SYNOPSIS REPORT

On

Sangeet: music streaming platform (Spotify Clone)

Submitted in partial fulfilment of the requirements of the degree of

BACHELOR OF BUSINESS ADMINISTRATION (COMPUTER APPLICATION)

Submitted by

Roll No.	**Name.**
70	Suraj Oswal

DEPARTMENT OF SCIENCE AND TECHNOLOGY
ARIHANT COLLEGE OF ARTS, COMMERCE AND SCIENCE, CAMP, PUNE
FEB 2025

Arahant Education Foundation's

Arihant College Of Arts, Commerce & Science (ACACS) – Pune

Near Pulgate Bus Stand, Solapur Bazaar Road, Camp Pune – 01, 020-26332921 / 65213727

CERTIFICATE

This is to certify that the Project Report entitled **Sangeet: music streaming platform (Spotify Clone)** prepared by **Suraj Uttamchand Oswal** students of Bachelor of Business Administration (Computer Application) semester-IV course at Arihant College of Art's, Commerce and Science (ACACS), Pune-01.

To the best of our knowledge, this is original study done by the said student and important sources used by them have been duly acknowledged in this report.

The report is submitted in partial fulfillment Bachelor of Computer Science semester-III syllabus as per rule prescribed guideline of Savitribai Phule Pune University.

(Mrs. Pranali Deshmukh) Project Guide	(Mrs. Devyani Patil) HOD	(Dr. Kanchan Shinde) principle

Internal Examiner

External Examiner

❖ Index:

❖ Name of Project:

Sangeet**:** music streaming platform (Spotify clone).

❖ Project Partner Name and Roll No:

Roll No.	Name.
70	Suraj Oswal

❖ Introduction

The **Spotify Clone** is a simple web-based music player inspired by **Spotify**. It allows users to **play, pause, skip songs, adjust volume, and see the progress of the song** in a user-friendly way. The design looks similar to Spotify, giving a familiar and smooth experience.

This project mainly focuses on **front-end development**, meaning it is built using **HTML for structure, CSS for styling, and JavaScript for interactive features**. Users can explore **predefined playlists and songs**, but they cannot upload their own music or create accounts. It is meant for learning and practice rather than being a full replacement for Spotify.

By working on this project, I will get hands-on experience in:

- **HTML** – Creating the layout of the webpage.
- **CSS** – Styling the website to look attractive and easy to use.
- **JavaScript** – Making the music player work (like play, pause, and volume control).

Although this project is basic, it can be improved in the future by adding features like **user login, personalized playlists, and an online music library**. The **Spotify Clone** is a great way to understand how real-world music streaming websites work while practicing web development skills.

❖ Existing System

Before starting the development of this project, I explored the existing music streaming platforms available in the market, with a primary focus on **Spotify**, which is one of the most popular and widely used services for music lovers. Spotify offers a vast collection of songs, albums, and playlists, with a sleek and responsive user interface that works well on both desktop and mobile devices.

However, from a student and developer point of view, I was always curious to understand how such a platform works behind the scenes. I spent time observing the features, user interface design, and overall structure of the existing system. I noticed that Spotify uses modern web technologies and a smooth user experience with dynamic content loading, responsive design, and real-time audio playback control. It also includes features such as search functionality, playlist management, recently played items, and personalized suggestions.

While these systems are highly efficient and advanced, they are also built with complex back-end logic and massive databases. As a student, I wanted to take inspiration from such platforms and create a **simplified version** using only front-end technologies like **HTML, CSS, and JavaScript**. I focused mainly on replicating the basic user interface, navigation behavior, and song listing from albums stored in folders.

I was waiting to explore how user interactions like **menu toggling**, **album selection**, **song listing**, and **playback UI behavior** could be recreated using JavaScript logic. This exploration of the existing system gave me a solid foundation and reference point to build my own version from scratch, while keeping the project scope realistic and achievable within the limitations of a student-level web project.

❖ Proposed System

As a student with a passion for web development and music, I was always excited about the idea of creating my own music streaming platform—something simple yet functional, inspired by the interface and feel of popular apps like Spotify. This led me to propose a **Spotify Clone Web Project**, where the goal is to design and build a basic music streaming website using **HTML, CSS, and JavaScript**.

The proposed system aims to provide a user-friendly interface that allows users to interact with music albums, view songs, and simulate playback control (play, pause, seek, and volume). Since this is a student project, the focus is primarily on the **front-end functionality and design**, rather than complex server-side features like login, playlist saving, or streaming from a database.

In this system, albums are stored in folders, and songs are dynamically loaded based on the user's selection. The interface includes a **toggleable menu**, **search bar interaction**, a **responsive logo**, and a **playbar** with volume control. All these features are meant to mimic the modern feel of real-world music apps but implemented in a simpler, more understandable way.

I was waiting to implement this system not only to improve my coding skills but also to understand how real-time interactivity and dynamic content loading can be handled using JavaScript. This project gives me a chance to apply what I’ve learned in HTML, CSS, and JavaScript in a creative and practical way, and to build something that both looks good and functions smoothly.

The proposed system is ideal for showcasing my front-end development abilities, and it also opens the door to future enhancements—such as connecting to a backend, adding user login, or integrating a music API.

❖ Objective

- To create a simplified version like Spotify.
- To develop a functional music streaming web application.
- To implement an intuitive user interface with smooth navigation.
- To provide a responsive and interactive UI using HTML, CSS, and JavaScript.
- To explore web development technologies and enhance programming skills.
- To implement audio playback controls, responsive design.
- To lay the foundation for future backend integration (e.g., user authentication, databases).
- To gaining practical experience in front-end web development (HTML, CSS, JavaScript).

❖ Scope

- A fully responsive and visually appealing UI.
- Smooth music playback controls (Play, Pause, Next, Previous, mute, unmute, progress bar, volume bar).
- Dark-themed user interface for an immersive experience.
- Future enhancements: Backend integration for user authentication, database storage, and personalized playlists and more.
- Song listing with title.
- Responsive layout for desktop and mobile devices.
- Basic playlist management (static implementation).
- Static playlist data (later replaced with API integration).

❖ Modules

- **Home Page:** Displays right side playlists cards and left side playlist's songs.
- **Music Player:** Provide play, pause, next, previous, mute, unmute, volume slider for audio adjustment.
- **Playlist Management:** Displays pre-defined playlists and songs.
- **Search Functionality:** Enable users to find songs.
- **UI Components:** The sidebar, song list, player control, playlists, navigation bar.
- **Audio file:** handling and playback.
- **Progress tracking:** Real-time progress bar updates.

