NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

A MINI PROJECT REPORT

on

WEB DESIGN TECHNOLOGIES (24CSE361)

MUSIC PLAYLIST INTERFACE

Submitted by

P PUNITH

1NH24CS144

Under the guidance of

Ms. DIVYANSHI CHHABRA

Assistant Professor

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Academic Year: 2025-26 (ODD SEM)

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

CERTIFICATE

This is to certify that the mini project work titled "**Music Playlist interface"** is a bonafide work carried out by **P Punith (1NH24CS144)** in partial fulfillment of the degree of **Bachelor of Engineering** in **Computer Science and Engineering** of the New Horizon College of Engineering during the year **2025-2026.**

Signature of Guide   Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner   *Signature with date*

1.______________________________ ______________________________

2.______________________________ ______________________________

ABSTRACT

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, who’s constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman, New Horizon Educational Institutions, for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha,** Principal, New Horizon College of Engineering, for the constant support and encouragement.

I would like to thank **Dr. Anandhi R J**, Professor and Dean-Academics, NHCE, for her valuable guidance.

I would also like to thank **Dr. B. Rajalakshmi,** Professor and HOD, Department of Computer Science and Engineering, for the constant support.

I also express my gratitude to **Ms DIVYANSHI CHHABRA, Designation**, Department of Computer Science and Engineering, my mini project reviewer, for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

P PUNITH

1NH24CS144

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A music playlist interface in web design is a visually structured and interactive webpage that allows users to browse, manage, and play songs with ease. It combines functional UI elements—such as buttons, track lists, sliders, and album art—with aesthetic design principles like layout, color schemes, typography, and responsiveness.The goal of this interface is to create a smooth and engaging user experience by making music navigation simple, intuitive, and visually appealing. In this project, the music playlist interface is designed using modern web technologies such as HTML, CSS, and JavaScript to ensure usability, accessibility, and a clean interactive design across different devices.

1.1 PROBLEM DEFINITION

Many existing music interfaces are either visually cluttered, difficult to navigate, or lack personalization. Users often face challenges such as slow loading times, poor layout structure, non-responsive design on mobile devices, and confusing playback controls.
This project aims to solve these issues by developing a clean, responsive, and easy-to-use music playlist interface that provides smooth interaction, clear navigation, and visually appealing design, ensuring a more satisfying listening experience.

Key Problems Identified

1. Poor visual organization leading to difficulty in browsing songs.
2. Non-responsive layouts that perform badly on mobile screens.
3. Inconsistent or confusing playback controls.
4. Lack of user-friendly playlist management features.
5. Slow or inefficient interface interactions.
6. Outdated or unattractive color schemes and UI elements**.**

1.2 OBJECTIVES

1. To design a clean and modern interface for managing and playing music.
2. To ensure responsive design across desktops, tablets, and mobile devices.
3. To implement intuitive audio controls such as play, pause, next, previous, and volume.
4. To create a well-organized layout that allows users to easily browse and manage songs.
5. To improve user experience (UX) through smooth interactions and visually pleasing design.

6.

6. To develop the interface using HTML, CSS, and JavaScript with simple and efficient code.

1.3 METHODOLOGIES TO BE FOLLOWED

1. Requirement Analysis

- Identify essential features: playlist display, audio player, controls, animations.
- Decide target users and devices.

2. UI/UX Design

- Create wireframes and layouts.
- Choose color palettes, typography, icons, and spacing.
- Use design principles: alignment, contrast, balance, and readability.

3. Front-End Development

- Structure content with HTML.
- Style the interface using CSS (flexbox, grid, animations).
- Add interactivity and playback features using JavaScript.

4. Responsiveness

- Apply media queries for mobile and desktop.
- Test layout adaptability across various screen sizes.

5. Integration & Optimization

- Optimize images and audio for faster loading.
- Improve UI performance and reduce code redundancy.

CHAPTER 2

FUNDAMENTALS OF THE LANGUAGES USED

The **Music Playlist Interface** on Web Design covers the foundational languages of web development: HTML, CSS, and JavaScript. These languages are essential for creating structured, styled, and interactive web pages.

2.1 HTML (Hyper Text Markup Language)

HTML is the standard markup language for creating web pages. It structures the content of a webpage using elements enclosed in tags. HTML (Hyper Text Markup Language) was created by Tim Berners-Lee in 1989 as part of the World Wide Web project at CERN, aimed at enabling scientists to share documents through hyperlinks. The first version of HTML was introduced in 1991, consisting of 18 basic tags like <p> and <b> .

