A

Project

Report On

“Music Player Using HTML CSS & JavaScript”

To Submitted -

Amrutvahini Polytechnic, Sangamner

Department: - Information Technology

In Partial Fulfilment of the Requirement for the Diploma in - Information Technology

Subject: Client Side Scripting 22519

Submitted By -

Roll no	Name of Student
47	Hande Vedant Machhindra
44	Gunjal Prathamesh Suresh
35	Ghodekar Suraj Gorakh

Under The Guidance of

Prof. Jondhale D.R

Amrutvahini Polytechnic, Sangamner

(Approved by AICTE, NEW DELHI Affiliated MSBTE)

2024-25

Amrutvahini Polytechnic Sangamner,

Department: -Information Technology

Certificate -

This is to that the Project Report Entitled,

"Music Player Using HTML CSS & JavaScript"

Is a Benefited Work Carrier Out By -

Roll no	**Name of Student**
47	Hande Vedant Machhindra
44	Gunjal Prathamesh Suresh
35	Ghodekar Suraj Gorakh

In Partial Fulfillment of the Requirement for the Diploma In

Information Technology During The Academic year 2024-25

Prof. Jondhale D.R
(Project Guide)

Prof. Chaudhari.N.K.
Hod (IT)

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organization. We would to kind to extend our sincere thanks to all of them.

First and foremost, we want to thanks **Prof. Chaudhari.N.K.** HOD IT) Amrutvahini polytechnic, Sangamner for giving opportunity to work on this project.

We are highly indebted to **Prof. Jondhale D.R** (Project guide) for his guidance and constant supervision as well as foe providing Necessary information regarding the project & also for his support in the Project.

We would like to express our gratitude towards our parents & members of Information Technology department for their kind cooperation and encouragement which help us in completion of this Our thanks and appreciations also go to our colleague in developing.

The project and people who have willingly helped us with their abilities.

Roll no	**Name of Student**
47	Hande Vedant Machhindra
44	Gunjal Prathamesh Suresh
35	Ghodekar Suraj Gorakh

INDEX

Micro-Project

“Music Player Using HTML CSS & JavaScript”

1. Rationale :

There are several reasons why it's important to know how to solve equations. The most important one is the problem solving strategies you learn by working through them. It helps train your brain to think. Solving equations is a way of thinking that you will benefit from unconsciously in other parts of your life.

2. Aims & Benefits:

Aims: **“Music Player Using HTML CSS & JavaScript”**

Benefits:

1. Learning Experience:Web Development Skills: Building a music player enhances your understanding of HTML, CSS, and JavaScript, key technologies in web development.

2. Customization Tailored Design: You can design the interface to match your personal style or the branding of a project.

3. User Experience: Interactivity: Users get a dynamic experience, with real-time interactions such as play/pause buttons, volume sliders, and track progress bars.

4. Performance:Lightweight: A music player built with HTML, CSS, and JavaScript can be lightweight, improving load times compared to using heavy frameworks.

5. Integration with Other Technologies: APIs: You can integrate with various music APIs (like Spotify or SoundCloud) for streaming music, enhancing the player’s functionality.

6. Portfolio Enhancement: Showcase Projects: A custom music player can serve as an impressive addition to your portfolio, showcasing your skills to potential employers or clients.

3. Course Outcomes:

CI 504.1	Use3 different program flow control structure for design interactive web pages.
CI 504.2	Exceute3 programs on Arrays and functions in Java script.
CI 504.3	Implement3 event based web forms and handling cookies using Java script.
CI 504.4	Apply3 regular expressions for validations to design interactive webpages.
CI 504.5	Implement3 Menus and navigations in web Pages.

4. Literature Review:

A music player created using **HTML, CSS, and JavaScript** brings together essential web technologies to deliver a modern, interactive, and highly responsive media experience. The **HTML5 `<audio>` element** forms the backbone of the player by providing native support for various audio formats, eliminating the need for external plugins, and ensuring cross-browser compatibility. **CSS** plays a pivotal role in enhancing the aesthetic appeal, allowing developers to design visually attractive interfaces that are both responsive and accessible across different devices and screen sizes. With **JavaScript**, dynamic functionalities such as play/pause controls, volume adjustments, skip buttons, and playlist management can be seamlessly integrated. Research emphasizes the importance of performance optimization techniques, like preloading and lazy loading, which improve load times and ensure smooth user interaction. Furthermore, responsive design principles ensure that the player works flawlessly on various devices, from smartphones to desktops. Existing music platforms like **Spotify Web Player** and **SoundCloud** provide valuable insights into best practices for web-based music players, showing how the combination of these technologies can lead to feature-rich, user-friendly music platforms.

5. Actual Methodology Followed:

A Music Player built using HTML, CSS, and JavaScript is a web-based application that allows users to play, pause, skip, and control audio files directly from a browser. The development process of such a music player involves several key stages, typical in software development, to ensure the application is functional, user-friendly, and efficient.

