18/06/2024

PROJECT REPORT

Project Mentor: Parimal sir
Project submitted by: Hiral Kshatri (202326900042)
Jash Kevadiya (20232600039)
Naisargi Prajapati (202326900095)

Department of Animation It-IMS & Mobile Application
GUJARAT UNIVERSITY

About University

The Gujarat University is a statewide institution affiliating with many reputed colleges across the state of Gujarat, India. It has been given a B++ ranking by National Assessment and Accreditation Council (NAAC).

The Gujarat University was conceived in the nineteen twenties in the minds of the public-spirited. The University was set up after independence for India. In 1949; the University was incorporated under the Gujarat University Act of the State Government ‘as a teaching and affiliating University.

Gujarat University is an affiliating University at the undergraduate level, while it is a teaching one at the post-graduate level. 4 Indeed, the responsibility for post-graduate instruction has been statutorily given to the University, and accordingly it has evolved a plan of coordinated instruction under the direct control and supervision of the University so as to ensure efficient and diversified instruction.

INDEX

Introduction

In department of animation under b.sc IT course in sem II we are given project in extrenal exam. In which project was asked to be done solo or in group. In which I(202326900042), my classmate Jash Kevadiya(202326900039) and Naisargi Prajapati(20232600095) have worked together under this project.

I express my profound gratitude to Mr. Parimal Sir (Assistant professor), Department of Animation for the valuable help and guidance in the preparation of this Report. I would like to extend my sincere thanks to all Lab Assistants and all other staff members of B.sc IT Data Management & visual insight. Finally, I would also wish to record my gratefulness to all my friends and classmates for their help.

We have no valuable words to express our thanks to everyone for the valuable advice and suggestions for the corrections, modification, and improvement did enhance the perfection in performing our job well.

Abstract

The Spotify clone is a and maintains project repository of all the relevant information about any song which is available in the Spotify and makes it easy for a user. In the Spotify a user can simply search a song and get information about song as per need. As we can see that manually search it is difficult for user to search a song in playlist. Spotify clone provides user a simple and user friendly where a user can easily access relevant information of song. The project is done with the help of Html, CSS, JavaScript.

Problem in the Present System

- User may not get proper information about how much song is available in the playlist.
- User have to search song one by one in playlist.
- It takes lot of time and effort.

After Implementation of the project

- It will become easy for the user of spotify to access information of song which is available in the playlist.
- Provide secured system to the user.
- It do not takes of lot time and effort.

Objective

A Spotify Clone is a music streaming platform designed to replicate the features and functionalities of the popular music service, Spotify. It offers users the ability to stream a vast library of music, create playlists, and enjoy a personalized music experience.

The main objective of this project SPOTIFY CLONE is to develop easy to use and secure system that provide information of song which is available in playlist.

Scope

The SPOTIFY CLONE has a large scope in future because

1. It provide easy and user-friendly interface.
2. It maintains privacy of user and provides
3. It saves lot of time and requires less efforts.
4. It provides relevant information to user.
5. Song can be downloaded on the device.

Hardware /Software Requirements

Development Requirements:

- Windows 7 or higher
- Size:530 MB
- Stable network
- Minimum 2 GB RAM
- Minimum 16 GB ROM

Software Requirements

- Visual Studio Code

Technology Used:

- HTML
- CSS
- JAVASCRIPT

System Planning

Survey of technologies

1. HTML:

The provided HTML document outlines the structure of a Spotify-like web player interface. Here's a detailed breakdown of its components:

HTML Structure

- **Head Section**
  - **Meta Tags:** Includes charset and viewport settings for responsive design.
  - **Stylesheets**: Links to style.css and utility.css for styling.
  - **Favicon**: Links to a Spotify icon (spoify_icon.ico).
  - **Title:** Sets the page title to "Spotify - Web Player: Music for everyone".

- **Body Section**
  - **Main Container:** The main content is wrapped in a div with the classes container flex bg-black.

- **Left Sidebar:** A sidebar with class left containing:
  - **Home Section:** Includes a logo and navigation items (Home, Search).
  - **Library Section:** Titled "Your Library" with placeholders for a song list and footer links (Legal, Safety & Privacy, Cookies, About Ads, Accessibility).
  - **Right Content Area:** Main content area with class right bg-grey p1 containing:
    - **Header:** Contains navigation controls and buttons for "Sign up" and "Log in".
    - **Playlist Section:** Titled "Spotify-Playlist", displaying multiple playlist cards with images, titles, and descriptions.
    - **Playbar:** A control bar at the bottom with seek bar, song info, playback controls (previous, play, next), volume control, and song time display.