In 1995, HTML 2.0 became the first standardized version, introducing new features like forms and tables. Subsequent updates, such as HTML 3.2 in 1997, added support for CSS, while HTML 4.0 in 1999 integrated multimedia elements and scripting capabilities. During the late 1990s and early 2000s, XHTML emerged as a stricter version of HTML but faced adoption challenges. HTML5, introduced in 2014, revolutionized the web by adding semantic elements, multimedia capabilities, and cross-platform support, cementing its role as the cornerstone of modern web development.

Key Features:

• **Structure**: Defines elements like headings, paragraphs, lists, and images.

• **Links**: Connects to other documents using hyperlinks.

• **Forms**: Captures user input.

2.2 HTML TAGS

- <html>: Defines the root of the HTML document.
- <head>: Contains metadata (like the title, links to stylesheets, and scripts) which is not displayed on the web page itself.
- <title>: Defines the title of the document, which appears in the browser's title bar or tab.
- <body>: Defines the main content of the HTML document that is visible to the user.

Text and Formatting Tags

These tags are used to format text content.

- <h1> to <h6>: Define HTML headings, with <h1> being the most important and <h6> the least.
- <p>: Defines a paragraph.
- <br>: Inserts a single line break (self-closing tag).
- <b>: Defines bold text (stylistic only).
- <strong>: Defines important text (semantic, usually displayed in bold).
- <i>: Defines a part of text in an alternate voice or mood (stylistic, usually displayed in italics).
- <em>: Defines emphasized text (semantic, usually displayed in italics).

Links and Images

These tags are essential for navigation and multimedia.

- <a>: Defines a hyperlink, used to link to other pages or resources.
- <img>: Embeds an image (self-closing tag).

Lists Tags

These tags are used to create lists of items.

- <ul>: Defines an unordered (bulleted) list.
- <ol>: Defines an ordered (numbered or lettered) list.
- <li>: Defines a list item within <ul> or <ol> tags.

Semantic and Layout Tags

HTML5 introduced semantic tags that clearly define the structure and meaning of the content.

- <header>: Defines a container for introductory content or a set of navigational links.
- <nav>: Defines navigation links.
- <main>: Specifies the main content of a document.
- <section>: Defines a section in a document (e.g., chapters, headers, footers).

- <article>: Defines independent, self-contained content.
- <footer>: Defines a footer for a document or section.

Forms and Tables

These tags are used for user input and data presentation.

- <form>: Defines an HTML form for user input.
- <input>: Creates form inputs (e.g., text fields, checkboxes, radio buttons).
- <table>: Defines a table.
- <tr>: Defines a table row.
- <th>: Defines a table header cell.
- <td>: Represents a standard table data cell.

.

2.3 CSS

CSS (Cascading Style Sheets) was introduced in 1996 by the World Wide Web Consortium(W3C) as a solution to separate the content of web pages from their presentation. The concept was first proposed by HAkon Wium Lie in 1994, aiming to provide web developers with a way to control the appearance of web pages without altering their HTML structure. The first specification, CSS1, included basic styling features such as font customization, text alignment, and margins. In 1998, CSS2 expanded its capabilities with features like positioning, media types, and table styling. However, inconsistent browser support slowed adoption during the early 2000s. CSS3, introduced as a modular update, brought transformative features like animations, transitions, and responsive design through media queries. Today, CSS remains a fundamental technology in web development, empowering developers to create visually engaging and responsive websites. It defines the presentation of HTML documents, including layout, colors, fonts, and spacing, ensuring a seamless user experience across devices.

Key Features:

• **Selectors:** Targets HTML elements (e.g., p, #id, .class).

• **Box Model**: Defines padding, border, and margin for layout.

• **Flexibility**: Enables responsive design and animations.

2.4 JAVASCRIPT

JavaScript is a high-level, lightweight, and versatile programming language primarily used to create interactive and dynamic web applications. It enables developers to add features such as animations, form validations, and real-time content updates. Initially developed by Brendan Eich at Netscape in 1995, it was originally called Mocha, later renamed LiveScript, and eventually JavaScript to align with the popularity of Java at the time. JavaScript gained widespread adoption with the introduction of ECMAScript standards in 1997, which ensured consistency across implementations. Over the years, JavaScript has evolved significantly, adding modern features like asynchronous programming, modularization, and extensive libraries, making it a cornerstone of modern web development alongside HTML and CSS.

Key Features and examples:

- **Basics:** Understanding variables (var, let, const), data types, operators, and expressions.
- **Functions**: Writing reusable blocks of code to perform specific tasks.
- **DOM Manipulation**: Selecting and modifying elements dynamically.
- **Event Handling**: Responding to user actions like clicks and keypresses.
- **Form Validation**: Ensuring proper user input.