❖ Software Requirements

- **Code Editor:** VS Code
- **Browser:** Google Chrome
- **Web browser:** Chrome for testing
- **Local server:** Live Server extension on vs code for development

❖ Hardware Requirements

- **Processor:** Intel Core i3 (6th Gen) or above
- **RAM:** Minimum 4GB or above
- **Storage:** At least 256GB SSD or HDD
- **Operating System:** Windows/Linux /macOS

❖ Front-End

- **HTML:** Define the structure and layout of the web pages.
- **CSS:** Handle the visual presentation and styling, animations, responsive design of the application.
- **JavaScript:** Interactive features, event handling, implements dynamic behaviors, and music playback control.

❖ Back-End

(Currently, the project is front-end based. Future enhancements may include backend technologies like **Node.js, Express.js, MongoDB, or Firebase** to manage user authentication, song storage, and dynamic content).

❖ Merit

- It me helps to understanding **HTML, CSS, and JavaScript**, which are essential for web development.
- It provides a **smooth and user-friendly interface** similar to a real music player.
- It includes **play, pause, next, previous, volume control, and progress bar/seek bar**.
- Its great way for me to practice and improve my **coding skills**.
- The design and features can be easily **modified or expanded** for future improvement.
- Since it mainly focuses on the **front end**, there's **no need for a database or server setup**.
- This website Can be hosted on platforms like **GitHub Pages** for free.

❖ Demerit / Limitations

- users cannot **upload songs** or stream live music from a library.
- **Login, sign-up buttons and search bar** are just for display; they don’t function.
- It Doesn’t support advanced features like **creating playlists, shuffle, repeat, queue, or song recommendations**.
- It **lacks real-time streaming** on speaker.
- Since it’s only front-end, it **doesn’t store user preferences or history**.
- Can only play a **fixed number of songs** which added manually.

❖ Feasibility Study

- **Technical Feasibility:**

  This project is technically feasible because:

  - It uses common web technologies like **HTML, CSS, and JavaScript**.
  - The audio files are stored locally and **fetched from a predefined source**.
  - A real-time progress bar, volume control, and **basic media functions are easily achievable using JavaScript.**
  - Future enhancements like **backend integration, user authentication, and AI-based recommendations** can be implemented with technologies like **Node.js, Firebase, or Django.**

- **Economic Feasibility:**

  The project is economically feasible because:

  - **No major investment is required** since it is built using open-source technologies.
  - Hosting the project on platforms like **GitHub Pages, Netlify, or Vergel** is free for small-scale usage.
  - Future revenue generation is possible through **ads, premium subscriptions, or partnerships** if expanded.

- **Operational Feasibility:**

  The project is operationally feasible because:

  - Users can easily **navigate and interact** with the platform.
  - Basic functionalities like **play, pause, next, previous, volume control and Progress bar** are available.
  - The UI is **designed to mimic modern music streaming services**, ensuring familiarity for users.

❖ Applications

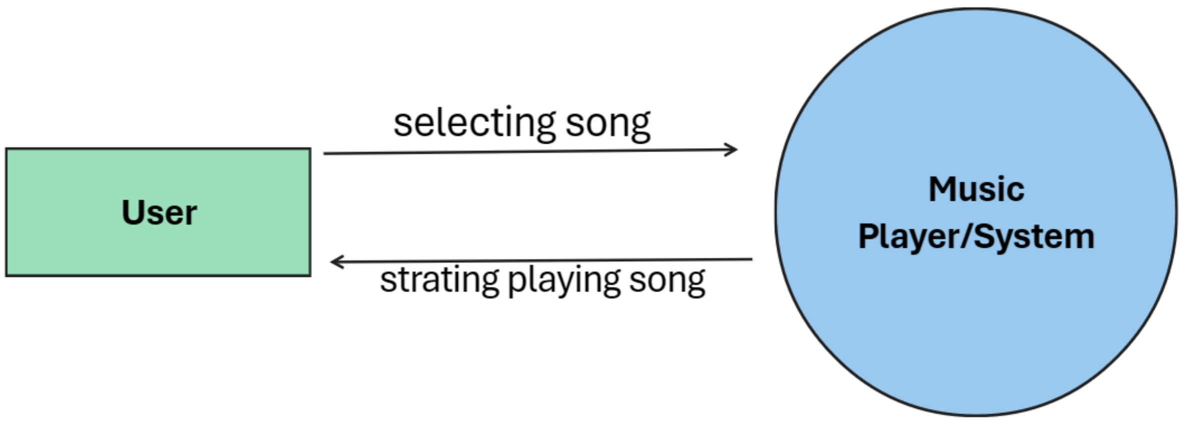
- **Personal Music Player:**
  - Users **can play, pause, skip, and adjust volume** for their local or predefined playlists.
  - A lightweight **alternative to traditional media players** like VLC or Windows Media Player.
- **Web-Based Audio Streaming Service:**
  - Can be expanded to stream **podcasts, audio books, or lectures**.
  - Useful for **radio stations or small-scale streaming platforms.**
- **Educational Platform for Web Development:**
  - Helps beginners understand **HTML, CSS, JavaScript, and Web Audio API.**
  - Can be used as a **college project** to demonstrate UI/UX, frontend development, and basic web functionalities.
- **Music Recommendation & AI Projects:**
  - Future enhancements can include **AI-based music recommendations** based on user preferences.
  - Can integrate **machine learning** to create auto-generated playlists based on listening history.
- **Custom Audio System for Businesses & Events:**
  - Can be used in **cafes, gyms, or restaurants** to play pre-curated playlists.
  - Can be modified for **event-based music streaming** where users can vote for songs.
- **Mobile & Web App Development:**
  - Can be converted into a **progressive web app (PWA)** for mobile usage.
  - Future development with **React Native or Flutter** to create a full-fledged mobile application.
- **Internal Use for Artists & Indie Creators:**
  - Can serve as a **personal music-sharing platform** for independent artists.
  - Can integrate **user-uploaded songs** for local artists to showcase their work.

