* make body background black and font color white
* copy svg of spotify icon in the "logo.svg" , we can use same svg for our web page. (if not running then check online for svg spec and there you will find "xmlmx" , that need to add)
* link css style
* make container and inside that make 2 div ----left and right (because spotify has to portion )
* update in css , make left div 25% and right 75% ( note use vw and vh for responsive webpage)
* reset all margin and padding of page.
* Make utility css file where we can store default styling , flex , red border, align center , justify center
* Add flex class to container to make inline
* Now construct left side bar :- now make 2 div , home and library

Inside home :- make div for logo and make unordered list and inside that add 2 list element -> home logo and serach logo

Save their svg file with same name

(how to capture svg files ? goto to inspect mode and then select that icon then copy svg into our vs code)

* In css adjust size(width) of home and search icon and remove list dots and make inline by inserting flex on it. Add gap also between icon and text.
* Add margin to left container to maintain some gaps from boundary.
* Make new css in utility “bg-grey” to make background black, and this add it to home and library class.
* Make “bg-black” add this to container.
* Invert image :- make class in css in name of invert ,

**Syntax :- filter : invert(1)**

* Now add this invert class to images
* Remove red border now, we had kept it only because we wanted to visualize the things
* Create new class to round the boarder ( in utlity)
* Create margin class to give margin between boxes or text (time stamp 22:00)
* Inside home give padding and add padding between icons(to adjust it)
* Note -> whichever padding / margin using , give specific name to them because we need to maintain uniformity , (example :- home and library will get same type of design )
* Extra -> download what font extension , which will tell which font we are using ,

Then download that font from google font. If not available for download then we can take any similar font .

Import that style and add into css file.

Add this font-family in required text

* Now , we need to modify library div
* Save svg file of library icon, and invert it, and adjust its width.
* Make div in name of Footer in library , copy its elements directly from spotify website
* Remove class in copied elements by using chat gpt
* Adjust font , color grey , make gap ….. to shift it below , make library height to 50vh then make library position relative and footer absolute and bottom :0
* Because of link its color does not changes , therefore add color to its “a” attribute .
* If font same font used in all classes then you can add in body and remove it from other places
* Note :- to cover whole page I have used , in parent flex-direction column and in child we can give flex :1 to acquire

Now on right side :-

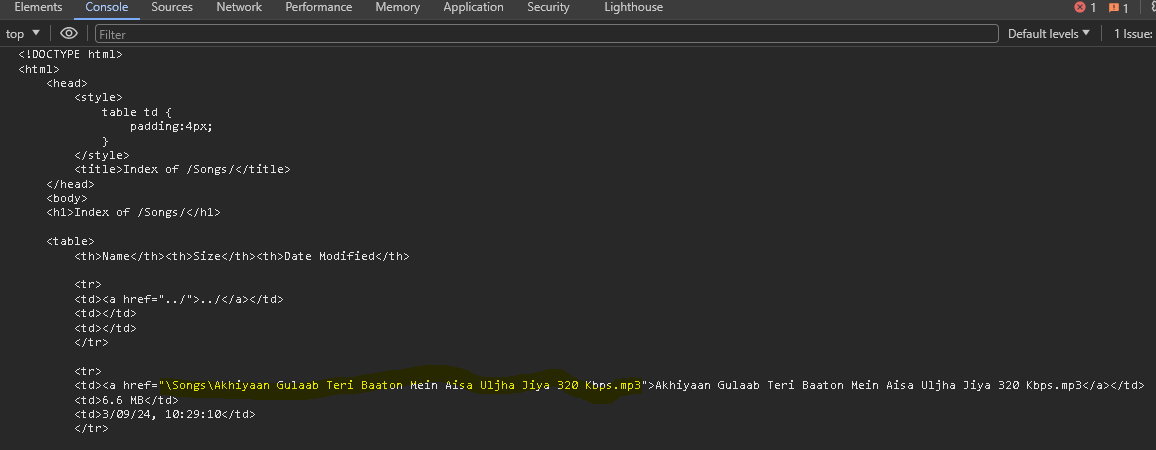
* Add header (add div with class header) , inside this we have to add “< >” and log in and signup
* Inside header create 2 div ;- one for nav bar and second buttons( signup and login)
* Use hugeicons for ready made icons, take <> this icon from there
* Some times we can change svg color also, look into svg
* Give padding to both div
* Adjust margin of right box
* Give header background color
* Timestamp :- 50:39
* Make another div for spotify playlist
* Now make cards div:- img,h2,paragraph
* Take img from there link , set their width and give object -fit: contain