1. **Requirement Analysis :** The first step in creating a music player is to clearly identify the core functionalities needed in the application. This stage focuses on understanding user needs and defining the following requirements:

a. Core Playback Features: The music player should be able to play, pause, stop, skip to the next track, and go back to the previous track.
b. Volume Control: The user must be able to adjust the volume or mute the audio.
c. Track Progress Control: A progress bar should show the current position of the track and allow users to seek within the song.
d. Playlist Management: Users should be able to load multiple audio tracks and switch between them.
e. User Interface (UI): The interface must be intuitive, responsive, and accessible, ensuring ease of use on both desktop and mobile devices.

2. Design Phase: After gathering the requirements, the design phase focuses on the layout and structure of the music player. This involves creating a visual design and determining how the user will interact with the player.

a. HTML Structure: The basic structure of the player is created using HTML. The HTML elements typically include buttons for playback controls (play, pause, next, previous), volume control (volume slider or mute button), and a progress bar.
b. CSS for Styling: The design of the player is achieved using CSS to make the interface visually appealing and responsive. CSS is used to style buttons, control layout, and ensure that the player looks good on different screen sizes. Transitions and animations can also be added to enhance user experience, such as changing the appearance of buttons when they are clicked or hovered over.
c. JavaScript Logic: The core logic of the music player is implemented using JavaScript. JavaScript handles the audio control (play, pause, stop), updates the progress bar as the track plays, and manages playlist functionality. Additionally, event listeners are added to handle user interactions with the player controls.

3. Implementatio: The implementation phase involves coding the music player based on the designs and requirements from the previous stages.

a. HTML for Structure: The audio player is embedded in the webpage using HTML’s <audio> element. Control buttons such as play, pause, next, and previous are added using HTML buttons and other interactive elements like sliders for volume control.
b. CSS for Styling and Layout: CSS is used to style the interface, ensuring it is user-friendly and responsive. It includes defining the size, shape, and color of buttons, sliders, and the progress bar. CSS media queries are used to ensure the player is responsive and works well on both large screens (desktops) and small screens (mobile devices).
c. JavaScript for Interactivity: JavaScript is used to control the audio playback, manage the playlist, and update the UI in real-time. Key features include:
d. Play/Pause Functionality: JavaScript controls the playing and pausing of the audio element.

e. Volume Control: The volume is adjusted using the audio.volume property.
f. Progress Bar: JavaScript dynamically updates the progress bar as the track plays using the timeupdate event and allows users to seek within the track.

4. Testing: Testing ensures that the music player functions as expected and provides a smooth user experience.

a. Functional Testing: Each feature, including playback controls, volume adjustment, and track progress, is tested to ensure it works correctly. Testing should also cover how the player handles invalid input (e.g., when a song file is missing or corrupt).
b. Cross-Browser Testing: Since the music player is web-based, it needs to be tested across multiple browsers (Chrome, Firefox, Safari, etc.) to ensure compatibility.
c. Edge Case Testing: Testing should cover edge cases, such as fast clicking of buttons, handling of large files, or invalid file formats.

Music Player Project Overview:

1. HTML (Structure):The HTML file provides the foundation of the music player, organizing elements that allow interaction with the player. Key components include:

a. Track Information: Displays the current track's artist and image.
b. Playback Controls: Buttons for playing, pausing, skipping to the next or previous tracks.
c. Volume Control: A volume slider to adjust the sound output and a mute button.
d. Song Progress: A slider that shows the song's progress and allows users to jump to specific points in the track.
e. Autoplay Option: A button to toggle autoplay, which plays the next song automatically when the current track ends.

2. CSS (Styling):The CSS file handles the visual aesthetics of the music player, ensuring a modern and user-friendly design. Key points include:

a. Layout: The player is designed using flexbox to create a responsive layout, ensuring all elements are well-aligned and evenly spaced.
b. Styling Elements:
c. The music player's buttons (play, pause, next, previous) are styled using FontAwesome icons, giving them a polished look.
d. The background features a semi-transparent dark theme, which highlights the album art and controls.
e. Volume and Duration Sliders: Custom styles for the range sliders give them a sleek, modern appearance.
f. Responsiveness: Media queries are used to ensure the layout adjusts smoothly on mobile devices, making the player fully responsive. The image size, button layout, and text size all scale appropriately for smaller screens.

3. JavaScript (Functionality): The JavaScript file is responsible for adding interactivity and managing the core functionalities of the music player. Key features include:

a. Loading and Managing Tracks:
b. A playlist of songs is predefined in the code with file paths, artist names, and album art.
c. The load_track() function is used to load a new song, displaying the relevant track image and artist name, and updating the current track index.
d. Playback Controls:
e. The player can toggle between playing and pausing a track using the justplay() function, with the play button changing its icon dynamically.
f. The next_song() and previous_song() functions allow users to skip forward or backward through the playlist.
g. Volume and Mute:

h. Users can adjust the volume using a range slider, and the volume_change() function sets the current volume based on the slider’s value.
i. The player also includes a mute_sound() function to instantly mute the sound.
j. Song Progress:
k. A progress bar shows the current time of the song in relation to its total duration.
l. The range_slider() function continuously updates the progress bar as the song plays. Users can also drag the slider to move to specific parts of the song with the change_duration() function.
m. Autoplay Feature:
n. The autoplay_switch() function toggles the autoplay feature on and off. When autoplay is enabled, the next track will automatically play when the current one ends.
o. Dynamic Interface Updates:
p. The number of songs and the current song number are displayed dynamically, so users know where they are in the playlist. This is updated every time a new track is loaded.