❖ Future Enhancements

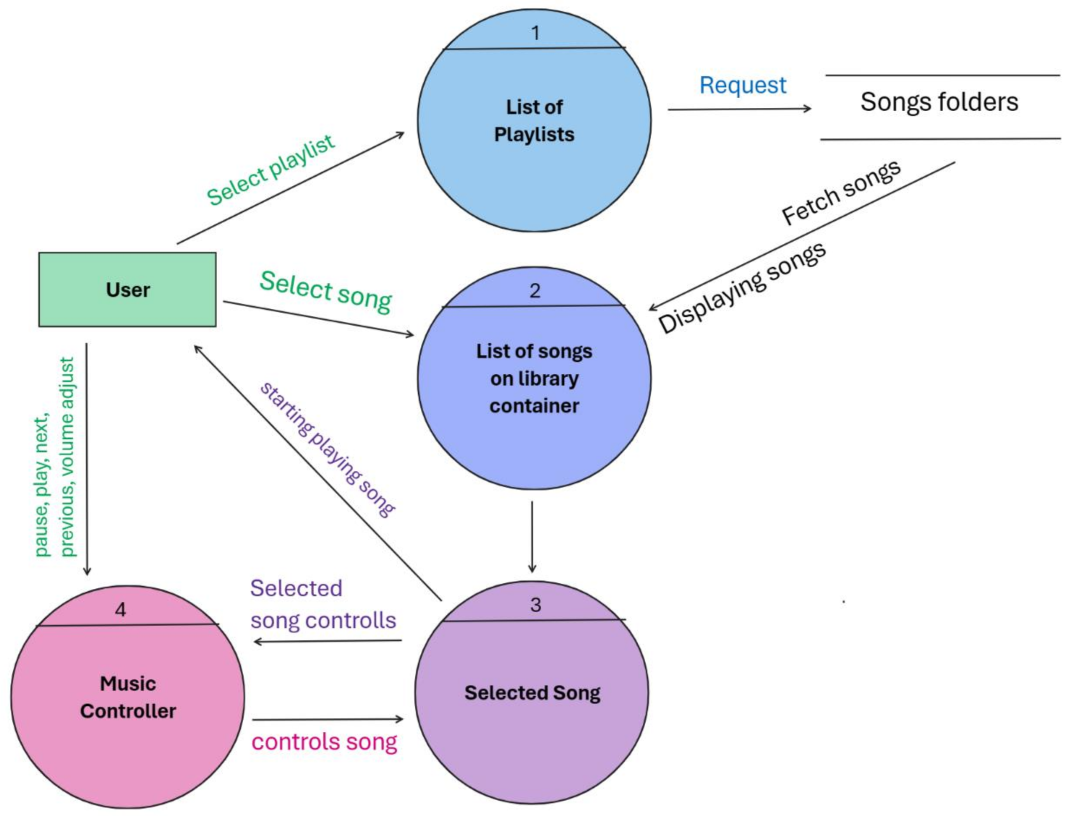
- **User Authentication & Profile Management:**
  - Implement actual **sign-up and login** functionality using Firebase, Node.js, or any backend.
  - Allow users to create, edit, and manage their **profiles**.
- **Dynamic Playlist Management:**
  - Enable users to **create, edit, delete, and share playlists.**
  - Implement a **"Like" button** for users to save their favorite songs.
- **Streaming from a Database:**
  - Instead of pre-defined songs, store audio files in a **database (MongoDB, Firebase, or MySQL)** and stream them dynamically.
  - Support cloud storage like **AWS S3 or Firebase Storage** for storing music files.
- **Lyrics Display:**
  - Show **synchronized lyrics** while playing a song (like Spotify & Apple Music).
  - Fetch lyrics from APIs like **Musixmatch**.
- **Song Recommendation System:**
  - Use **AI or ML** to recommend songs based on user listening history.
  - Implement a **"Recently Played" section.**
- **Offline Mode:**
  - Allow users to **download songs** and play them without the internet.
- **Dark & Light Mode Toggle:**
  - Add a **theme switcher** for a better user experience.
- **Social Sharing & Collaboration:**
  - Let users **share playlists** on social media.
  - Implement **collaborative playlists** where multiple users can add songs.
- **Podcast & Radio Integration:**
  - Support **podcasts and live radio streaming.**

❖ Data Flow Diagram (DFD)

➢ 0 level -



➢ 1 level -



❖ File System

Each songs album having **.json file** and **cover images**.

.json file -

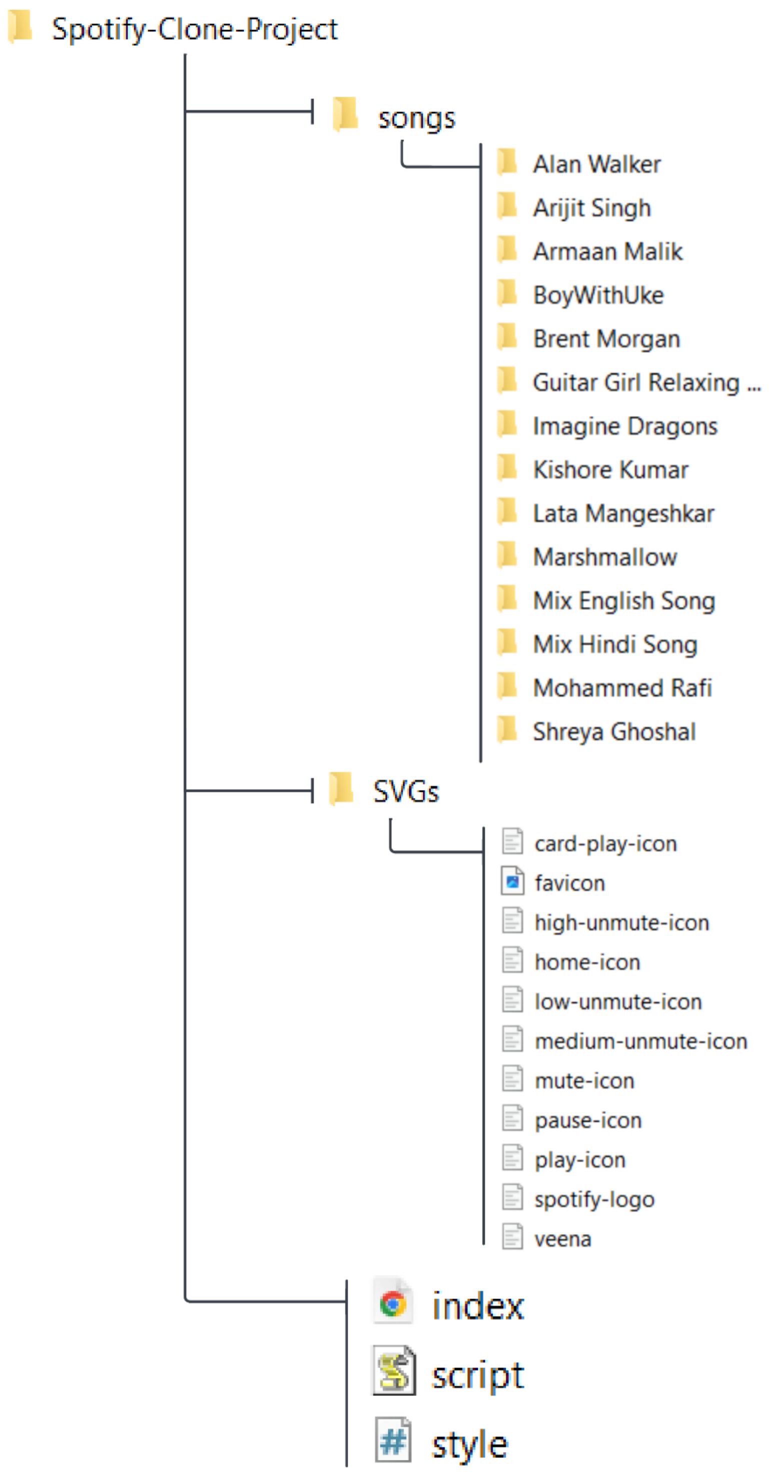
It contains album’s title which is get fetch by JavaScript and display on a screen in a web application.