2.5 XHTML

XHTML or Extensible HyperText Markup Language is a mix of HTML and XML, very similar to HTML but stricter. It's like a rulebook for creating web pages that browsers easily understand. Unlike HTML, you have to be careful and follow the rules exactly. Most browsers support it. Just think of it as a more precise way to write web code. It was developed by the World Wide Web Consortium (W3C) and helps web developers transition from HTML to XML. With XHTML, developers can enter the XML world with all its features while still ensuring backward and future compatibility of the content. The XHTML family includes three document types; the first is XHTML 1.0, which was recommended by W3C on January 26, 2000. The second is XHTML 1.1, which was recommended by W3C on May 31, 2001.The third is XHTML5, a standard used for developing an XML adaptation of the HTML5 specification. An XHTML document must have an XHTML <!DOCTYPE> declaration.

The elements of XHTML :

- **<html>**: The root element that encloses the entire document and must contain an xmlns attribute.
- **<head>**: Contains metadata about the document that does not appear in the main browser window.
  - **<title>**: Specifies the title of the document, displayed in the browser's title bar or tab.

- **<meta />**: Provides meta-information like character set and keywords.
- **<link />**: Links to external resources, such as stylesheets (CSS).
- **<script>**: Links to or contains executable scripts, such as JavaScript.
- **<body>** (or <frameset> for frameset DTD): Contains the visible content of the web page.

2.6 XML

Extensible Markup Language (XML) is a type of markup language that establishes a set of guidelines for encoding texts in a way that is both machine- and human-readable. For storing and transferring data on the web and in many other applications, XML is widely used. XML steps in as a versatile tool for encoding and organizing data in a way that both humans and machines can comprehend.

XML emerged in the late 1990s as a revolutionary concept in the evolving landscape of the internet. Before XML, HTML served as the predominant language for web content, but it lacked the flexibility needed for complex data representation. XML arrived as a solution, offering a standardized format for expressing diverse types of data in a hierarchical structure.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

1. Processor:
• Minimum: Dual-core processor (e.g., Intel Core i3 or equivalent).
• Recommended: Quad-core processor or higher for multitasking.
2. RAM:
• Minimum: 2 GB (sufficient for lightweight operations).
3. Storage:
• A minimum of 100 MB of free space (to store the HTML file, assets, and browser cache).
4. Display:
• Screen resolution of at least 1024x768 to accommodate the quiz UI.
• A modern display supporting standard web colors.
5.Input Devices:
• Keyboard and mouse (or touch input for tablets and mobile devices).
6. Internet Connection:
• Required since some animations in the document are procured from sources in the internet.

3.2 SOFTWARE REQUIREMENTS

1. Operating System:
• Windows (7 or higher), macOS (10.12 or higher), or a modern Linux distribution.
• For mobile: Android 5.0+ or iOS 10+.
2. Web Browser: • Modern web browsers like:
• Google Chrome (latest version preferred).
• Mozilla Firefox (latest version preferred).
• Microsoft Edge (latest version).
• Safari (for macOS/iOS).
• Ensure the browser supports ES6 JavaScript and CSS3 features.
3. Text Editor/IDE (for development or modification):
• Notepad, Visual Studio Code, Sublime Text, Atom, or any basic text editor.
4. Web Server (Optional):
• For local hosting (if not opening directly in a browser):
• Use a lightweight server like Python's built-in HTTP server (python -m http.server) or tools like XAMPP.

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

1. User-Friendly Navigation

- Provide an intuitive layout that allows users to easily browse, search, and select songs.
- Ensure clear visibility of playlist items, play controls, and navigation menus.

2. Attractive & Modern UI

- Use visually appealing colors, typography, and spacing.
- Maintain a balance between aesthetics and simplicity to improve usability.

3. Responsive Design

- Interface should automatically adapt to various screen sizes – desktop, tablet, and mobile.
- Maintain consistent functionality and readability across devices.

4. Smooth Playback Controls

- Provide seamless play, pause, next, previous, repeat, and shuffle options.
- Ensure smooth transitions between tracks without delays.

5. Efficient Content Organization

- Display playlists in a well-structured manner.
- Use categories, filters, or sorting features for easy navigation of songs.

6. Accessibility

- Ensure the interface is usable for all users, including those with disabilities.
- Add features like keyboard navigation, readable fonts, sufficient contrast, and alternative text.

7. Fast Loading & Performance Optimization

- Reduce load time by optimizing images, code, and audio files.
- Aim for smooth interactions without lag.

8. Personalization

- Allow users to create, edit, and manage custom playlists.
- Optionally include user preferences like theme mode (dark/light) or favorite tracks.

9. Consistent Design Language

- Maintain uniform colors, icons, buttons, and spacing throughout the interface.
- Follow web design standards for familiarity and ease of use.