(hint .card img{ width: 100% object-fir: contain;}

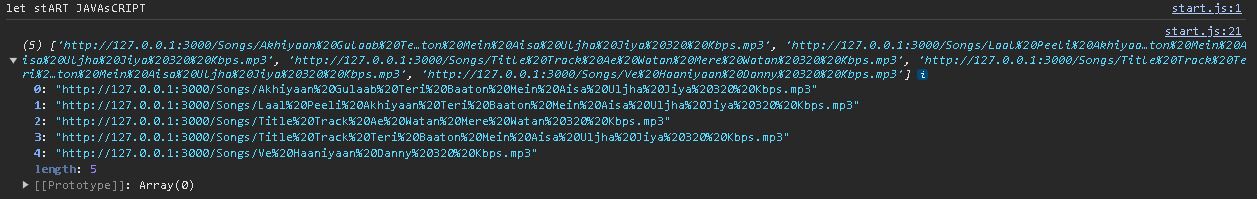
* Give padding , radius ,color inside card
* In card container do flex-wrap to wrap it
* Now Inside that image while hovering it show green button :- so to construct that take image from hugeicons and we can ask gpt to convert into round green
* For positioning use relative and absolute and left top position
* Use opacity and transition to give hover affect

Hint :- opacity :0

Transition all 1s ease-out

* Copy paste cards add scroll ( overflow: scroll) , ask gpt to make color of scroll to black
* Now set signup button
* Add hover effect also, add paddings
* Connect java script
* Create playbar div and stick it to below (hint make right div relative and playbar absolute , bottom zero)
* Make bg green , width 100% , give padding
* Add songs button , play , previous , forward
* Make it centred
* Make another div , song info and songtime
* Download songs in folder,
* Now our aim is, on left side library we need to maintain all songs.
* Make unordered list -> make list of songs “songlist”
* Time stamp 1:28 java starts
* Javascript starts
* 
* After running above code we get output as
* 
* We need to extract href files location
* async *function* main() {
* *let* a = await fetch("http://127.0.0.1:3000/Songs/");
* *let* response = await a.text();
* //why we creating element , why can not use response variable directly ?
* // becuase response is of string type and div is object type , so from object we can extract info easily
* *let* div = document.createElement("div");
* div.innerHTML = response;
* *let* as = div.getElementsByTagName("a");
* *let* songs = []; // songs will be stored here
* for (*let* index = 0; index < as.length; index++) {
* *const* element = as[index];
* if (element.href.endsWith(".mp3")) {
* //selecting songs link which ends with mp3
* songs.push(element.href);
* }
* }
* console.log(songs);
* }
* main();

output



Now we need to play audio , so take help of google how to play songs.

 document.querySelector(".play").addEventListener("click", () *=>* {

*var* audio = new *Audio*(songs[0]);

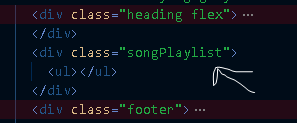
    audio.play();

  });

Now all that extracted songs need to add in web page ,

First create another div inside library “songPlayList”

So inside this with help of javaScript we will push list of all songs <li></li>

html

* Print "let start JavaScript" to the console.

Define an asynchronous function named getSongs:

* Make an asynchronous request to fetch data from the URL "http://127.0.0.1:3000/Songs/".
* Convert the response to text format.
* Create a new div element.
* Set the innerHTML property of the div to the fetched response.
* Get all <a> elements from the div.
* Initialize an empty array named songs.
* Loop through each <a> element:
* If the href attribute of the element ends with ".mp3":
* Push the part of the href after "/Songs/" and before "mp3" into the songs array.
* Return the songs array.

Define an asynchronous function named main:

* Call the getSongs function and await its result, storing it in the variable songs.
* Print the songs array to the console.
* Get the first <ul> element within the element with the class "songPlaylist".
* Loop through each song in the songs array:
* Append an <li> element to the <ul> with the following content:
* An <img> element with class "invert icon-width" and source "./images/all/music.svg".
* A <div> element with class "info" containing:
* A <div> element with the song name, replacing "%20" with spaces.
* Another <div> element with the text "Mukul".
* A <div> element with class "playNow" containing:
* A <span> element with the text "Play Now".
* An <img> element with class "roundPlay invert icon-width" and source "./images/all/play2.svg".
* Add an event listener to the element with class "play":
* When clicked, create a new Audio object with the first song from the songs array and play it.
* Add an event listener to the audio object:
* When the "loadeddata" event is triggered, log the duration of the audio in seconds to the console.

async *function* getSongs() {

*let* a = await fetch("http://127.0.0.1:3000/Songs/");

*let* response = await a.text();

  //why we creating element , why can not use response variable directly ?