Key Features and Functionalities:

- ✓ Track Loading: Dynamically loads songs from an array and updates the displayed artist, album art, and progress bar.
- ✓ Play/Pause Control: Users can toggle between playing and pausing the song with a single button.
- ✓ Next/Previous Track Navigation: Users can skip to the next or previous song in the playlist.
- ✓ Volume Control: Adjustable volume slider and a mute button provide fine control over the sound output.
- ✓ Progress Bar: Users can track the progress of a song and manually adjust the playback position.
- ✓ Autoplay Option: Automatically plays the next song when the current one finishes if enabled.
- ✓ Responsive Design: The player adapts to different screen sizes, ensuring a seamless experience across desktop and mobile devices.

Code :

HTML -

```html
<!DOCTYPE html>
<html>
  <head>
    <title>VPS Music</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" />
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>

  <body>
    <div class="main">
      <p id="logo"><i class="fa fa-music"></i>VPS Music</p>

      <!-- show_song_number -->
      <div class="show_song_no">
        <p id="present">1</p>
        <p>/</p>
        <p id="total"></p>
      </div>

      <!--- left part --->
      <div class="left">
        <!--- song img --->
        <img id="track_image" />
        <div class="volume">
          <p id="volume_show">90</p>
          <i
            class="fa fa-volume-up"
            aria-hidden="true"
            onclick="mute_sound()"
            id="volume_icon"></i>
          <input
            type="range"
            min="0"
            max="100"
            value="90"
            onchange="volume_change()"
            id="volume" />
        </div>
      </div>
      <!--- right part --->
      <div class="right">
        <!--- song title & artist name --->
        <div class="song_detail">
          <!-- <p id="title"></p> -->
          <p id="artist"></p>
        </div>



        <!--- middle part --->
        <div class="middle">
          <button onclick="previous_song()" id="pre">
            <i class="fa fa-step-backward" aria-hidden="true"></i>
          </button>
          <button onclick="justplay()" id="play">
            <i class="fa fa-play" aria-hidden="true"></i>
          </button>
          <button onclick="next_song()" id="next">
            <i class="fa fa-step-forward" aria-hidden="true"></i>
          </button>
        </div>

        <!--- song duration part --->
        <div class="duration">
          <input
            type="range"
            min="0"
            max="100"
            value="0"
            id="duration_slider"
            onchange="change_duration()" />
          <button id="auto" onclick="autoplay_switch()">
            Auto Play &nbsp;<i
              class="fa fa-circle-o-notch"
              aria-hidden="true"></i>
          </button>
        </div>
      </div>
    </div>

    <!--- Add javascript file --->
    <script src="script.js"></script></body></html>
```