Notes on Specific Elements

- **SVG Icons:** Used for navigation arrows and playback controls to maintain scalability and visual clarity.
- **Playlist Cards:** Each card includes a play button (SVG), image, title, and description. The cards are differentiated using the data-folder attribute.
- **Footer Links:** Each link points to various Spotify legal and policy pages.

External Files

- **Stylesheets:** The design relies on style.css and utility.css for styling.
- **JavaScript:** Functionality is managed by script.js.

Example Usage

- This document can be a basis for a web-based music player, mimicking Spotify's interface. With proper backend integration and JavaScript functionality, it could dynamically load playlists, handle user authentication, and manage playback controls.

❖ **Potential Improvements**

1. **Accessibility:** Ensure all interactive elements are accessible via keyboard and screen readers.

2. SEO: Add relevant meta tags and improve content semantics for better search engine optimization.

3. Responsiveness: Ensure the layout adjusts well on various screen sizes.

4. Interactivity: Enhance with JavaScript for dynamic content loading and user interactions.

2. CSS:

General Styles (style.css)

- **Fonts:** Uses the "Roboto" font family from Google Fonts.
- **Variables:** Defined a custom property --a initialized to 0.
- **Reset:** Resets margin and padding for all elements.
- **Body:** Sets background to black and text color to white.
- **Containers:**
  - left and right have specific width settings and padding.
  - home and library sections have padding and background color.

❖ **Specific Styling:**

- **Home List Items:** Flexbox layout with gap and padding; bold text.
- **Library Footer:** Flexbox layout with small font size, grey color, and gap; positioned at the bottom.
- **Header**: Flexbox layout for spacing between elements; background color set.
- **Card Container:** Flexbox layout for playlist cards; supports wrapping and scrolling with a max height.

- **Cards:** Relative positioning, background color, padding, border radius, and hover effects.
- **Play Bar:** Fixed position with styles for seek bar, song info, control buttons, and volume control.

Utility Styles (utility.css)

- **Flexbox Utilities:** Classes for display: flex, justify-content: center, and align-items: center.
- **Background and Color Utilities:** Classes for setting background color to black and text color to white.
- **Border and Rounded Corners:** Classes for borders and rounded corners.
- **Margin and Padding:** Utility classes for consistent spacing.
- **Scrollbar Customization:** Custom styles for scrollbar appearance.

3. JAVASCRIPT:

- **Variables and Functions**

  - **CurrentSong:** An instance of the Audio object to manage audio playback.
  - **songs:** An array to store the list of songs fetched from the server.
  - **Currfolder:** A string to store the current folder path.
  - **minutes(seconds)** A helper function to format time in minutes and seconds.

- **getSongs(folder)**

1. Fetch songs: Makes an HTTP request to fetch the list of songs from the specified folder.
2. Parse response: Converts the response into HTML, extracts song links ending in .mp3, and stores them in the songs array.
3. Display songs: Updates the song list UI with the fetched songs and attaches click event listeners to each song to handle playback.

- **playMusic(track, pause = false)**

1. Set source: Sets the src of the currentSong object to the selected track.
2. Play/Pause: Plays the song unless the pause parameter is true.
3. Update UI: Updates the current song information and the initial time display.

- **displayalbums()**

1. Fetch albums: Makes an HTTP request to fetch the list of albums (folders) in the songs directory.
2. Display albums: Processes the response to identify folders and potentially display them (though the function seems incomplete as it doesn't update the UI).