10. Security & Data Protection

- Ensure secure handling of user-generated playlists and account data (if login is used).
- Protect media access from unauthorized usage.

CHAPTER 5

IMPLEMENTATION

1. Setting Up the HTML Structure

The first step in implementation is creating the basic layout using **HTML**. The interface is divided into the following components:

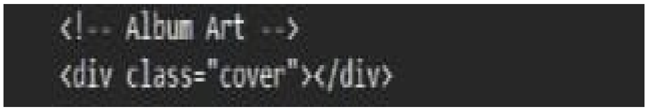
- **Main Container (player-box)**

This holds the entire music player including:

- Album cover
- Song information
- Playback controls
- Playlist

It acts as the main UI block.

- **Album Cover Section**



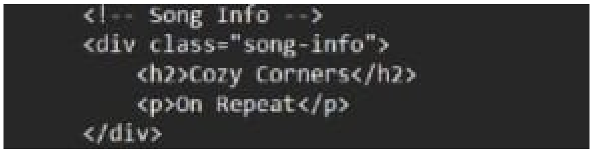
```
<!-- Album Art -->
<div class="cover"></div>
```

Fig 5.1 CODE SNIPPET FOR ALBUM COVER

This area displays the currently playing song's album art using a background image.

- **Song Information Section**

Displays the song title and subtitle (like artist name or status – "On Repeat").



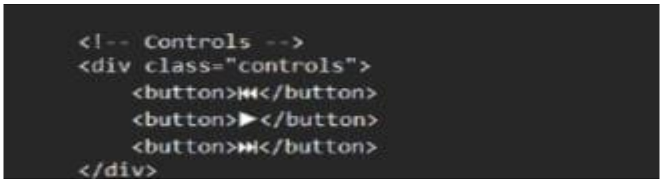
```
<!-- Song Info -->
<div class="song-info">
    <h2>Cozy Corners</h2>
    <p>On Repeat</p>
</div>
```

Fig 5.2 CODE SNIPPET FOR SONG INFORMATION SECTION

- **Controls Section**

Contains the play, previous, and next buttons:

<button>⏮</button> <button>▶</button> <button>⏭</button>



```
<!-- Controls -->
<div class="controls">
    <button>⏮</button>
    <button>▶</button>
    <button>⏭</button>
</div>
```

Fig 5.3 CODE SNIPPET FOR CONROLS SECTION

- **Playlist Section**

Each track entry includes a small image (thumbnail) and the track title:

```
<div class="track">
<img src="...">
<div class="track-title">Song Name</div>
</div>
```

This forms the skeleton of the interface.

2. Applying CSS for Styling and Layout

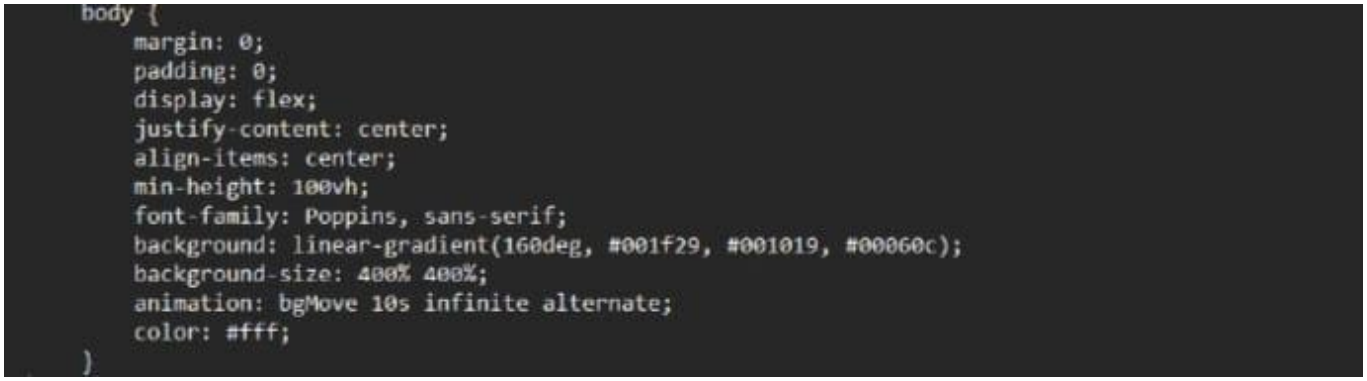
CSS is used to make the interface look attractive and modern.