CSS

```css
* {
  margin: 0;
  padding: 0;
  font-family: cursive;
}
body {
  min-height: 100vh;
  display: grid;
  place-items: center;
  background: rgba(0, 0, 0, 0.5);
}
```

```css
.main {
  /* margin-top: 15px; */
  position: relative;
  height: 80vh;
  width: 80%;
  display: flex;
  align-items: center;
  justify-content: center;
  background: #232427;

  border-radius: 8px;
  box-shadow: -3px -2px 151px 12px rgba(255, 1,
153, 1);
}
.main button {
  padding: 10px 12px;
  margin: 0 10px;
}
.main #logo {
  position: absolute;
  top: 10px;
  left: 30px;
  font-size: 25px;
  color: #ccc;
}
.main #logo i {
  margin-right: 15px;
}
/* left & right part */
.left {
  width: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
}

/* song image */
.left img {
  height: 300px;
  width: 80%;
  border-radius: 15px;
  object-fit: cover;
  box-shadow: inset 2px 2px 5px rgba(0, 0, 0, 0.5),
    inset -2px -2px 5px rgb(80, 195, 224), 5px 5px
15px rgba(0, 0, 0, 0.3),
    -5px -5px 15px rgba(255, 1, 153, 0.7);
  padding: 5px;
```

```css
  /* -3px -2px 151px 12px rgba(255, 1, 153, 0.6) */
}

/* both range slider part */
input[type="range"] {
  -webkit-appearance: none;
  width: 50%;
  outline: none;
  height: 10px;
  margin: 0 15px;
  overflow: hidden;
  border-radius: 25px;
}
input[type="range"]::-webkit-slider-thumb {

  -webkit-appearance: none;
  height: 10px;
  width: 10px;
  background: #ff02e1;
  cursor: pointer;
  box-shadow: -415px 0 0 400px #ff02e1;
}
.right input[type="range"] {
  width: 40%;
}

/* volume part */
.left .volume {
  margin-top: 25px;
  width: 80%;
  height: 30px;
  display: flex;
  align-items: center;
  justify-content: center;
  color: #ff02e1;
  /*  border: 1px solid #fff;*/
}
.volume input[type="range"] {
  flex: 1;
}
.left .volume p {
  font-weight: bold;
  font-size: 15px;
}
.left .volume i {
  cursor: pointer;
  padding: 8px 12px;
  background: rgba(245, 245, 245, 0.1);
  border-radius: 10px;
}
.left .volume i:hover {
  background: rgba(246, 0, 209, 0.1);
}
.volume #volume_show {
  padding: 8px 12px;
  margin: 0 5px 0 0;
  background: rgba(245, 245, 245, 0.1);
  border-radius: 10px;
}

/* right part */
.right {
  width: 50%;
  padding: 10px 0;
  display: flex;
  align-items: center;
  flex-direction: column;
}
.right .middle {

  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
}
.right .middle button {
  border: none;
  height: 70px;
  width: 70px;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: pointer;
  outline: none;
  transition: 0.5s;
  background: #232427;
  box-shadow: inset 2px 2px 5px rgba(0, 0, 255,
0.5),
    inset -2px -2px 5px rgb(0, 204, 255), 5px 5px
15px rgba(0, 0, 0, 0.3),
    -5px -5px 15px rgba(255, 255, 255, 0.3);
}
.song_detail {
  position: relative;
  width: 80%;
  overflow: hidden;
  margin-bottom: 6.5em;
  /*  border: 1px solid #fff;*/
}
/* .song_detail #title {
  text-transform: capitalize;
  color: #fff;
  font-size: 35px; */
/* } */
.song_detail #artist {
  text-transform: capitalize;
  color: #fff;
  font-size: 35px;
  margin-top: 5px;
  justify-content: center;
  /* align-items: center; */
}
.right .duration {
  margin-top: 3em;
  position: relative;
  display: flex;
  align-items: center;
  justify-content: center;
  width: 80%;
  /* border: 1px solid rgba(2, 240, 200, 0.724); */
}
.duration input[type="range"] {
  flex: 10;
  /* box-shadow: -3px -2px 151px 12px rgba(2, 57,
237, 0.673); */
}

.right #auto {
  font-size: 17px;

  text-align: center;
  cursor: pointer;
  border: none;
  padding: 5px 7px;
  color: #04e0fd;
  background: rgba(255, 255, 255, 0.2);
  outline: none;
  border-radius: 100px;
  box-shadow: inset 2px 2px 5px rgba(0, 0, 255,
0.5),
    inset -2px -2px 5px rgb(0, 204, 255), 5px 5px
15px rgba(0, 0, 0, 0.3),
    -5px -5px 15px rgba(255, 255, 255, 0.1);
```

```css
}
/* #play {
  background: #148f77;
} */
.right button:hover {
  background: #148f77;
}
.right i:before {
  color: #04e0fd;
  font-size: 23px;
}
.show_song_no {
  position: absolute;
  top: 10px;
  right: 10px;
  width: 30px;
  height: 20px;
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 8px 12px;
  color: #ebebeb;
  border-radius: 5px;
  background: rgba(255, 255, 255, 0.2);
  box-shadow: inset 2px 2px 5px rgba(0, 0, 0, 0.2),
    inset -2px -2px 5px rgba(255, 255, 255, 0.1),
    5px 5px 15px rgba(0, 0, 0, 0.3), -5px -5px 15px
rgba(255, 255, 255, 0.1);
}
.show_song_no p:nth-child(2) {
  margin: 0 5px;
}

/*responsive*/
@media (max-width: 700px) {
  .main {
    min-height: 100vh;
    width: 100%;
    flex-direction: column;
  }
  .right {

    margin-top: 50px;
    width: 60%;
  }
  .right .duration {
    width: 90%;
  }
  .left {
    margin-top: 5em;
    width: 60%;
  }
  .left img {
    min-width: 90%;
    height: 180px;
  }
  .main #logo {
    display: none;
  }
  .song_detail {
    position: absolute;
    top: 5px;
    left: 10px;
    width: 80%;
    height: 70px;
  }
  .song_detail #title {
    font-size: 1.8em;
  }
}

@media (max-width: 500px) {
  .main {
    min-height: 100vh;
    width: 100%;
    flex-direction: column;
  }
  .right {
    margin-top: 50px;
    width: 80%;
  }
  .left {
    margin-top: 5em;
    width: 80%;
  }
  .left img {
    min-width: 90%;
    height: 180px;
  }
  .main #logo {
    display: none;
  }
  .song_detail {
    position: absolute;
    top: 5px;
    left: 10px;

    width: 80%;
    height: 70px;
  }
  .song_detail #title {
    font-size: 1.5em;
  }
  .song_detail #artist {
    font-size: 0.8em;
  }
  .right .middle button {
    height: 62px;
    width: 62px;
  }
```