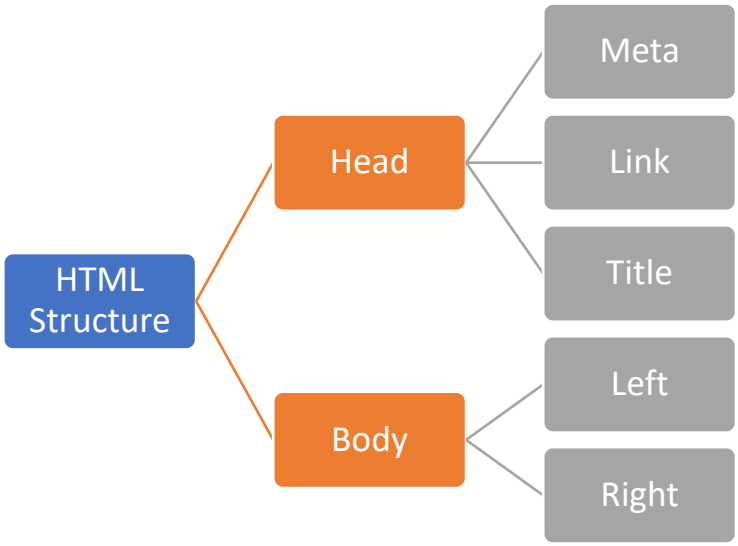
- **main()**

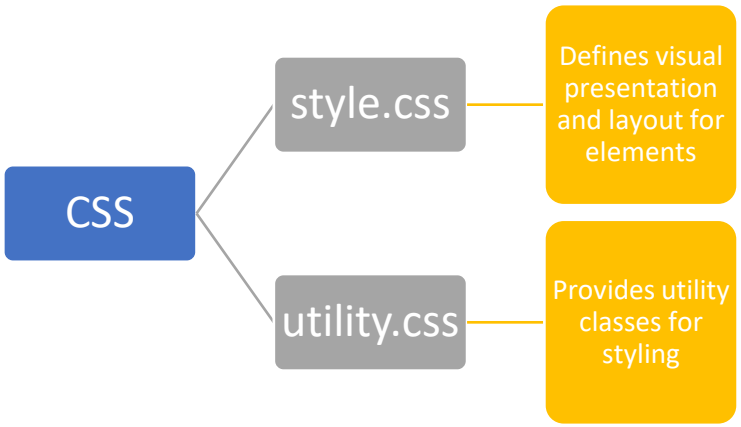
1. Initialize: Fetches the initial list of songs and plays the first one in pause mode.
2. Display albums: Calls the displayalbums() function.
3. Event listeners:
    - **Play/Pause**: Toggles between playing and pausing the current song.
    - **Time update:** Updates the song's current time and seek bar position as the song plays.
    - **Seek bar:** Allows users to seek within the song by clicking on the seek bar.
    - **Previous/Next:** Plays the previous or next song in the list.
    - **Volume control:** Adjusts the song volume based on user input.
    - **Album click:** Loads the song list for the selected album.
    - **Mute/Unmute:** Toggles between muting and unmuting the song.

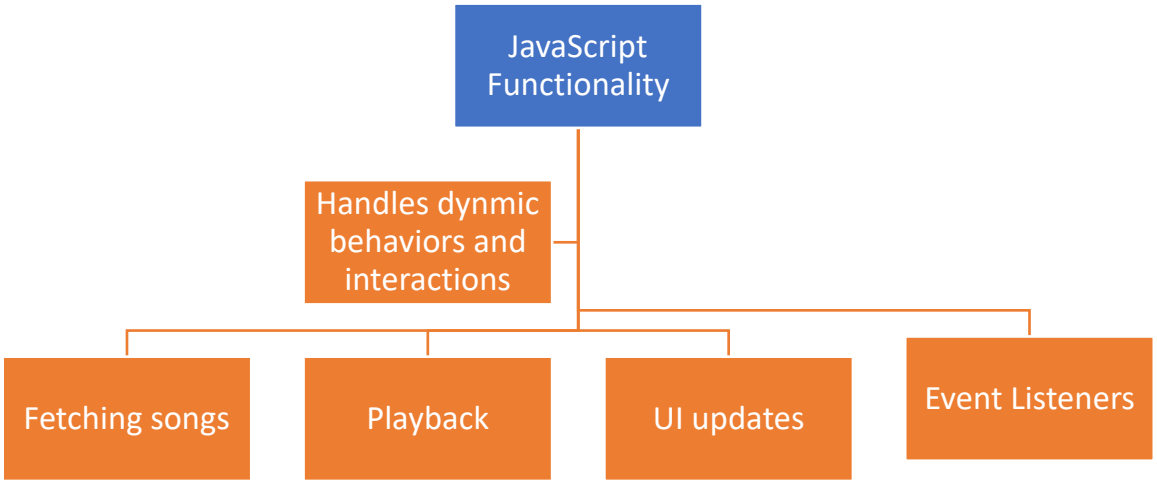
- **Execution**

  The main() function initializes the application by fetching the initial set of songs and setting up the UI and event listener

Diagram:


HTML Structure
Head
Meta
Link
Title
Body
Left
Right


CSS
style.css
Defines visual presentation and layout for elements
utility.css
Provides utility classes for styling


JavaScript Functionality
Handles dynmic behaviors and interactions
Fetching songs
Playback
UI updates
Event Listeners