- **Global Styling**

The body is styled with:

- Animated gradient background
- Center alignment using flexbox
- Aesthetic font (Poppins)
- Full-screen height

This creates a clean and attractive background for the music UI.



```
body {
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    font-family: Poppins, sans-serif;
    background: linear-gradient(160deg, #001f29, #001019, #00060c);
    background-size: 400% 400%;
    animation: bgMove 10s infinite alternate;
    color: #fff;
}
```

Fig 5.4 CODE SNIPPET FOR APPLYING CODE STYLING

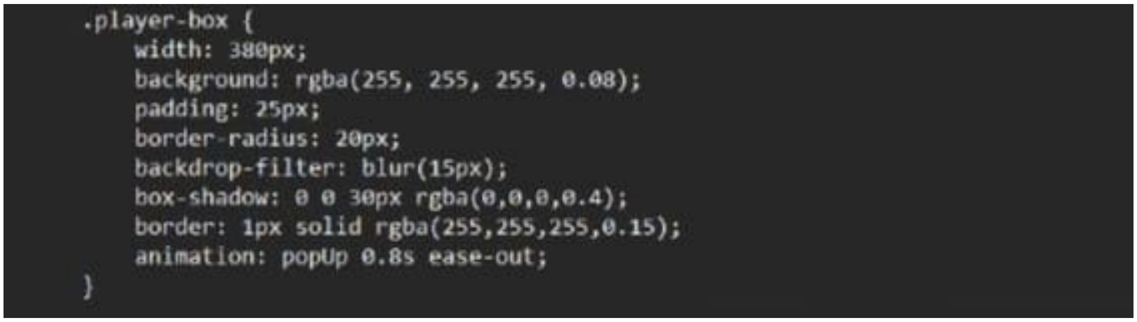
Music Playlist Interface

➢ Creating the Glassmorphism Player Box

background: rgba(255, 255, 255, 0.08);
backdrop-filter: blur(15px);
This gives:

- Glass-like transparent effect
- Smooth rounded corners
- Drop shadow

This is a modern design technique widely used in music UIs.

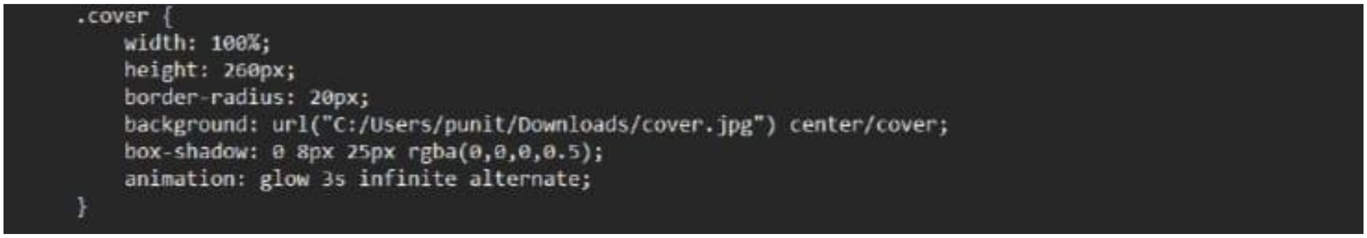


```
.player-box {
    width: 380px;
    background: rgba(255, 255, 255, 0.08);
    padding: 25px;
    border-radius: 20px;
    backdrop-filter: blur(15px);
    box-shadow: 0 0 30px rgba(0,0,0,0.4);
    border: 1px solid rgba(255,255,255,0.15);
    animation: popUp 0.8s ease-out;
}
```

Fig 5.5 CODE SNIPPET FOR CREATING PLAYER BOX

Used as a dynamic background image that:

- Matches the width of the player
- Has glowing animation
- Has shadow for depth



```
.cover {
    width: 100%;
    height: 260px;
    border-radius: 20px;
    background: url("C:/Users/punit/Downloads/cover.jpg") center/cover;
    box-shadow: 0 8px 25px rgba(0,0,0,0.5);
    animation: glow 3s infinite alternate;
}
```

Fig 5.6 CODE SNIPPET FOR ALBUM COVER STYLING

➢ Buttons and Controls

The buttons are implemented using:

- Circular shapes
- Hover animations (scale-up effect)
- Transparent backgrounds
- Smooth transitions

This creates interactive and visually interesting controls.