JAVASCRIPT -

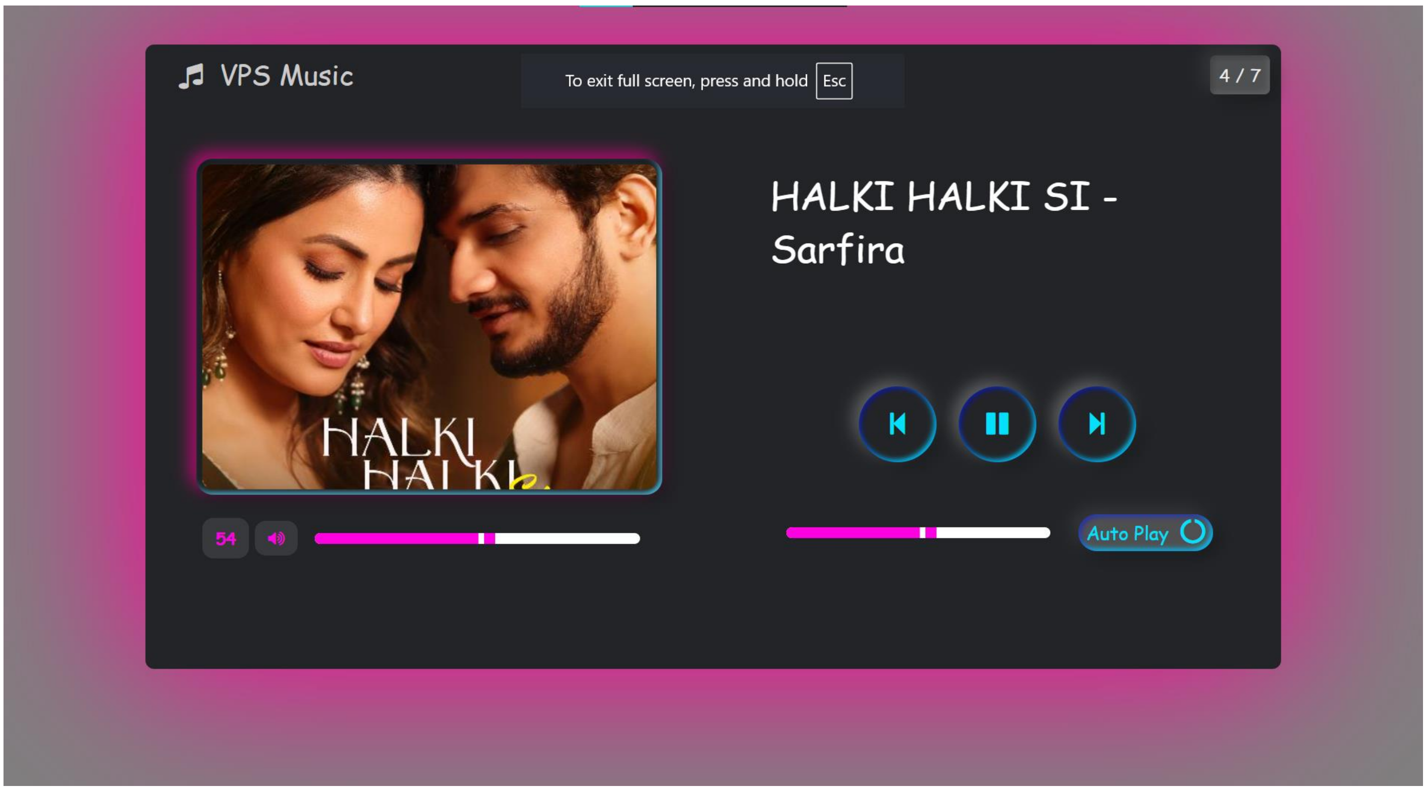
```
let previous = document.querySelector("#pre");
let play = document.querySelector("#play");
let next = document.querySelector("#next");
// let title = document.getElementById("#title");
let recent_volume = document.querySelector("#volume");
let volume_show = document.querySelector("#volume_show");
let slider = document.querySelector("#duration_slider");
let show_duration = document.querySelector("#show_duration");
let track_image = document.querySelector("#track_image");
let auto_play = document.querySelector("#auto");
let present = document.querySelector("#present");
let total = document.querySelector("#total");
let artist = document.querySelector("#artist");

let timer;
let autoplay = 0;

let index_no = 0;
let Playing_song = false;
```