System Coding

Index.html(Main page):-

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="style.css">
<link rel="stylesheet" href="utility.css">
 <link rel="icon" href="">
 <link rel="icon" href="spoify_icon.ico">
<title>Spotify - Web Player: Music for everyone</title>
</head>
<body>
   <div class="container flex bg-black">
      <div class="left">
         <div class="home bg-grey rounded m1 p1">
            <div class="logo"><img width="135" class="invert" src="logo.svg" alt=""></div>
            <ul>
               <li><img class="invert " src="home.svg" alt="home">Home</li>
               <li><img class="invert " src="search.svg" alt="search">Search</li>
            </ul>
         </div>
         <div class="library bg-grey rounded m1 p1">
            <div class="heading">
               <img class="invert heading" src="playlist.svg" alt="">
               <h2>
                  Your Library
               </h2>
            </div>
            <div class="songlist">
               <ul>
               </ul>
            </div>
            <div class="footer">
               <div>
                  <a href="https://www.spotify.com/in-en/legal/" id=""><span>Legal</span></a>
               </div>
               <div>
                  <a href="https://www.spotify.com/in-en/safetyandprivacy/" id=""><span>Safety &amp;amp; Privacy
                        Center</span></a>
               </div>
               <div>
                  <a href="https://www.spotify.com/in-en/legal/cookies-policy/" id=""><span>Cookies</span></a>
               </div>
               <div>
                  <a href="https://www.spotify.com/in-en/legal/privacy-policy/#s3" id=""><span>About
                        Ads</span></a>
               </div>
               <div>
                  <a href="https://www.spotify.com/in-en/accessibility/" id=""><span>Accessibility</span></a>
               </div>
            </div>
         </div>
      </div>
      <div class="right bg-grey  p1 ">
         <div class="header">
            <div class="nav">
               <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24" color="white"
```

```
            fill="none">
            <path d="M15 6C15 6 9.00001 10.4189 9 12C8.99999 13.5812 15 18 15 18" stroke="currentColor"
              stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round" />
          </svg>
          <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24" color="white"
            fill="none">
            <path d="M9.00005 6C9.00005 6 15 10.4189 15 12C15 13.5812 9 18 9 18" stroke="currentColor"
              stroke-width="1.5" stroke-linecap="round" stroke-linejoin="round" />
          </svg>
        </div>
        <div class="button">
          <button class="signupbtn">Sign up</button>
          <button class="loginbtn">Log in</button>
        </div>
      </div>
      <div class="spotify-playlist">
        <h1>Spotify-Playlist</h1>
        <div class="cardcontainer">
          <div data-folder="cs" class="card ">
            <div class="play">
              <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
                <circle cx="30" cy="30" r="25" fill="#00FF00" />
                <polygon points="22,17 22,43 38,30" fill="black" />
              </svg>
            </div>
            <img src="/image/Alan_Walker.jpg" alt="">
            <h2>Alan Walker songs</h2>
            <p>Alan walker all mix songs</p>

          </div>
          <div data-folder="ncs" class="card">
            <div class="play">
              <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
                <circle cx="30" cy="30" r="25" fill="#00FF00" />
                <polygon points="22,17 22,43 38,30" fill="black" />
              </svg>
            </div>
            <img src="/image/happyhits.jpg" alt="">
            <h2>Happy Hits</h2>
            <p>Hits to boost your mood and fill you with happiness</p>

          </div>
          <div data-folder="stranger" class="card">
            <div class="play">
              <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
                <circle cx="30" cy="30" r="25" fill="#00FF00" />
                <polygon points="22,17 22,43 38,30" fill="black" />
              </svg>
            </div>
            <img src="/image/stranger.jpg" alt="">
            <h2>Stranger Things </h2>
            <p>The stranger things webseries all songs</p>

          </div>
          <div data-folder="alan" class="card">
            <div class="play">
              <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
                <circle cx="30" cy="30" r="25" fill="#00FF00" />
                <polygon points="22,17 22,43 38,30" fill="black" />
              </svg>
```

```
    </div>
    <img src="/image/hanumanji.jpg" alt="">
    <h2>hanumanji</h2>
    <p>Hanuman chalisa and more</p>