Cover image –

cover’s image art as album covers which is get fetch by JavaScript and display on a screen in a wed application.

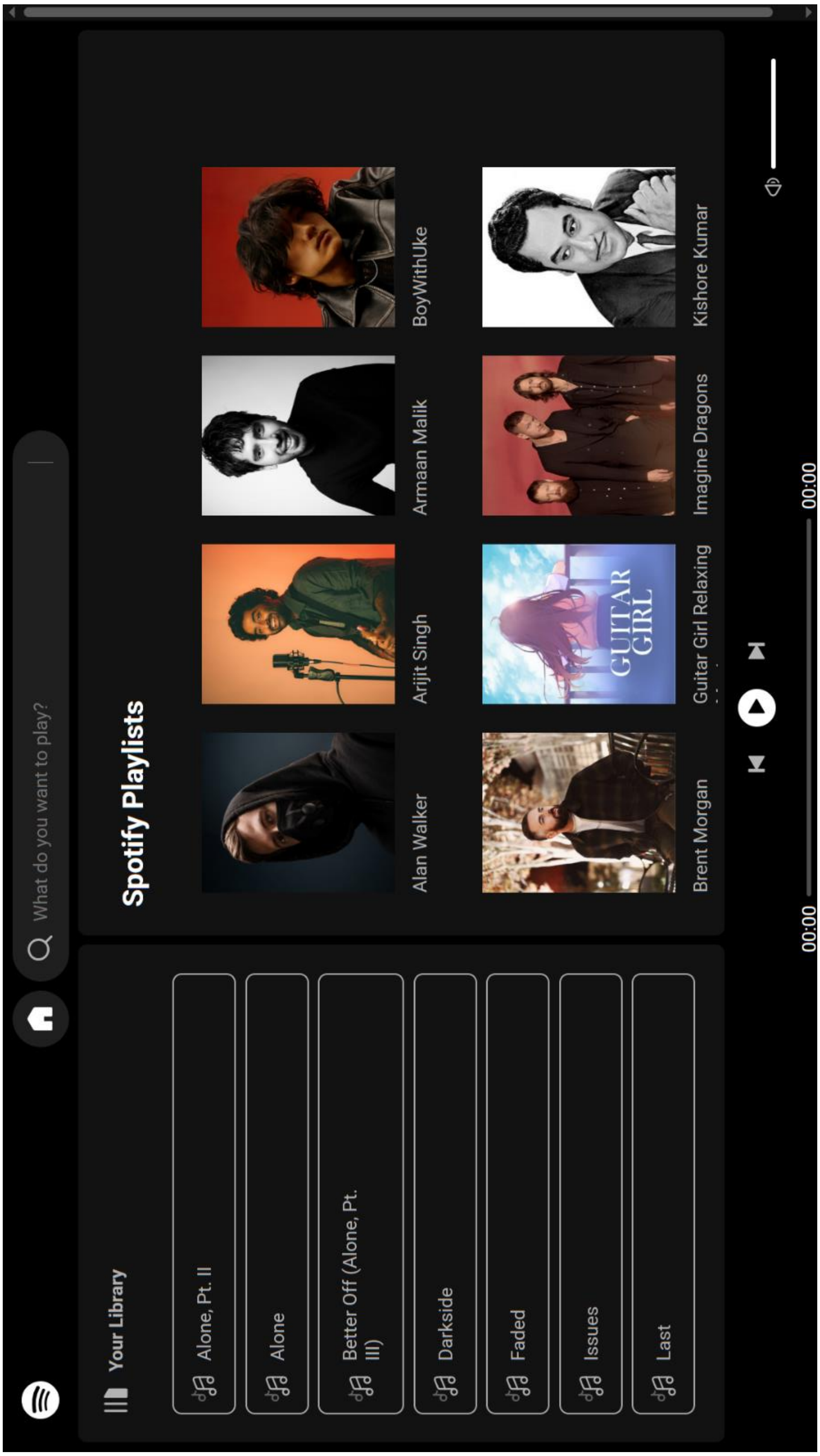
SVG image –

Some SVG images are accessed from the SVGs folder, while others are inserted directly in the code and displayed in the web application.