  // becuase response is of string type and div is object type , so from object we can extract info easily

*let* div = document.createElement("div");

  div.innerHTML = response;

*let* as = div.getElementsByTagName("a");

*let* songs = []; // songs will be stored here

  for (*let* index = 0; index < as.length; index++) {

*const* element = as[index];

    if (element.href.endsWith(".mp3")) {

      //selecting songs link which ends with mp3

      songs.push(element.href.split("/Songs/")[1].split("mp3")[0]);

    }

  }

  return songs;

}

async *function* main() {

  //get list of songs

*let* songs = await getSongs();

  console.log(songs);

*let* songUL = document

    .querySelector(".songPlaylist")

    .getElementsByTagName("ul")[0];

  for (*const* song of songs) {

    songUL.innerHTML =

      songUL.innerHTML + //here ` backtick is used , just below esc button

      `<li>

    <img

      class="invert icon-width"

      src="./images/all/music.svg"

      alt=""

    />

    <div class="info">

      <div>${song.replace(/%20/g, " ")}</div>

      <div>Mukul</div>

    </div>

    <div class="playNow">

      <span>Play Now</span>

      <img

        class="roundPlay invert icon-width"

        src="./images/all/play2.svg"

        alt=""

      />

    </div>

  </li>`;

  }

  //play songs

  document.querySelector(".play").addEventListener("click", () *=>* {

*var* audio = new *Audio*(songs[0]);

    audio.play();

  });

  audio.addEventListener("loadeddata", () *=>* {

*let* duration = audio.duration;

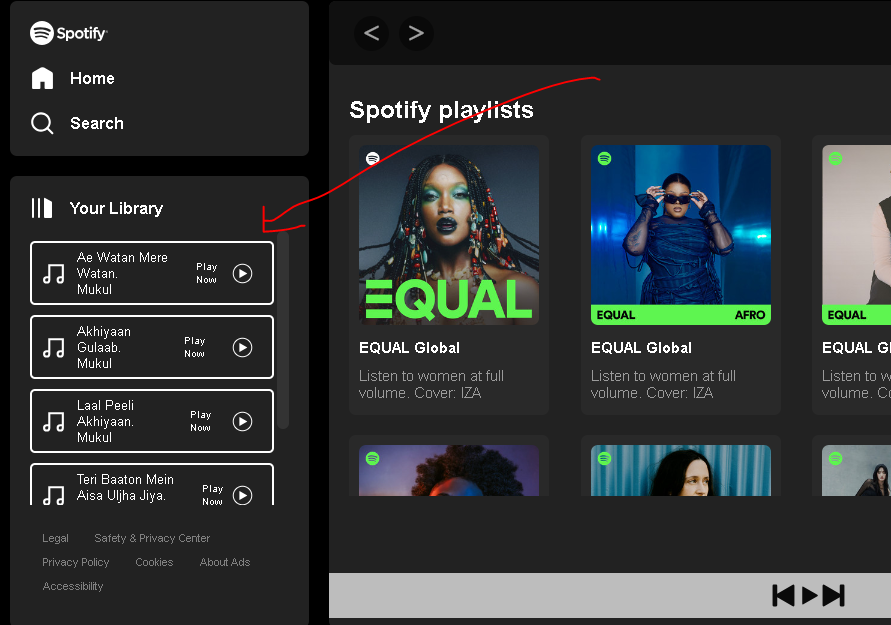
    console.log(duration);

    //duration variable holds duration in seconds

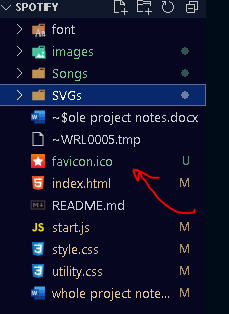
  });

}

main();



Now whenever we click on play button song will show to playbar (name and duration)

Additional tip favicon.ico file if you put into your folder then tab will look like same icon



Timestamp : 2:11

* Make seek bar in playbar
* Now whenever we click on songs present inside library it should play
* So for that we add event listener ,so it capture name of song clicked and will pass on to play music function
* Note- we have declare currentSong variable globally if we declare locally then whenever we click song it will create multiple currentsong variable for each song , so if we declare globally then it would update on that global variable only.
* *let* currentSong = new Audio();
* //attach an event listner to each song
* *Array*.from(
* document.querySelector(".songPlaylist").getElementsByTagName("li")
* ).forEach((*e*) *=>* {
* *e*.addEventListener("click", (*element*) *=>* {
* console.log(*e*.querySelector(".info").firstElementChild.innerHTML);
* playMusic(*e*.querySelector(".info").firstElementChild.innerHTML);
* });
* });