</div>
<div data-folder="moving" class="card">
    <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
            <circle cx="30" cy="30" r="25" fill="#00FF00" />
            <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
    </div>
    <img src="/image/moving.jpg" alt="">
    <h2>Getting Peace</h2>
    <p>Give You peace and Calmness Playlist</p>
</div>
<div data-folder="rain" class="card">
    <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
            <circle cx="30" cy="30" r="25" fill="#00FF00" />
            <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
    </div>
    <img src="/image/nightrain.jpg" alt="">
    <h2>Night Rain</h2>
    <p>This playlist feel you good</p>
</div>
<div data-folder="rapsong" class="card">
    <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
            <circle cx="30" cy="30" r="25" fill="#00FF00" />
            <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
    </div>
    <img src="/image/rapsong.jpg" alt="">
    <h2>Rap Song</h2>
    <p>Includes good and aggressive rap song</p>
</div>
<div data-folder="marron5" class="card">
    <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
            <circle cx="30" cy="30" r="25" fill="#00FF00" />
            <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
    </div>
    <img src="/image/marron5.jpg" alt="">
    <h2>marron 5</h2>
    <p>marron 5 hit songs playlist makes you agressive and cool</p>
</div>
<div data-folder="justin" class="card">
    <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
            <circle cx="30" cy="30" r="25" fill="#00FF00" />
            <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
    </div>
    <img src="/image/justin.png" alt="">
    <h2>marron 5</h2>
    <p>justin biber hit songs</p>
```

```
    </div>
    <div data-folder="music" class="card">
      <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
          <circle cx="30" cy="30" r="25" fill="#00FF00" />
          <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
      </div>
      <img src="/image/music.png" alt="">
      <h2>Alok Songs</h2>
      <p>Alok New Songs</p>
    </div>
    <div data-folder="foryou" class="card">
      <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
          <circle cx="30" cy="30" r="25" fill="#00FF00" />
          <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
      </div>
      <img src="/image/foryou.jpeg" alt="">
      <h2>For you</h2>
      <p>music for you</p>
    </div>
    <div data-folder="morning" class="card">
      <div class="play">
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 60 60" width="48" height="48">
          <circle cx="30" cy="30" r="25" fill="#00FF00" />
          <polygon points="22,17 22,43 38,30" fill="black" />
        </svg>
      </div>
      <img src="/image/morning.jpeg" alt="">
      <h2>Morning</h2>
      <p>Morning musics</p>
    </div>
  </div>
  <div class="playbar">
    <div class="seekbar">
      <div class="circle">
      </div>
    </div>
    <div class="abovebar">
      <div class="songinfo">
      </div>
      <div class="songbuttons">
        <img id="previous" src="prevsong.svg" alt="">
        <img id="play" src="play.svg" alt="">
        <img id="next" src="nextsong.svg" alt="">
      </div>
      <div class="timevol"></div>
      <div class="set">
        <div class="songtime"></div>
      </div>
      <div class="volume">
        <img src="volume.svg" alt="">
        <div class="range">
          <input type="range" name="volume" id="">
        </div>
      </div>
    </div>
  </div>
```