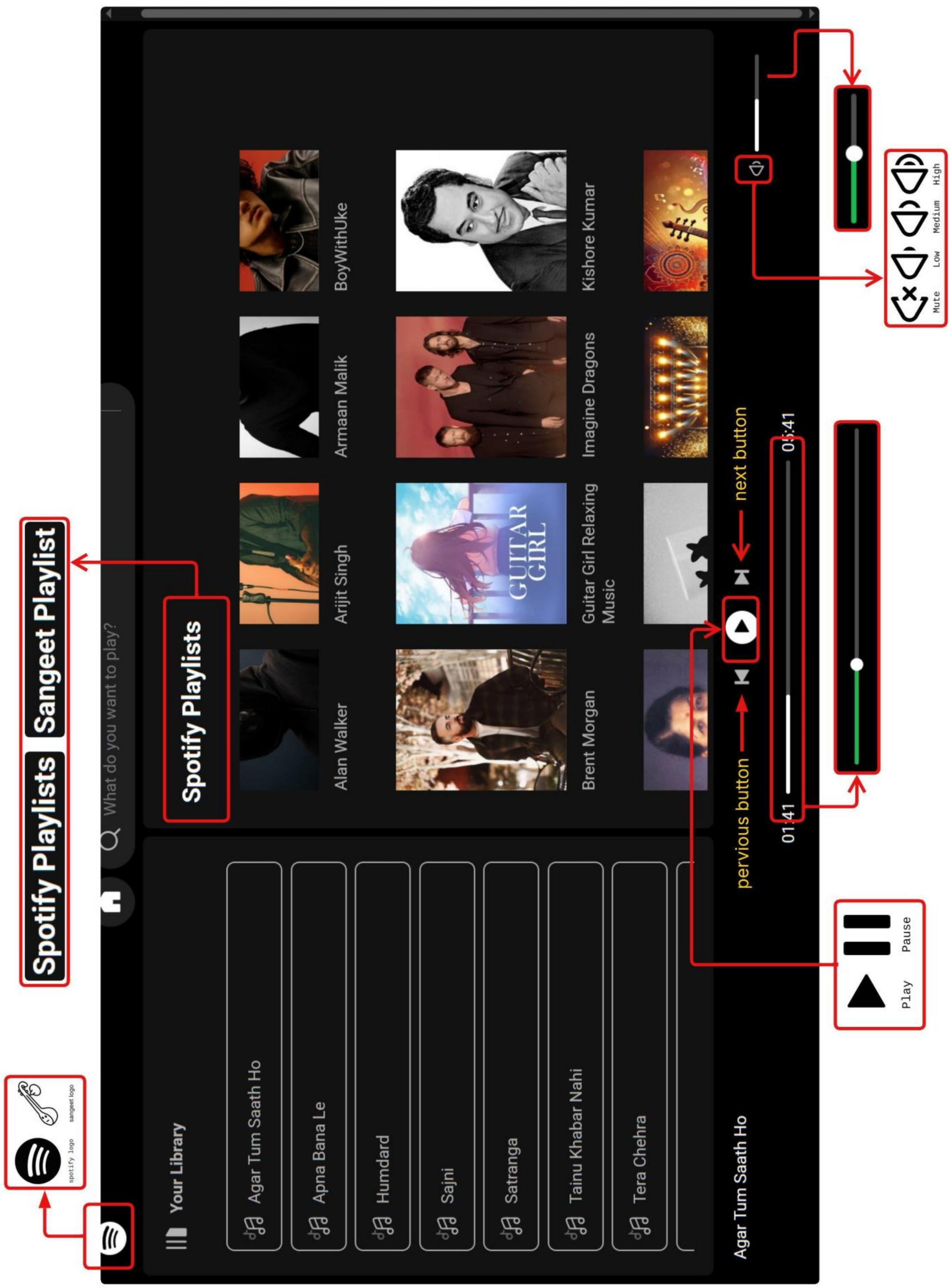


❖ Output Screenshots:

When open web application, it looks like this and application page size (1214px X 715px)



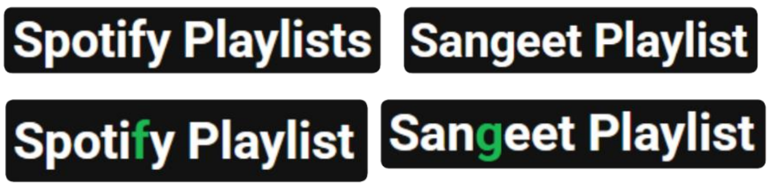
Here in this image are showing the some features:

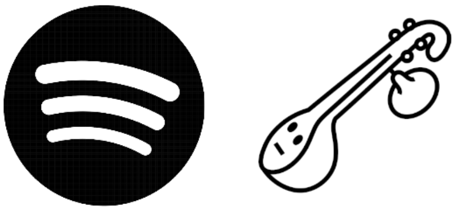


Here is explanation of a above image which show some features:

1. Change on a logo and text -

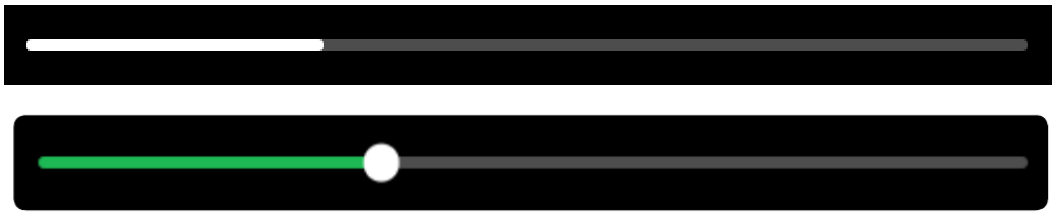
When click on web application logo, it changes logo and text “spotify playlists” into “sangeet playlists” and we hover cursor on specific letter it changes text color white to green


Spotify Playlists
Sangeet Playlist
Spotify Playlist
Sangeet Playlist



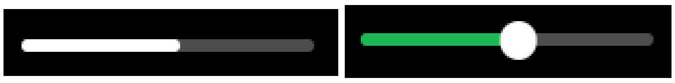
2. seek bar

Seek bar allows us to move to different parts of a song by click on the seek bar. It shows the **current playback position** and updates in real-time as the song plays and When **hover on seek bar** then progress line color **get change from white to green.**



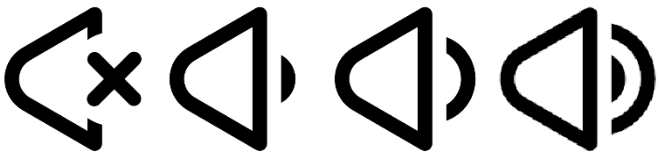
3. volume bar

A **volume bar** allows us to **increase, decrease, or mute the audio volume**. It visually represents the **current volume level** and provides control over the audio output. When **hover on volume bar** then volume level line color **get change from white to green.**



4. volume level

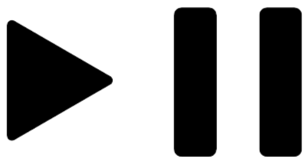
Based on volume level, volume icon get change as fellows:



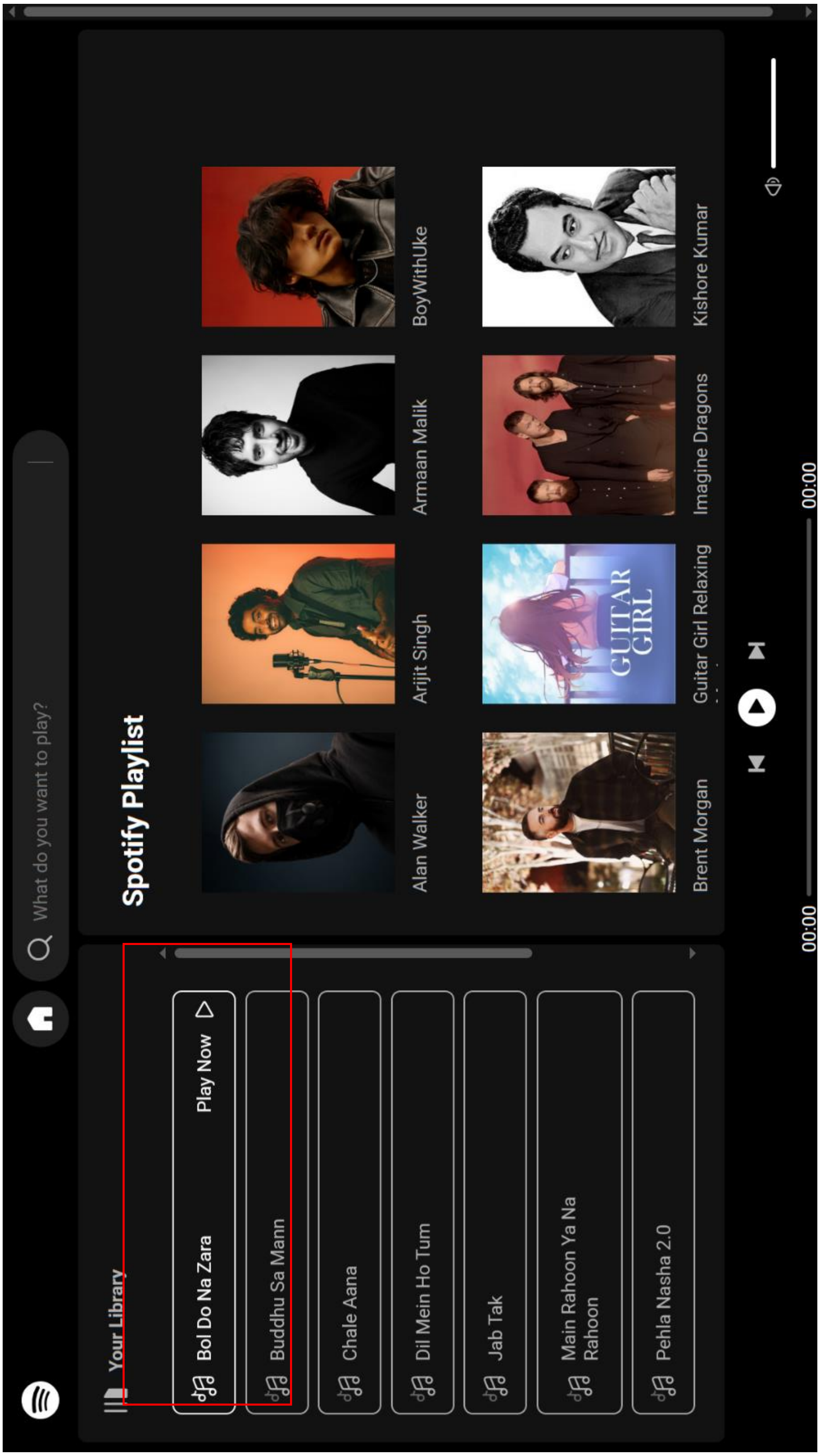
- **At zero it get mute icon** and volume get stopped. We can also click on any volume icon to stops the audio sound.
- If range is between **1 to 33** then **low** volume icon shows up.
- If range is between **34 to 66** then **medium** volume icon shows up.
- If range is between **37 to 100** then **high** volume icon shows up.

5. play or pause

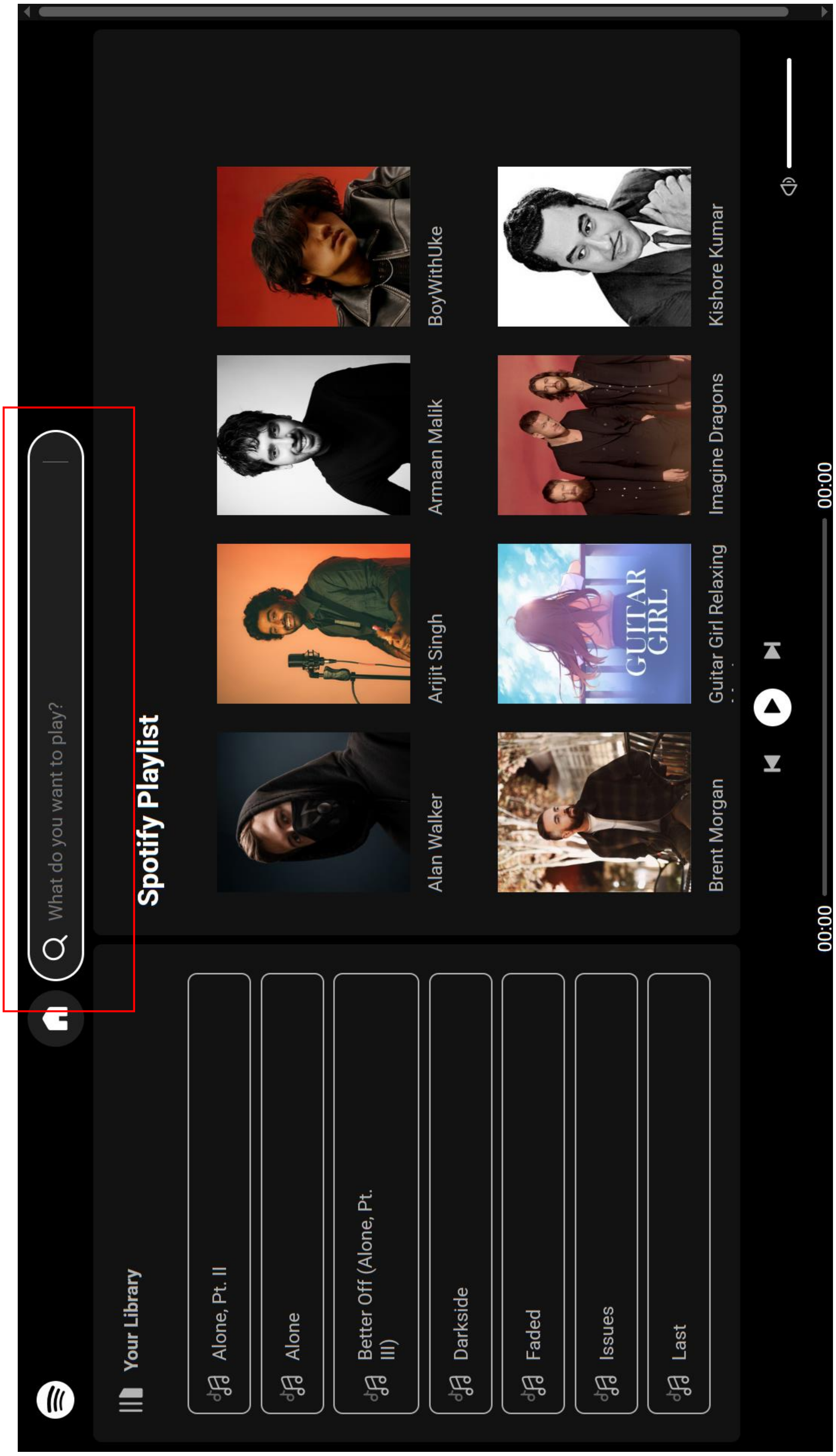
When we click on **play** button then **song gets play** and click on **pause** button to **stop the song.**