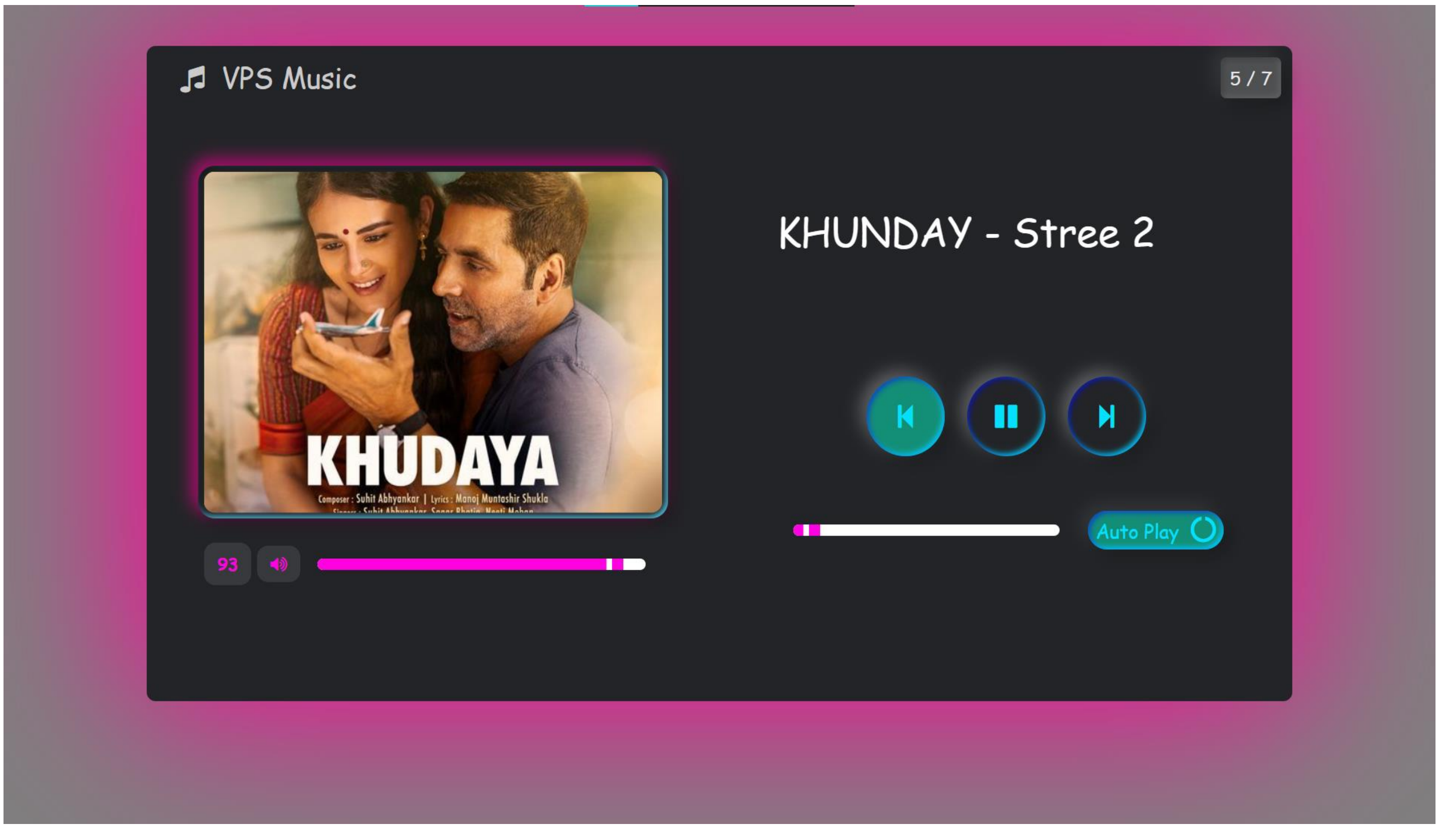
```
//create a audio Element
let track = document.createElement("audio");
//All songs list
const All_song = [

  {
    path: "song1.mp3",
    img: "img1.jpg",
    singer: "AAJ KI RAT - Mr . And Mrs. Mahi",
  },
  {
    path: "song2.mp3",
    img: "img2.jpg",
    singer: " AAYI NAI - Asees Kaur",
  },
  {


    path: "song3.mp3",
    img: "img3.jpg",
    singer: " AGAR HO TUM -  Chandu Champion",
  },
  {
    path: "song4.mp3",
    img: "img4.jpg",
    singer: " HALKI HALKI SI - Sarfira ",
  },
  {
    path: "song5.mp3",
    img: "img5.jpg",
    singer: "  KHUNDAY - Stree 2",
  },
  {
    path: "song6.mp3",
    img: "img6.jpg",
    singer: " MERE MEHBOOB MERE SANAM - Bad News",
  },
  {
    path: "song7.mp3",
    img: "img7.jpg",
    singer: " TU HI HYE CHAMPION - Bad News",
  },
];

// All functions

// function load the track
```