```
        </div>
      </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Style.css:-

```
@import
url('https://fonts.googleapis.com/css2?family=Roboto:it
al,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,
300;1,400;1,500;1,700;1,900&display=swap');

:root {
    --a: 0;
}

* {
    margin: 0px;
    padding: 0px;
    font-family: "Roboto", sans-serif;
}
body {
    background-color: black;
    color: white;
}
.left {
    width: 25vw;
    padding: 10px;
}
.right {
    width: 75vw;
}
.home {
  padding: 10px;
}
.home ul li {
    display: flex;
    gap: 25px;
    width: 25px;
    list-style: none;
    padding-top: 25px;
    font-weight: bold;
}
.heading {
    display: flex;
    gap: 15px;
    width: 100%;
    padding-top: 14px;
    padding: 23px 14px;
    font-weight: bold;
    align-items: center;
    font-size: 13px;
}
.heading img {
    width: 30px;
}
.library {
    min-height: 80vh;
    position: relative;
}
.footer {
    display: flex;
    font-size: 10px;
    color: grey;
    list-style: none;
    gap: 13px;
    position: absolute;
    bottom: 0;
    padding: 10px 0;
}
.footer a {
    color: grey;
}
.right {
    margin: 16px 0;
    position: relative
}
.header {
    display: flex;
    justify-content: space-between;
    background-color: rgb(34, 34, 34);
}
.header>* {
    padding: 20px;
}
.spotify-playlist {
    padding: 16px;
}
.cardcontainer {
    margin: 30px;
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
    overflow-y: scroll;
    max-height: 70vh;
}
.card {
    position: relative;
    width: 200px;
    padding: 10px;
    border-radius: 5px;
    background-color: #252525;
    transition: all 0.9s;
}
.card:hover {
    background-color: black;
    cursor: pointer;
    --a: 1;
}
.card>* {
    padding-top: 10px;
}
.card img {
    width: 100%;
    object-fit: contain;
}
.play {
```

```
    width: 28px;
    height: 28px;
    background-color: #00FF00;
    border-radius: 50%;
    border: 2px solid black;
    padding: 4px;
    display: flex;
    align-items: center;
    justify-content: center;
    position: absolute;
    top: 168px;
    right: 17px;
    opacity: var(--a);
    transition: all 1s ease-out;
}
.button>* {
    margin: 0 12px;
}
.signupbtn {
    background-color: rgb(34, 34, 34);
    ;
    color: grey;
    font-weight: bold;
    cursor: pointer;
    border: none;
    outline: none;
    font-size: 16px;
}
.signupbtn:hover {
    font-size: 17px;
    color: white;
}
.loginbtn {
    background-color: white;
    border-radius: 21px;
    color: black;
    font-weight: bold;
    padding: 10px;
    width: 79px;
    cursor: pointer;
    font-size: 16px;
}
.loginbtn:hover {
    font-weight: bold;
    font-size: 17px;
}
.playbar {
    position: fixed;
    bottom: 30px;
    filter: invert(1);
    background-color: #dad5d5;
    border-radius: 10px;
    height: 60px;
    width: 70%;
    padding: 12px;
}
.songbuttons {
    display: flex;
    justify-content: center;
    gap: 12px;
    height: 40px;
}
.songlist {
    height: 544px;
    overflow: auto;
    margin-bottom: 44px;
}
.songlist ul {
    padding: 0 10px;
    font-size: 15px;
}

.songlist ul li {
    list-style-type: decimal;
    display: flex;
    justify-content: space-between;
    gap: 12px;
    cursor: pointer;
    padding: 12px 0;
    border: 1px solid white;
    margin: 12px 0;
    padding: 10px;
    border-radius: 5px;
    align-items: center;
    /* width:290px; */
}
.songlist .info {
    font-size: 13px;
    width: 344px;
}
.playnow {
    display: flex;
}
.playnow span {
    font-size: 15px;
    width: 64px;
    padding: 12px;
    justify-content: center;
    align-items: center;
}
.seekbar {
    height: 4px;
    width: 98%;
    background-color: black;
    position: absolute;
    bottom: 0;
    border-radius: 10px;
    margin: 6px;
    cursor: pointer;
}
.circle {
    width: 13px;
    height: 13px;
    border-radius: 13px;
    background-color: black;
    position: relative;
    bottom: 6px;
    left: 0%;
    transition: left 0.5s;
}
```

```
.songbuttons img {
    cursor: pointer;
}
.timevol {
    display: flex;
    justify-content: space-evenly;
    align-items: center;
}
.songinfo {
    word-wrap: break-word;
    width: 500px;
    height: 29px;
}
.songinfo {
    color: black;
    padding: 0 12px;
    width: 380px;
    overflow: hidden;
    height: 58px;
}
.songtime {
    width: 125px;
    padding: 0 12px;
    color: black;
}
.abovebar {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
.info div {
    word-break: break-all;
    overflow: auto;
    height: auto;
}
.volume {
    display: flex;
    align-items: center;
    gap: 12px;
    cursor:pointer;
}
.volume input {
    cursor: pointer;
}
```

Utility.css:-

```
.border{
    border :2px solid red;
    margin:3px;
}
.flex{
    display:flex;
}
.justify-center{
    justify-content: center;
}
.items-center {
    align-items: center;
}
.bg-black{
    background-color: black;
    color:white;
}
.invert{
    filter:invert(1);
}
.size{
    width:25px;
}
.bg-grey{
    background-color: #121212
}
.rounded{
    border-radius:7px;
}
.m1{
    margin:5px;
}
.p1{
    padding:10px
}
::-webkit-scrollbar {
    width: 10px;
  }
  ::-webkit-scrollbar-track {
    background: #333;
  }
  /* Handle */
  ::-webkit-scrollbar-thumb {
    background: #666;
    border-radius: 5px;
  }
  ::-webkit-scrollbar-thumb:hover {
    background: #888;
  }
```