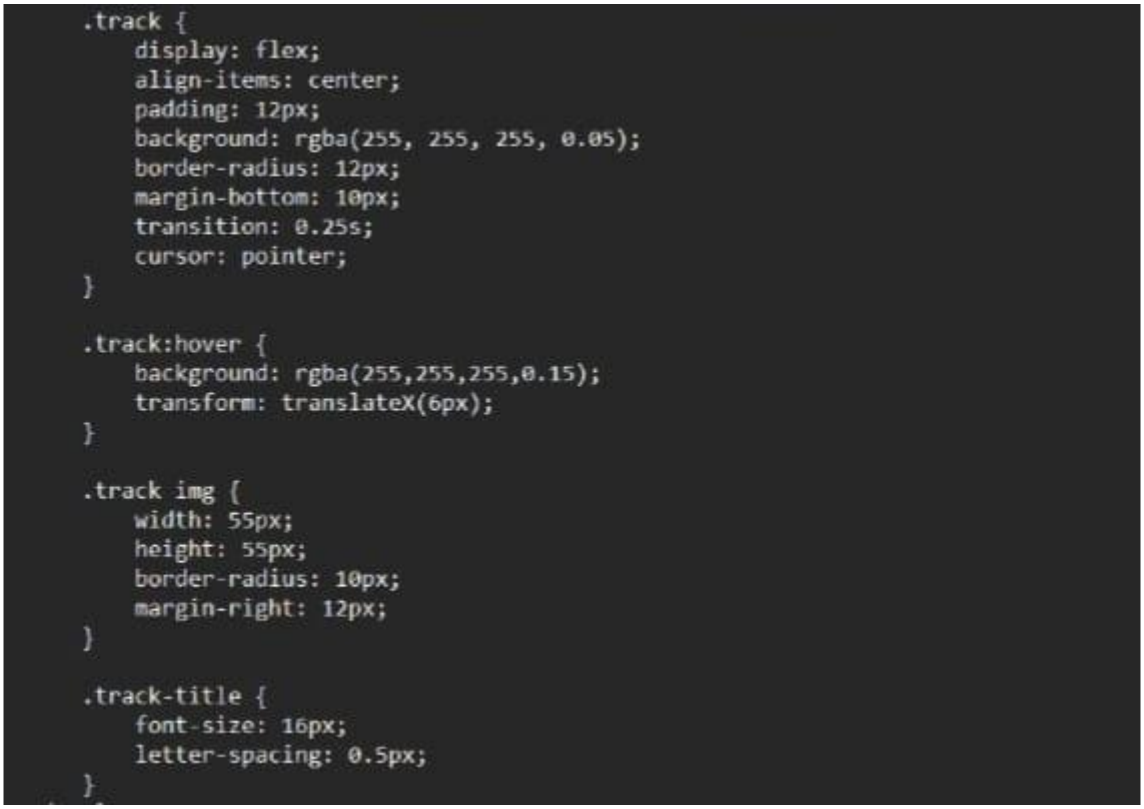
Music Playlist Interface

➢ **Playlist Design**

Each track is styled as a horizontal card with:

- Song cover thumbnail
- Title section
- Hover animation (sliding effect)

This improves the user experience and allows easy selection of songs.



```
.track {
    display: flex;
    align-items: center;
    padding: 12px;
    background: rgba(255, 255, 255, 0.05);
    border-radius: 12px;
    margin-bottom: 10px;
    transition: 0.25s;
    cursor: pointer;
}

.track:hover {
    background: rgba(255,255,255,0.15);
    transform: translateX(6px);
}

.track img {
    width: 55px;
    height: 55px;
    border-radius: 10px;
    margin-right: 12px;
}

.track-title {
    font-size: 16px;
    letter-spacing: 0.5px;
}
```

Fig 5.7 CODE SNIPPET FOR SLIDDING EFFECT (ANIMATION)

3. Adding Animations for Aesthetic Effects

Several CSS animations are implemented:

Background Animation

Creates a moving gradient effect.

Pop-Up Animation

The player box smoothly appears when the page loads.

➢ **Glow Animation**

The album cover glows subtly to make the UI visually engaging.

These animations enhance the interface’s modern look and feel.

4. Image Integration

All images (album cover + track thumbnails) are linked through:

```
background: url("C:/...");                    or
                                    <img src="C:/...">
```

These images make the playlist realistic and visually appealing.
(For deployment on a website, these should be replaced with online image URLs.)

```
<div class="track">
        <img src="C:/Users/punit/Downloads/starboy.png">
        <div class="track-title">Starboy - The Weekend, Draft Punk</div>
     </div>
```

5. Responsive Behavior

The use of:

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

ensures the UI scales properly on:

- Phones
- Tablets
- Computers

The flexible units and centered layout help maintain consistent appearance across screens.

CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Music playlist interface</title>

<style>
    /* ---------------- GLOBAL ---------------- */
    body {
        margin: 0;
        padding: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        min-height: 100vh;
        font-family: Poppins, sans-serif;
        background: linear-gradient(160deg, #001f29, #001019, #00060c);
      background-size: 400% 400%;
        animation: bgMove 10s infinite alternate;
        color: #fff;
    }

    @keyframes bgMove {
        from { background-position: 0% 50%; }
        to { background-position: 100% 50%; }
    }

    /* ---------------- CONTAINER ---------------- */
    .player-box {
        width: 380px;
        background: rgba(255, 255, 255, 0.08);
        padding: 25px;
        border-radius: 20px;

 backdrop-filter: blur(15px);
```

```
box-shadow: 0 0 30px rgba(0,0,0,0.4);
      border: 1px solid rgba(255,255,255,0.15);
      animation: popUp 0.8s ease-out;
   }

   @keyframes popUp {
      from { transform: scale(0.8); opacity: 0; }
      to { transform: scale(1); opacity: 1; }
   }

   /* ---------------- ALBUM COVER ---------------- */
   .cover {
      width: 100%;
      height: 260px;
      border-radius: 20px;
      background: url("C:/Users/punit/Downloads/cover.jpg") center/cover;
      box-shadow: 0 8px 25px rgba(0,0,0,0.5);
      animation: glow 3s infinite alternate;
   }

   @keyframes glow {
      from { box-shadow: 0 0 15px #a855f73a; }
      to   { box-shadow: 0 0 25px #a855f7; }
   }

   /* ---------------- TEXT ---------------- */
   .song-info {
      text-align: center;
      margin-top: 20px;
   }

   .song-info h2 {
      margin: 0;
      font-size: 22px;
      letter-spacing: 1px;
   }

   .song-info p {
      margin: 3px;
```

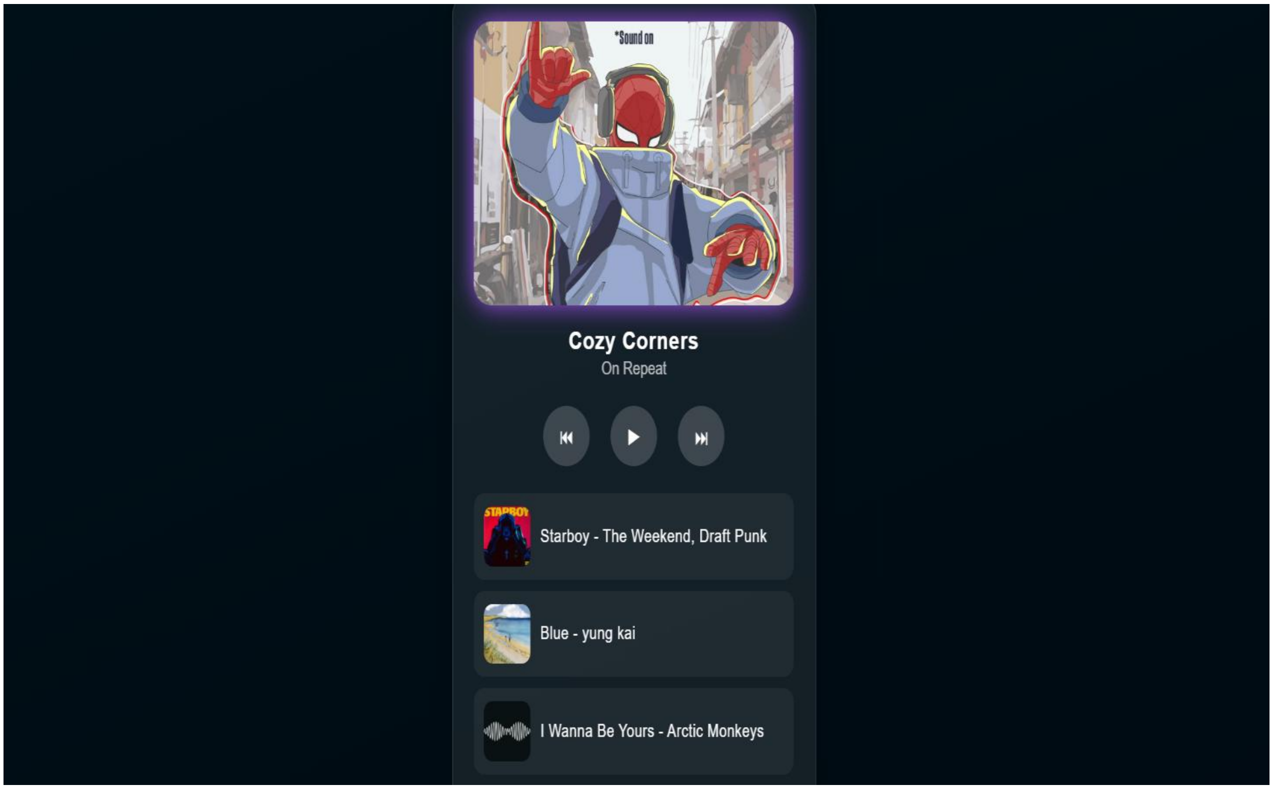
```
opacity: 0.7;
  }