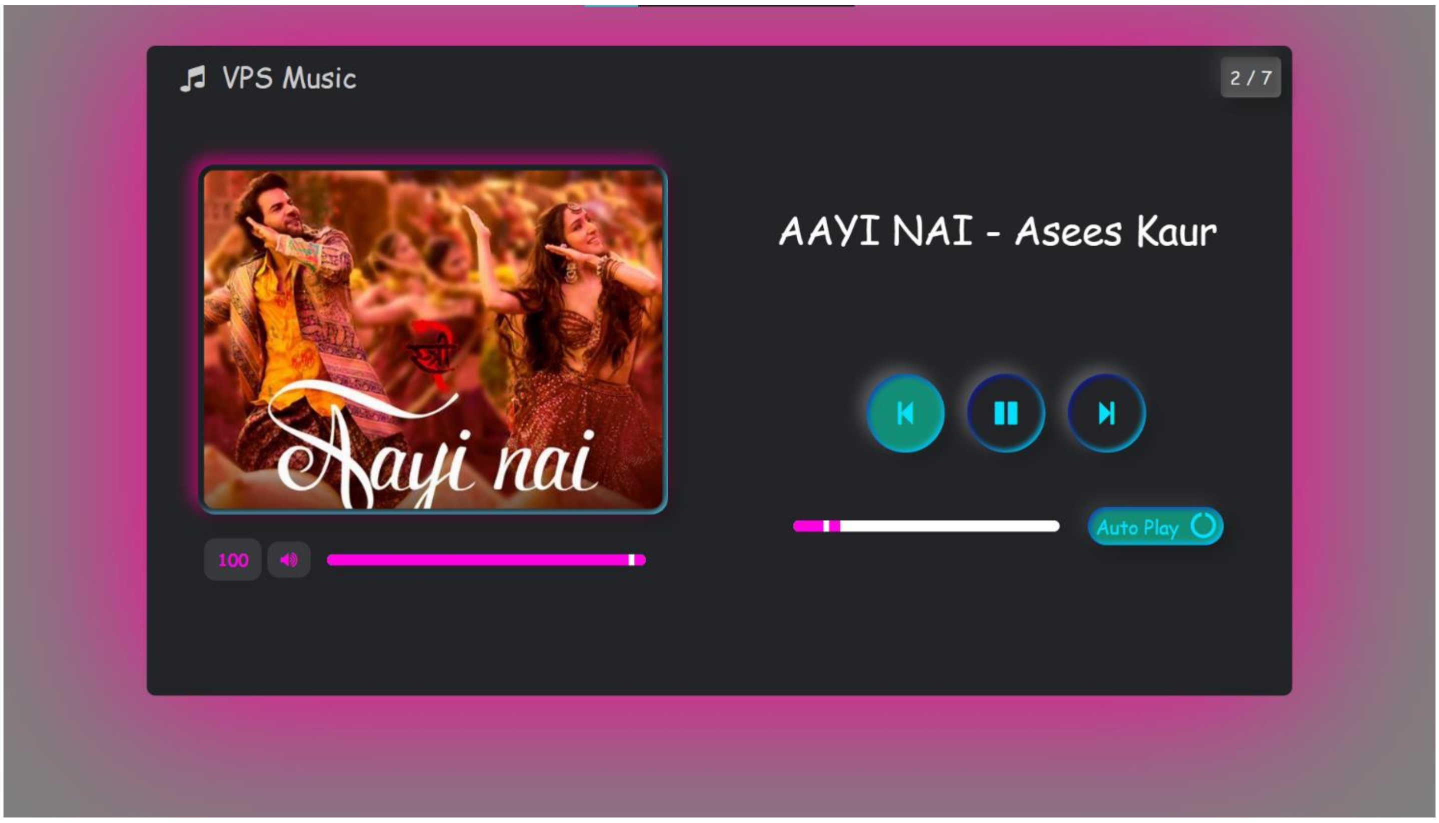
```
function load_track(index_no) {
  track.src = All_song[index_no].path;
  track_image.src = All_song[index_no].img;
  artist.innerHTML = All_song[index_no].singer;
  // title.innerText = All_song[index_no].name;
  track.load();

  total.innerHTML = All_song.length;
  present.innerHTML = index_no + 1;
  timer = setInterval(range_slider, 1000);
}

load_track(index_no);

//mute sound function
function mute_sound() {
  track.volume = 0;
  volume.value = 0;
  volume_show.innerHTML = 0;
}

// checking.. the song is playing or not
function justplay() {
  if (Playing_song == false) {
    playsong();
  } else {
    pausesong();
  }
}

// reset song slider
function reset_slider() {
  slider.value = 0;
}

// play song
function playsong() {
  track.play();
  Playing_song = true;
  play.innerHTML = '<i class="fa fa-pause"></i>';
}

//pause song
function pausesong() {
  track.pause();
  Playing_song = false;
  play.innerHTML = '<i class="fa fa-play"></i>';
}
```