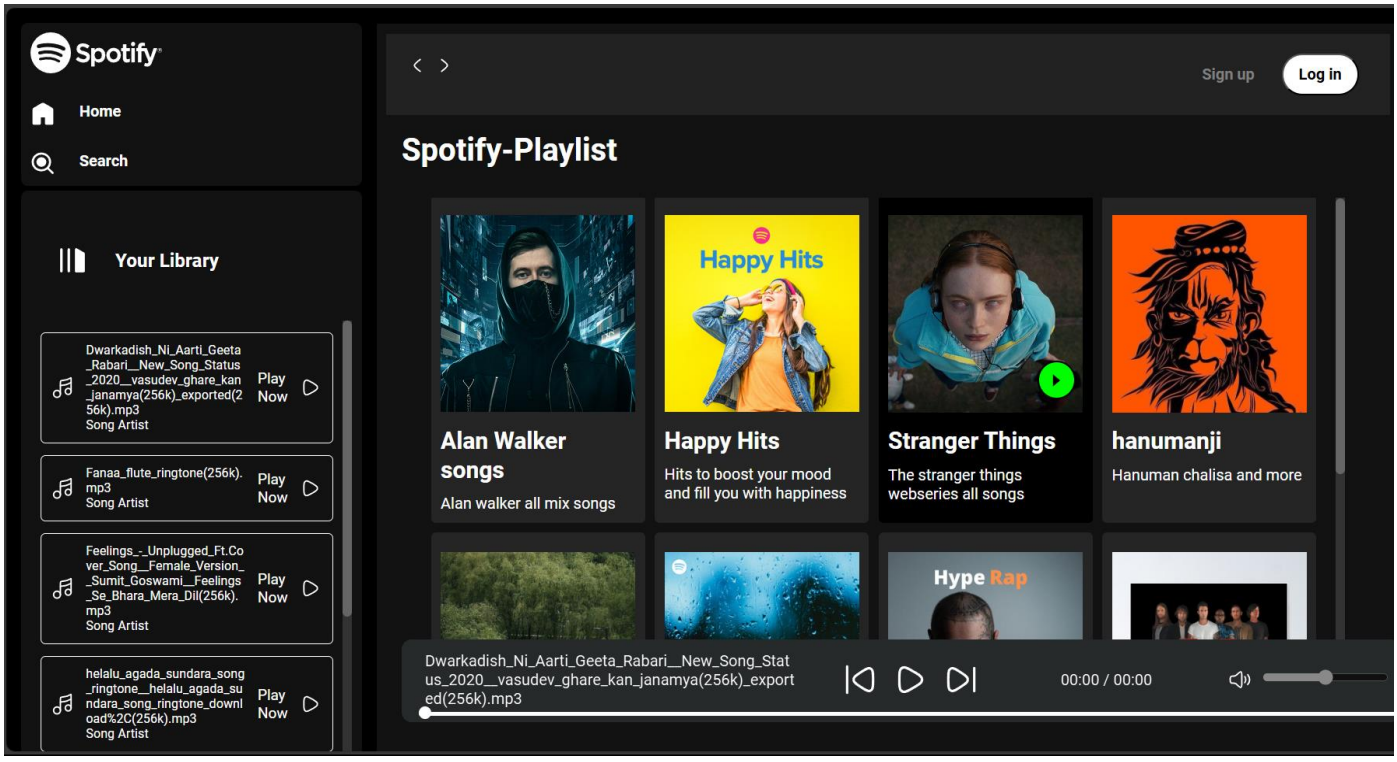
Script.js:-

```
let currentSong = new Audio();
let songs;
let currfolder;

function minutes(seconds) {
    if (isNaN(seconds) || seconds < 0) {
        return "00:00";
    }
    const minutes = Math.floor(seconds / 60);
    const remainingSeconds = Math.floor(seconds %
60);

    // Format minutes and seconds with leading zeros if
necessary
    const minutesString = String(minutes).padStart(2,
'0');
    const secondsString =
String(remainingSeconds).padStart(2, '0');

    return ${minutesString}.${secondsString};
}

async function getSongs(folder) {
```

```
    currfolder = folder;
    let a = await fetch(http://127.0.0.1:5500/${folder}/)
    let response = await a.text();
    let div = document.createElement("div")
    div.innerHTML = response;
    let as = div.getElementsByTagName("a")
    songs = []
    for (let index = 0; index < as.length; index++) {
        const element = as[index];
        if (element.href.endsWith(".mp3")) {
            songs.push(element.href.split(/${folder}/)[1])
        }
    }


    // show all the song in the playlist
    let songUL =
document.querySelector(".songlist").getElementsByTa
gName("ul")[0]
    songUL.innerHTML = ""
    for (const song of songs) {
        songUL.innerHTML = songUL.innerHTML +
`<li>

                        <img class="invert" src="music.svg"
alt="">
                        <div class="info">
                            <div>${song.replaceAll("%20", "
")} </div>
                            <div>Song Artist</div>
                        </div>
                            <span>Play Now</span>
                            <img class="invert"
src="play.svg" alt=""> </li>`;
    }

    // Attach an evnt listner to each song

Array.from(document.querySelector(".songlist").getEle
mentsByTagName("li")).forEach(e => {
        e.addEventListener("click", element => {

console.log(e.querySelector(".info").firstElementChild.
innerHTML)

playMusic(e.querySelector(".info").firstElementChild.i
nnerHTML)
        })
    })

}

const playMusic = (track, pause = false) => {
    currentSong.src = /${currfolder}/ + track
    if (!pause) {
        currentSong.play()
        play.src = "pause.svg";
    }
    document.querySelector(".songinfo").innerHTML =
decodeURI(track)
    document.querySelector(".songtime").innerHTML =
"00:00 / 00:00"
}

async function displayalbums() {
    let a = await fetch(http://127.0.0.1:5500/songs/)
    let response = await a.text();
    let div = document.createElement("div")
    div.innerHTML = response;
    let anchors = div.getElementsByTagName("a")
    let cardcontainer =
document.querySelector(".cardcontainer")
    Array.from(anchors).forEach(async e => {
        if (e.href.includes("/songs")) {
            let folder = e.href.split("/").slice(-2)[0]
        }
    })
}
async function main() {
    //get the list of all the songs
    await getSongs("songs/ncs")
    playMusic(songs[0], true)

    //Display all the albums on the page
    displayalbums()

    //Attach an event listener to play , next and previous
    play.addEventListener("click", () => {
        if (currentSong.paused) {
            currentSong.play()
            play.src = "pause.svg"
        } else {
            currentSong.pause()
            play.src = "play.svg"
        }
    })

    //Listen for timeupdate event
    currentSong.addEventListener("timeupdate", () => {
