need for a straightforward and efficient platform for users to manage and play their music. The challenges involve creating an intuitive user interface, ensuring high-quality audio playback, implementing robust library management features, and optimizing cross-platform compatibility. The successful completion of this project will result in a user-friendly music player application that caters to the diverse preferences of users seeking simplicity and functionality in their music experience.

**Understanding**

The essence of the project, centered around creating a basic music player application, involves addressing multifaceted challenges to deliver an enhanced user experience. The core objectives include designing an intuitive and visually appealing interface, ensuring top-notch audio playback across various file formats, and implementing a robust library management system. Cross-platform compatibility optimization is pivotal, encompassing both desktop and mobile environments Overall, the goal is to produce a user-friendly music player application that seamlessly blends simplicity with advanced functionalities for an enriched music consumption experience.

**Introduction**

In the dynamic landscape of digital entertainment, the demand for a user-friendly and versatile music player application has become increasingly pronounced. This project is dedicated to the development of a basic music player application, addressing the evolving needs of users seeking a seamless platform for managing and enjoying their music collections. A successful implementation involves overcoming challenges in designing an intuitive user interface, ensuring high-quality audio playback, and optimizing cross-platform compatibility. Key aspects include the incorporation of robust library management features, playlist functionalities, and search capabilities to enhance the overall user experience. This venture aims to strike a balance between simplicity and advanced functionalities, delivering a music player application that aligns with modern user expectations across various devices and operating systems.

**Why We Need This**

In the contemporary digital era, where music has become an integral part of daily life, the need for a purposeful and user-friendly music player application has never been more pronounced. Existing applications often present complexities that hinder a seamless and enjoyable user experience. This project addresses this gap by aiming to develop a basic music player that not only simplifies the organization and playback of music but also enhances the overall user experience. The envisioned application will cater to users who value simplicity without compromising essential functionalities. By providing an intuitive interface, reliable audio playback, and versatile features such as playlist management and search capabilities, this project seeks to meet the evolving expectations of music enthusiasts. In essence, the development of this music player application responds to the growing demand for a streamlined and efficient platform to manage and enjoy diverse music collections in today's digital landscape.

**ALGORITHM**

1. **HTML Structure:**

* Use HTML to define the structure of your music player. Include an <audio> element to handle audio playback, and buttons for controlling play/pause, stop, previous, and next functionalities.

1. **CSS Styling:**

* Apply CSS styling to make your music player visually appealing. You can use styles to control the layout, colours, and fonts. Ensure that the player is user-friendly.
* Implement responsive design using media queries to adapt the layout for different screen sizes.

1. **JavaScript Logic**

* **Use Array of Songs:** Create an array to store the filenames or URLs of your songs. This array will serve as your playlist.
* **Current Song Index:** Maintain a variable to keep track of the currently playing song's index. This allows you to easily navigate through the playlist.
* **Loading a Song:** Write a function to set the src attribute of the <audio> element to the current song, effectively loading and preparing it for playback.
* **Play/Pause Control:** Implement a function that toggles between playing and pausing the audio. This is usually achieved using the play() and pause() methods of the <audio> element.
* **Stop Functionality:** Create a function to stop the audio playback. This may involve setting the audio's currentTime back to 0 and pausing the audio.
* **Previous and Next Song:** Implement functions to move to the previous and next songs in the playlist. Update the current song index accordingly.

1. **Audio element :**

* The **`<audio>`** element is used for embedding sound content in your web page. Set its ` **src**` attribute dynamically to change the source of the audio file.
* Use the `**play()`** and ` **pause()`** methods to control playback.
* Listen for events such as `**ended`** to detect when a song finishes, allowing you to move to the next one.

1. **User interaction:**

* Buttons like play/pause, stop, previous, and next provide a way for users to interact with the music player.
* Use event listeners to capture button clicks and invoke corresponding functions.

1. **Updating UI:**

* **State Indicators:** Update the user interface to reflect the current state of the player (playing, paused, stopped). This could involve changing button styles or displaying relevant information.
* **Progress Bar:** For a more sophisticated player, consider adding a progress bar to show the current playback position within a song.

1. **JavaScript - Song Management:**

* Organize your songs in an array or any other data structure. Maintain an index to keep track of the current song.
* Implement logic to handle song transitions, like moving to the next or previous song.

1. **JavaScript – Responsive Design:**

* Use JavaScript to dynamically apply additional styles or adjust existing styles based on the device's screen size.
* Alternatively, rely on CSS media queries for responsive design.