/* ---------------- CONTROLS ---------------- */
  .controls {
     margin: 25px auto;
     display: flex;
     justify-content: center;
     gap: 25px;
  }

  .controls button {
     width: 55px;
     height: 55px;
     border-radius: 50%;
     border: none;
     background: rgba(255, 255, 255, 0.1);
     backdrop-filter: blur(6px);
     color: #fff;
     font-size: 18px;
     cursor: pointer;
     transition: 0.2s;
  }

  .controls button:hover {
     transform: scale(1.15);
     background: rgba(255, 255, 255, 0.2);
  }

  /* ---------------- PLAYLIST ---------------- */
  .playlist {
     margin-top: 25px;
  }

  .track {
     display: flex;
     align-items: center;
     padding: 12px;
```

```
background: rgba(255, 255, 255, 0.05);
    border-radius: 12px;

    margin-bottom: 10px;
    transition: 0.25s;
    cursor: pointer;
  }


 .track:hover {
    background: rgba(255,255,255,0.15);
    transform: translateX(6px);
  }

  .track img {
    width: 55px;
    height: 55px;
    border-radius: 10px;
    margin-right: 12px;
  }

  .track-title {
    font-size: 16px;
    letter-spacing: 0.5px;
  }
</style>
</head>

<body>

<div class="player-box">

  <!-- Album Art -->
  <div class="cover"></div>

  <!-- Song Info -->
  <div class="song-info">
    <h2>Cozy Corners</h2>
    <p>On Repeat</p>
    </div>

    <!-- Controls -->
    <div class="controls">
        <button>⏮</button>
        <button>▶</button>
        <button>⏭</button>
    </div>

    <!-- Playlist -->
    <div class="playlist">


  <div class="track">
            <img src="C:/Users/punit/Downloads/starboy.png">
            <div class="track-title">Starboy - The Weekend, Draft Punk</div>
        </div>

        <div class="track">
            <img src="C:/Users/punit/Downloads/blue.png">
            <div class="track-title">Blue - yung kai</div>
        </div>
       <div class="track">
            <img src="C:/Users/punit/Downloads/wanna be yours.png">
            <div class="track-title">I Wanna Be Yours - Arctic Monkeys </div>
        </div>
<div class="track">
            <img src="C:/Users/punit/Downloads/sahiba.png">
            <div class="track-title">Sahiba - Aditya Rikhari </div>
        </div>
</div>
</div>

</body>
</html>
```

CHAPTER 6

RESULTS

A modern dark-themed music player UI with glowing album art, playback buttons, and a clean playlist list.


*Sound on
Cozy Corners
On Repeat
⏮ ▶ ⏭
STARBOY
Starboy - The Weekend, Draft Punk
Blue - yung kai
I Wanna Be Yours - Arctic Monkeys

6.1 RESULT OF MUSIC PLAYLIST INTERFACE

CHAPTER 7

CONCLUSION

The music playlist interface provides a clean, user-friendly, and visually appealing way for users to access, organize, and enjoy their songs. With clear navigation, intuitive controls, and an attractive layout, it enhances the overall listening experience. By combining functional design with aesthetic elements, the interface ensures convenience, efficiency, and an engaging interaction for users while managing their music. The development of the music playlist interface marks the successful completion of a project that combines creative design, structured implementation, and user-centered thinking to deliver a meaningful digital experience. Throughout the project, the central focus has been to understand how users interact with music applications and how interface design can enhance their overall engagement. The final interface reflects a careful balance of simplicity, aesthetics, and functionality, ensuring that users can browse, organize, and enjoy their music collections without confusion or unnecessary complexity.

One of the key achievements of this interface is its clarity. A music application must communicate information—such as song titles, durations, artists, and controls—quickly and without overwhelming the user. The design prioritizes readability with clean layouts, consistent spacing, and well-structured sections. This allows the interface to guide users naturally, reducing cognitive load and making interactions feel effortless. The organized visual hierarchy also ensures that primary actions, such as playing a song or navigating through playlists, remain at the forefront.

Ultimately, the music playlist interface stands as a complete demonstration of thoughtful planning, effective design choices, and careful execution. It not only fulfills the basic requirements of organizing and playing music but also enhances user interaction through its visually appealing and user-friendly layout. The project illustrates how even a simple application can benefit from attention to usability, structured design, and aesthetic values. By creating an interface that is clear, functional, and enjoyable to use, the project successfully meets its goals and provides a strong foundation for further improvements, additional features, or future enhancements.

REFERENCES

- https://devdevout.com/css/css-animated-backgrounds
- https://www.w3schools.com/tags/tag_html.asp
- https://github.com/topics/web-development-project
- https://www.w3schools.com/js/js_object_property.asp
- https://developer.mozilla.org/en-US/docs/Learn/CSS