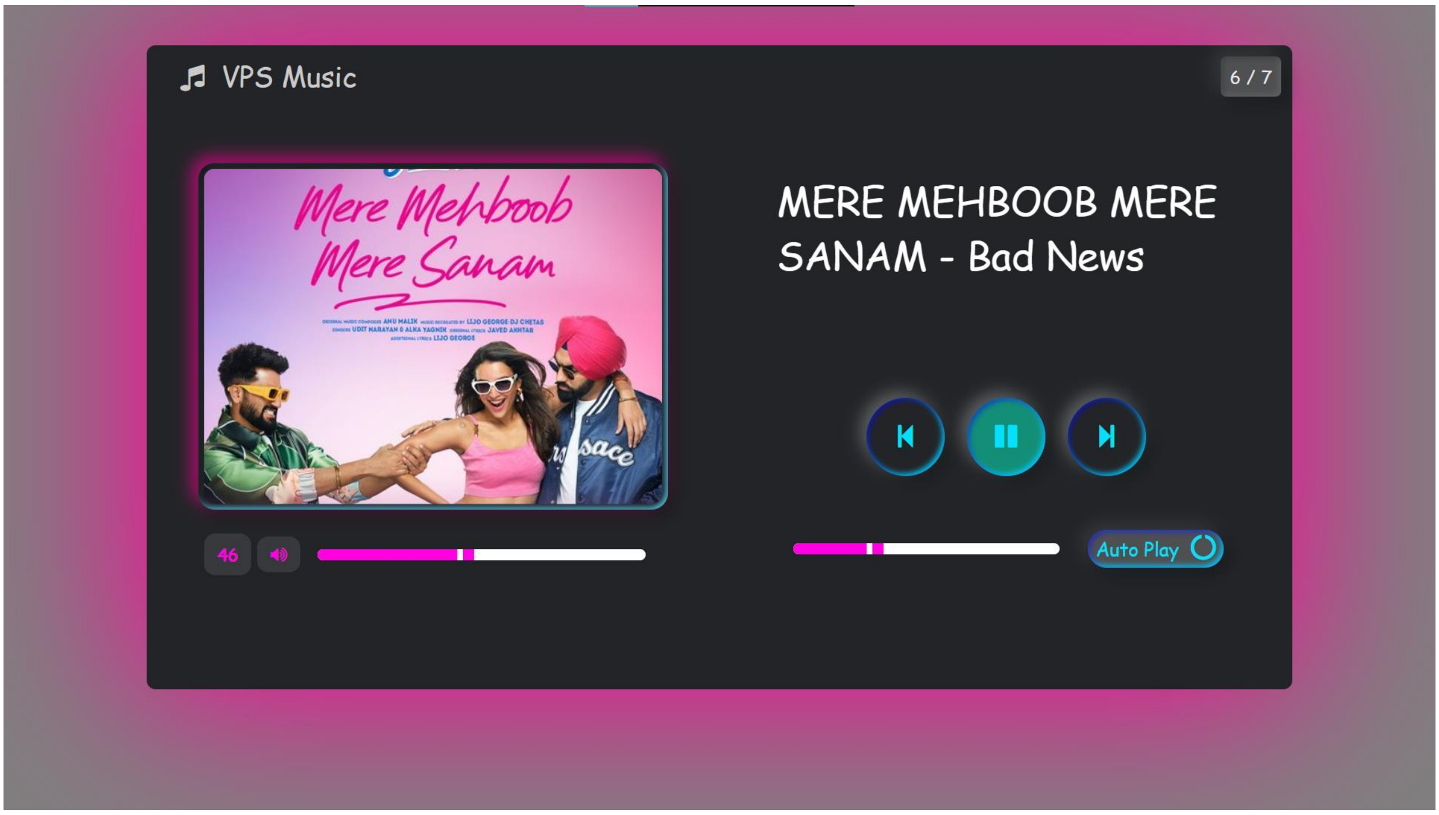
```
// next song
function next_song() {
  if (index_no < All_song.length - 1) {
    index_no += 1;
    load_track(index_no);
    playsong();
  } else {
    index_no = 0;
    load_track(index_no);
    playsong();
  }
}

// previous song
function previous_song() {
  if (index_no > 0) {
    index_no -= 1;
    load_track(index_no);
    playsong();
  } else {
    index_no = All_song.length;
    load_track(index_no);
    playsong();
  }
}

// change volume
function volume_change() {
  volume_show.innerHTML = recent_volume.value;
  track.volume = recent_volume.value / 100;
}

// change slider position
function change_duration() {

  slider_position = track.duration * (slider.value / 100);
  track.currentTime = slider_position;
}

// autoplay function
function autoplay_switch() {
  if (autoplay == 1) {
    autoplay = 0;
    auto_play.style.background = "rgba(255,255,255,0.2)";
  } else {
    autoplay = 1;
    auto_play.style.background = "#148F77";
  }
}

function range_slider() {
  let position = 0;

  // update slider position
  if (!isNaN(track.duration)) {
    position = track.currentTime * (100 / track.duration);
    slider.value = position;
  }

  // function will run when the song is over
  if (track.ended) {
    play.innerHTML = '<i class="fa fa-play" aria-hidden="true"></i>';
    if (autoplay == 1) {
      index_no += 1;
      load_track(index_no); playsong();}}}
```

1. Output :




VPS Music
5 / 7
KHUNDAY - Stree 2
KHUDAYA
93
Auto Play


VPS Music
2 / 7
AAYI NAI - Asees Kaur
Aayi nai
100
Auto Play



6. Reference :

https://chat.openai.com/chat
https://www.geeksorgeeks.org/
https://www.Googal.com/
https://www.github.com /

7. Resources Used :

Sr. no	Instruments	Specifications	Quantity	Remark
01	Computer system	Laptop I5, RAM 8 GB SSD 512 GB	2.	ok
02	Software	VS Code	1.	ok
03	Any other resources use	MS Word Chrome	1.	ok

8. Skill Development :

1. Communication skill
2. Leadership skill
3. Team work skill
4. Scientific approach development
5. Data collection skill
6. Research skill

9. Conclusion :

The development of the **Music Player using HTML, CSS, and JavaScript** demonstrates how modern web technologies can be effectively utilized to create a functional, responsive, and user-friendly media application. The player supports essential features such as play/pause controls, next/previous track navigation, volume adjustment, and a progress bar, providing a seamless user experience. The implementation highlights the power of JavaScript in managing multimedia content,

Prof.Jondhale D.R
(Project Guide)

Prof.Chaudhari N.K
(H.O.D) IT