1. **Testing:**

* **Test Cross-Browser Testing:** Test your music player on different web browsers to ensure compatibility. This is crucial for providing a seamless experience to users regardless of their browser choice.
* **Device Testing:** Test on various devices (desktops, laptops, tablets, and mobile phones) to verify responsiveness.

1. **Documentation:**

* Provide clear documentation for developers, explaining how to use and customize the code.
* Include information on the structure of the player data, API endpoint, and any other relevant details.

1. **Further Enhancement:**

* **Volume Control:** Add a slider or buttons to control the volume of the audio.
* **Playlist Display:** Display the list of songs in a playlist format for better user interaction.
* **Additional Controls:** Consider adding shuffle, repeat, and other controls based on the complexity you want to achieve.

By combining these elements, you can create a basic music player application that allows users to play, pause, stop, and navigate through a list of songs on your webpage.

**HTML CODE**

**Index.html**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Music Player</title>

    <link rel="icon" href="favicon.ico" type="image/x-icon" />

    <link rel="stylesheet" href="style.css" />

    <script

      src="https://kit.fontawesome.com/b127f1c0f9.js"

      crossorigin="anonymous"

    ></script>

  </head>

  <body>

    <div class="container">

      <div class="music-player">

        <nav>

          <div class="circle">

            <i class="fa-solid fa-angle-left" onclick="backwardSong()"></i>

          </div>

          <div class="circle">

            <i class="fa-solid fa-bars"></i>

          </div>

        </nav>

        <img

          src="media/jordon-conner-tIr-PWgSYB4-unsplash.jpg"

          class="song-img"

          alt="Song Image"

        />

        <h1>Pehle Bhi Main</h1>

        <p>Raj Shekhar and Vishal Mishra</p>

        <audio controls id="song">

          <source

            src="media/Pehle Bhi Main Animal 128 Kbps.mp3"

            type="audio/mpeg"

          />

        </audio>

        <input type="range" value="0" id="progress" aria-label="Progress Bar" />

        <div class="controls">

          <div>

            <i class="fa-solid fa-backward" onclick="backwardSong()"></i>

          </div>

          <div onclick="playPause()">

            <i class="fa-solid fa-play" id="ctrlicon"></i>

          </div>

          <div><i class="fa-solid fa-forward" onclick="forwardSong()"></i></div>

        </div>

      </div>

    </div>

    <script defer src="./script.js"></script>

  </body>

</html>

**CSS CODE**

\* {

  padding: 0;

  margin: 0;

  font-family: "popins", sans-serif;

  box-sizing: border-box;

}

.container {

  width: 100%;

  height: 100%;

  background-image: url("media/elice-moore-E--AUpYXbjM-unsplash.jpg");

  background-repeat: no-repeat;

  background-attachment: fixed;

  background-size: 100% 100%;

  background-position: center;

  display: flex;

  align-items: center;

  justify-content: center;

  flex-wrap: wrap;

}

.music-player {

  background: #ffe0e5;

  width: 400px;

  padding: 25px 35px;

  text-align: center;

}

nav {

  display: flex;

  justify-content: space-between;

  margin-bottom: 30px;

}

nav .circle {

  border-radius: 50%;

  width: 40px;

  height: 40px;

  line-height: 40px;

  background: #fff;

  color: #f53192;

  box-shadow: 0 5px 10px rgba(255, 26, 26, 0.22);

}

.song-img {

  width: 240px;

  height: 240px;

  border-radius: 50%;

  border: 8px solid #fff;

  box-shadow: 0 10px 60px rgba(255, 26, 26, 0.22);

}

.music-player h1 {

  font-size: 20px;

  font-weight: 400;

  color: #333;

  margin-top: 20px;

}

.music-player p {

  font-size: 14px;

  margin-bottom: 30px;

  color: #333;

}

#progress {

  appearance: none;

  width: 100%;

  height: 6px;

  background: #f53192;

  border-radius: 4px;

  cursor: pointer;

  margin: 40px 0;

}

#progress::-webkit-slider-thumb {

  appearance: none;

  background: #f53192;

  width: 30px;

  height: 30px;

  border-radius: 50px;

  border: 8px solid #fff;

  box-shadow: 0 5px 5px rgba(255, 26, 26, 0.22);

}

.controls {

  display: flex;

  justify-content: center;

  align-items: center;

}

.controls div {

  width: 60px;

  height: 60px;

  margin: 20px;

  background: #fff;

  display: inline-flex;

  align-items: center;

  justify-content: center;

  border-radius: 50%;

  color: #f53192;

  box-shadow: 0 10px 20px rgba(255, 26, 26, 0.22);

  cursor: pointer;