*const* playMusic = (*track*) *=>* {

  currentSong.src = "/Songs/" + *track* + ".mp3";

  currentSong.play();

};

* Now make play button cursor pointer.
* Capture click event on play button ,so when song is paused it will start play button if not then pause , along with that need to change its svg also

*const* play = document.querySelector(".play > img");

  play.addEventListener("click", () *=>* {

    if (currentSong.paused) {

      currentSong.play();

      play.src = "./images/all/pause.svg";

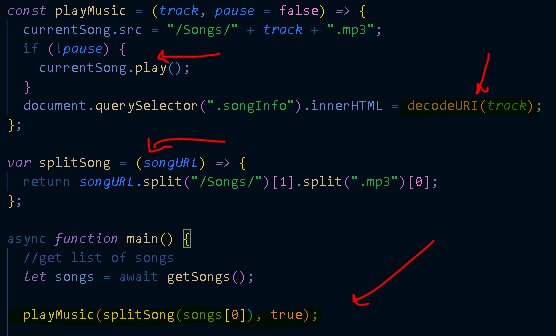
    } else {

      currentSong.pause();

      play.src = "./images/all/play2.svg";

    }

  });

* Timestamp 2:36
* now we need to add song info and time
* inside info inner html add track and duration
* now make function for time update which convert second to minute:second
* *function* formatTime(*seconds*) {
* // Calculate minutes and seconds
* *seconds* = Math.round(*seconds*);
* *let* minutes = Math.floor(*seconds* / 60);
* *let* remainingSeconds = *seconds* % 60;
* // Add leading zeros if necessary
* *let* formattedMinutes = minutes < 10 ? `0${minutes}` : `${minutes}`;
* *let* formattedSeconds =
* remainingSeconds < 10 ? `0${remainingSeconds}` : `${remainingSeconds}`;
* // Return the formatted time
* return `${formattedMinutes}:${formattedSeconds}`;
* }
* currentSong.addEventListener("timeupdate", () *=>* {
* console.log(currentSong.currentTime, currentSong.duration);
* document.querySelector(".songTime").innerHTML = `${formatTime(
* currentSong.currentTime
* )} / ${formatTime(currentSong.duration)}`;
* });
* Now whenever we load site , it should have atleat one song on load with pause condition.
* Therefore we add 1st song by default
* Add playMusic fxn in main which will run as soon it load
* 

Now that seek bar should move as song play.

So left percentage we change according completion of song

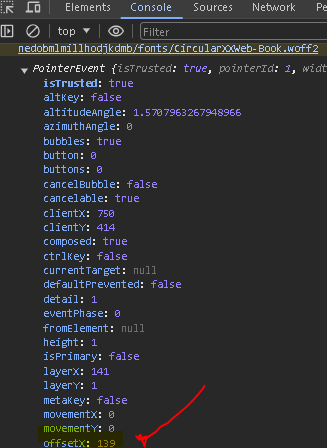
document.querySelector(".seekbar").addEventListener("click", (*e*) *=>* {

*let* percent = (*e*.offsetX / *e*.target.getBoundingClientRect().width) \* 100;

    currentSong.currentTime = (currentSong.duration \* percent) / 100;

    //document.querySelector(".circle").style.left = percent + "%";

  });

here if you console.log(e) then you will find out right side image, in that you will find offset value, that is value of x coordinate where you have

clicked on seek bar.

So for shifting circle to wherever we have clicked

Here we find full width of that seek bar with help

Of function getBoundingClientRect().width

We find percentage , how much its shifting then

Accordingly we shift circle and change current time

Also. Check above code

* Now we will write media query
* So in mobile size website , left div should not be visible and right should occupy all place
* Make left class absolute , left position -150% (so , it will not visible) and make it z-index =1 (it will come z axis 1 up)
* Add some transition
* Change width and height accordingly
* Same right div also change its width and height fully occupy

Add Hamburger , add it in nav bar . make display none in and visible in media query

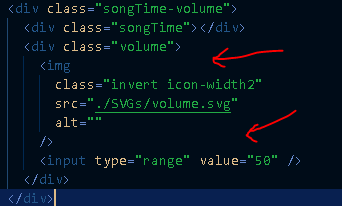
Add cross in left div , make it absolute and set left top position for cross

* Add eventListner for hamburger and for cross as well

Play bar change its width and seek bar as well

Note :- if you give z-index then even you transit with hamburger then also that div will float over the screen , example “playbar given z-index 1”

And if you want to change relative position according to screen make dimension in percentage

* Now we add eventListner to previous and next button
* document.querySelector(".previous").addEventListener("click", () *=>* {
* *let* index = songs.indexOf(currentSong.src);
* if (index - 1 >= 0) {
* playMusic(splitSong(songs[index - 1]));
* } else {
* alert("its first song");
* }
* });
* document.querySelector(".forward").addEventListener("click", () *=>* {
* *let* index = songs.indexOf(currentSong.src);
* if (index + 1 < songs.length) {
* play.src = "./images/all/pause.svg";
* playMusic(splitSong(songs[index + 1]));
* } else {
*       alert("its last song");
* }
* });
* Add volume icon, inside top bar and its functionality

Here we added input type is range so we can extract

Values out of it.