        document.querySelector(".songtime").innerHTML
=
${minutes(currentSong.currentTime)}:${minutes(curre
ntSong.duration)}
        document.querySelector(".circle").style.left =
(currentSong.currentTime / currentSong.duration) *
100 + "%";
    })

    //Add an event listener to seekbar

document.querySelector(".seekbar").addEventListener(
"click", e => {
        let percent = (e.offsetX /
e.target.getBoundingClientRect().width) * 100;
        document.querySelector(".circle").style.left =
+percent + "%";
        currentSong.currentTime =
((currentSong.duration) * percent) / 100
    })
```

```
    // Add an evnt listener to previous and next

    previous.addEventListener("click", () => {
        console.log("Previous clicked")
        let index =
songs.indexOf(currentSong.src.split("/").slice(-1)[0])
        if ((index - 1) >= 0) {
            playMusic(songs[index - 1])

        }
    })

    next.addEventListener("click", () => {
        console.log("Next clicked")

        let index =
songs.indexOf(currentSong.src.split("/").slice(-1)[0])
        if ((index + 1) < songs.length) {
            playMusic(songs[index + 1])
        }

    })

    // Add an event to volume

document.querySelector(".range").getElementsByTagN
ame("input")[0].addEventListener("change", (e) => {
        console.log("Setting volume to", e.target.value,
"/100");
        currentSong.volume = (e.target.value) / 100;
    })

    // Load the playlist whenever card is clicked

Array.from(document.getElementsByClassName("card
")).forEach(e => {
        e.addEventListener("click", async item => {
            songs = await
getSongs(songs/${item.currentTarget.dataset.folder})
        })
    })

    //Add evnt listner to mute the track
    document.querySelector(".volume
img").addEventListener("click", e => {
        console.log(e.target)
        if (e.target.src.includes("volume.svg")) {
            e.target.src = e.target.src.replace("volume.svg",
"mute.svg")
            currentSong.volume = 0;

document.querySelector(".range").getElementsByTagN
ame("input")[0].value = 0;
        } else {
            e.target.src = e.target.src.replace("mute.svg",
"volume.svg")
            currentSong.volume = .10;

document.querySelector(".range").getElementsByTagN
ame("input")[0].value = 10;
        }
    })
}
main()
```

Result



About Spotify

- A Spotify clone app is a music streaming app which provide a free plateform to listen music without downloading it.
- The software is user-friendly from login to listening music. With the various features, users can enjoy listening to their favourite songs with ease.
- Some music player only avalible in online .so spotify clone provide the offline plateform for users . Users will use the offline option to stream the album and listen without using the internet.
- It is a freemium app also, some basic features are with adversting and limited control.

Spotify is music player app. It has 433 million monthly active users. The app is good but user has so many complained and issue about this app. Like too many Ads, low sound quality, limits of downloading songs, promotion of podcasts over music and data usage etc. These are some drawbacks of this platform. So we decide to make same music streaming platform like spotify and resolve the user’s problems and also add some additional feature in our web application.