}

.controls div:nth-child(2) {

  transform: scale(1.5);

  background: #f53192;

  color: #fff;

}

**JAVASCRIPT CODE**

**Script.js**

let progress = document.getElementById("progress");

let song = document.getElementById("song");

let ctrlicon = document.getElementById("ctrlicon");

song.onloadedmetadata = function() {

    progress.max = song.duration;

    progress.value = song.currentTime;

}

function playPause() {

    if (!song.paused) {

        song.pause();

        ctrlicon.classList.remove("fa-pause");

        ctrlicon.classList.add("fa-play");

    } else {

        song.play();

        ctrlicon.classList.add("fa-pause");

        ctrlicon.classList.remove("fa-play");

    }

}

song.addEventListener('timeupdate', function() {

    progress.value = song.currentTime;

});

progress.oninput = function() {

    song.currentTime = progress.value;

    if (song.paused) {

        song.play();

        ctrlicon.classList.add("fa-pause");

        ctrlicon.classList.remove("fa-play");

    }

};

function forwardSong() {

    // Add logic for forwarding the song (e.g., increase currentTime)

    // You can adjust the forward duration based on your preference

    song.currentTime += 5; // Forward by 5 seconds (adjust as needed)

}

function backwardSong() {

    // Add logic for backwarding the song (e.g., decrease currentTime)

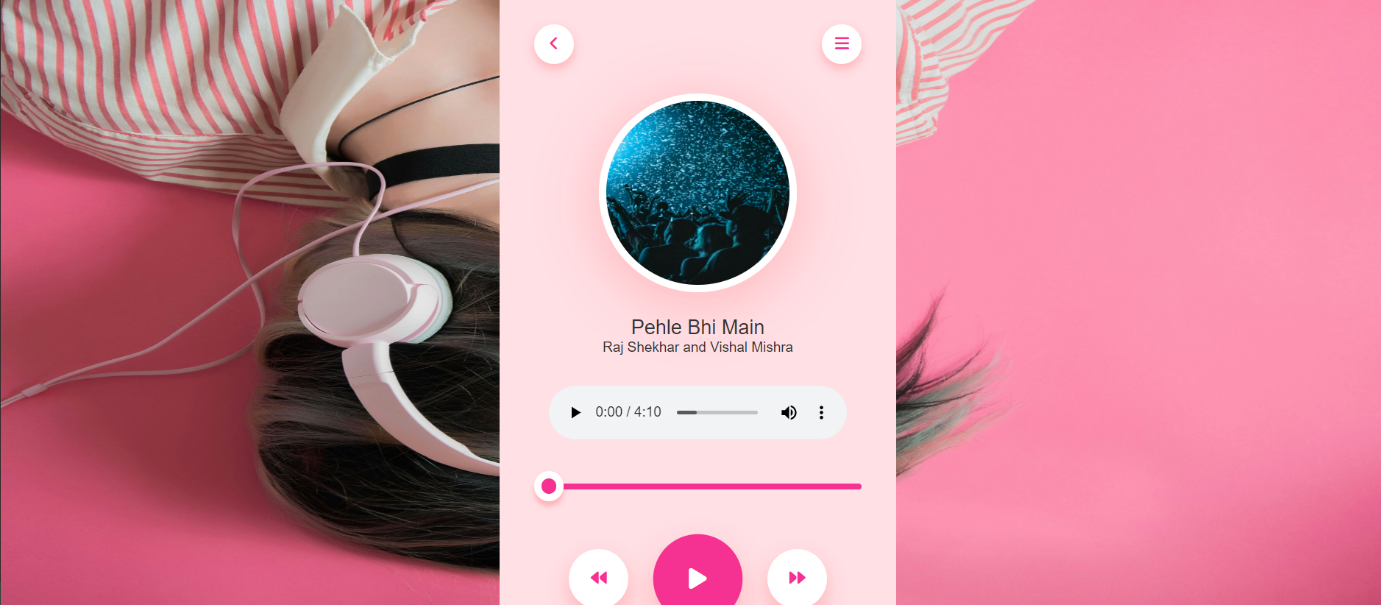
    // You can adjust the backward duration based on your preference

    song.currentTime -= 5; // Backward by 5 seconds (adjust as needed)

}

**OUTPUT IMAGES**

.



**A screenshot of a music player

Description automatically generated**

**A screenshot of a video player

Description automatically generated**

**REFERENCE**

1. MDN Web Docs - <https://developer.mozilla.org/>
2. YouTube - <https://www.youtube.com/>
3. 3.Google - https://www.google.com/