In javaScript , we have used “change” instead of “click” and set volume

* Mute directly while clicked on volume button

document

    .querySelector(".volume")

    .getElementsByTagName("input")[0]

    .addEventListener("change", (*e*) *=>* {

      currentSong.volume = parseInt(*e*.target.value) / 100;

    });

Timestamp :- 4:02

Taking songs from folders , 4:05

* Created folder ncs and cs (non copy rights, copyrights)
* Replaces all location where songs are loading with ‘Songs/{folder name}
* Inside box in html code we will add data-folder = “{folder name}”

Check below learning point for data-\*

Learning new concept:-

the **data-\*** attribute is a feature introduced in HTML5 that allows you to store custom data attributes directly on HTML elements. These attributes can be accessed and manipulated using JavaScript, providing a way to associate additional data with HTML elements without using non-standard attributes or altering the HTML structure.

The **data-\*** attributes consist of two parts:

1. **Prefix (data-)**: This is a fixed prefix that must precede the custom attribute name. It indicates that the attribute is a custom data attribute and should not interfere with standard HTML attributes.
2. **Custom Attribute Name**: This is the name you choose to define for your custom data attribute. It can contain any characters (except spaces), and you can define multiple custom attributes for an element.

Here's an example of how you can use **data-\*** attributes in HTML:

HTML code

<div id="myElement" data-user-id="123" data-username="john\_doe" data-active="true"></div>

In this example, the **<div>** element has three custom data attributes: **data-user-id**, **data-username**, and **data-active**. These attributes store additional information about the element (user ID, username, and active status).

You can access and manipulate **data-\*** attributes using JavaScript. For example, to access the value of a **data-\*** attribute, you can use the **getAttribute()** method or directly access the **dataset** property of the element:

javascriptCopy code

const element = document.getElementById('myElement');

const userId = element.getAttribute('data-user-id');

const userName = element.dataset.username;

const isActive = element.dataset.active === 'true'; // Convert string value to boolean

You can also modify the values of **data-\*** attributes using the **setAttribute()** method or by directly modifying the **dataset** property:

javascriptCopy code

element.setAttribute('data-active', 'false');

element.dataset.username = 'new\_username';

Overall, **data-\*** attributes provide a convenient way to store custom data associated with HTML elements, making it easier to manage and manipulate data within your JavaScript code.

* Below we are clicking on box card and loading songs in playlist
* Changes we have done is like , loading songs code we shifted to getSongs function

*Array*.from(document.getElementsByClassName("box")).forEach((*e*) *=>* {

*e*.addEventListener("click", async () *=>* {

      songs = await getSongs(`Songs/${*e*.dataset.folder}`);

    });

  });

Timestamp:- 4:20

* Make function in name of displayAlbums
* Fetch the folders name
* After that open json file inside that folder and extract title and description of song
* Inside card add more boxes inside which change name of box with help of folder name , contain image(cover)

In main

 await displayAlbum();

async *function* displayAlbum() {

*let* a = await fetch(`http://127.0.0.1:3000/songs/`);

*let* response = await a.text();

*let* div = document.createElement("div");

  div.innerHTML = response;

*let* anchors = div.getElementsByTagName("a");

*let* cardContainer = document.querySelector(".cards");

*let* array = *Array*.from(anchors);

  for (*let* index = 0; index < array.length; index++) {

*const* e = array[index];

    if (e.href.includes("/songs")) {

*let* folder = e.href.split("/").slice(-2)[0];

*let* a = await fetch(`http://127.0.0.1:3000/songs/${folder}/info.json`);

*let* response = await a.json();

      cardContainer.innerHTML =

        cardContainer.innerHTML +

        `<div data-folder="${folder}" class="box bder-rnd">

        <img class="bder-rnd" src="./Songs/${folder}/cover.jpg" alt="" />

        <h2>${response.title}</h2>

        <p>${response.description}</p>

        <div class="play-button">

          <img src="./SVGs/play.svg" alt="" />

        </div>

      </div>

      `;

    }

  }

}