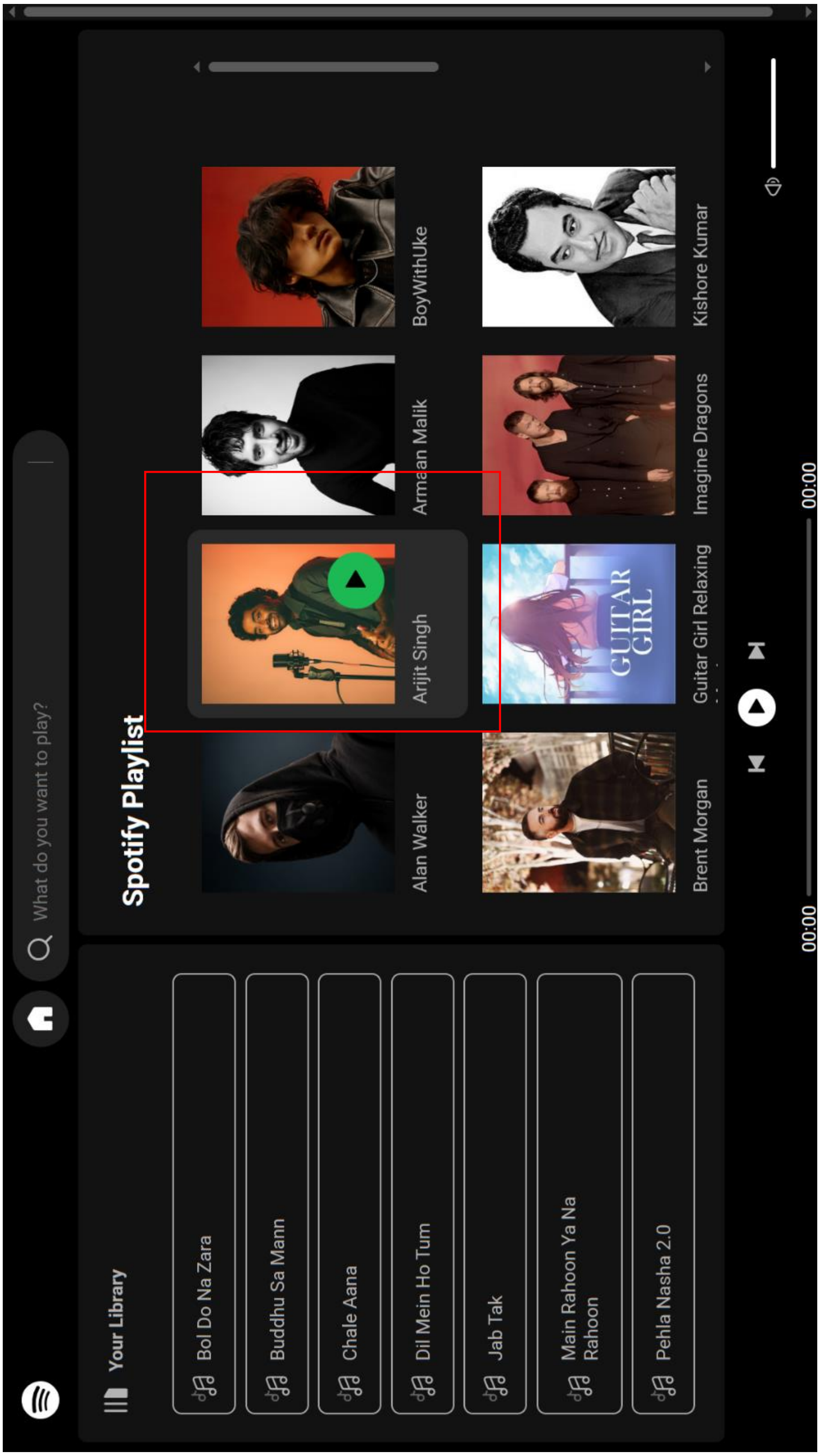
when we hover cursor on list of songs, it looks like this which shows in a red box:



when click on search bar, it looks like this which shows in a red box:

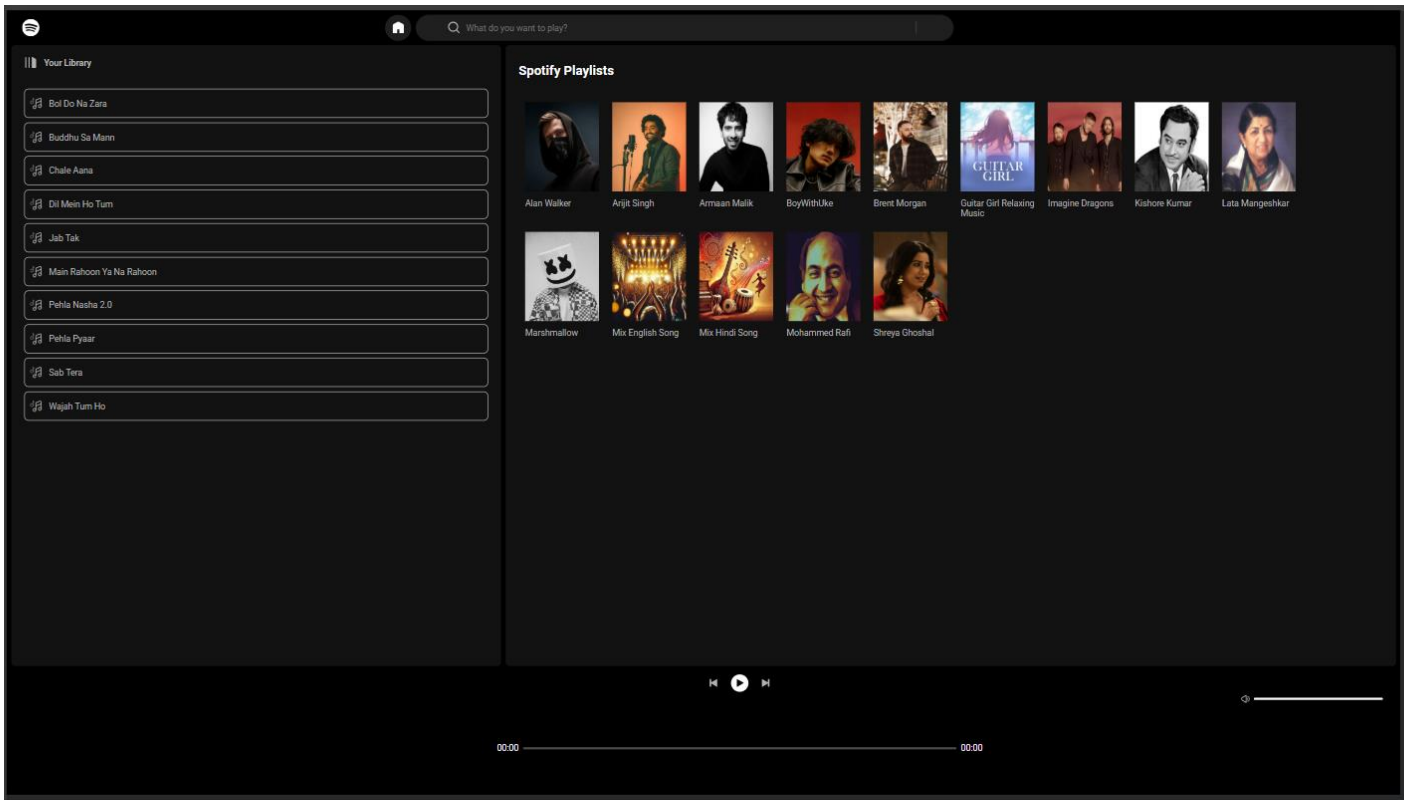


When we hover cursor on any album card, card around background will change and play icon shows up like this that's show in a red box:

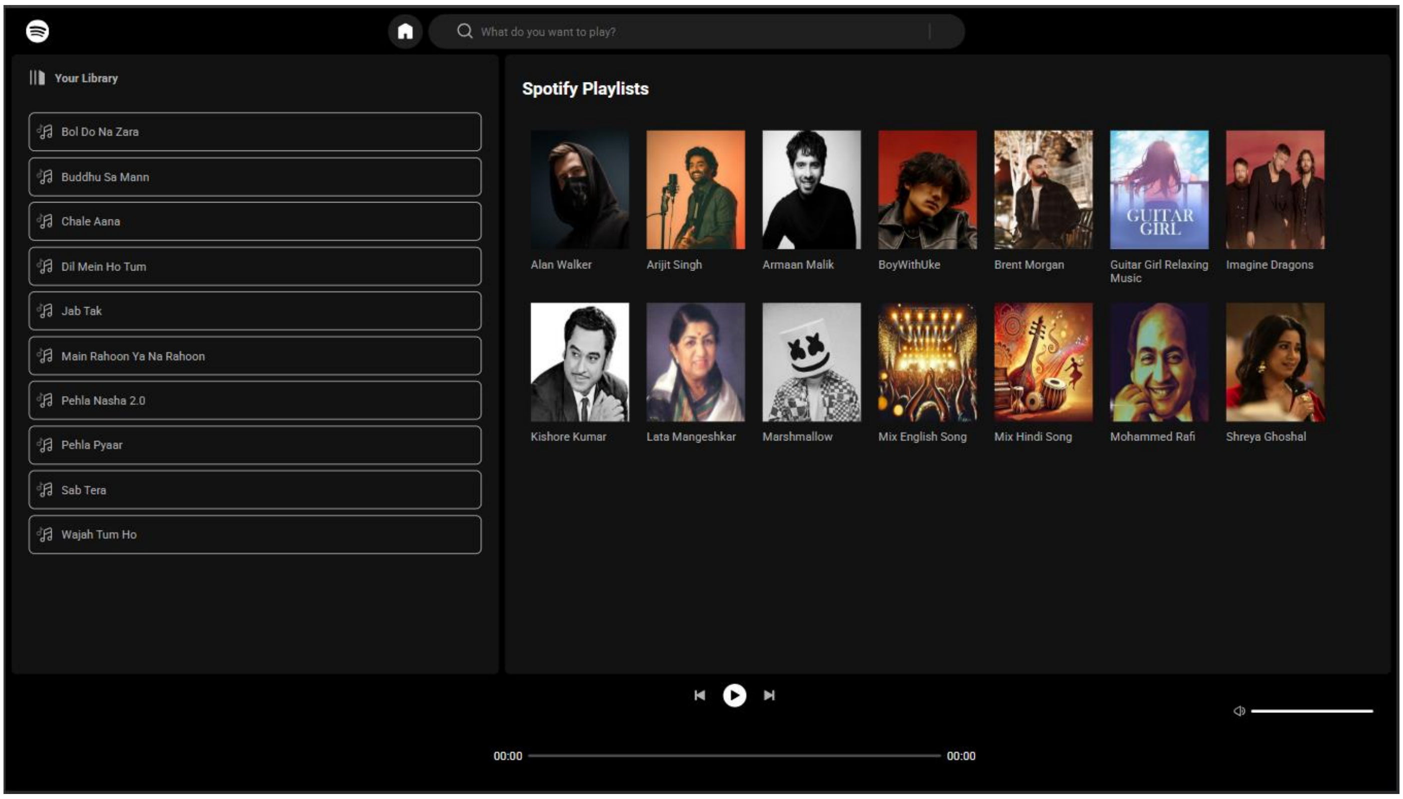


❖ Responsive Design Screenshots based on Computer and Tablets

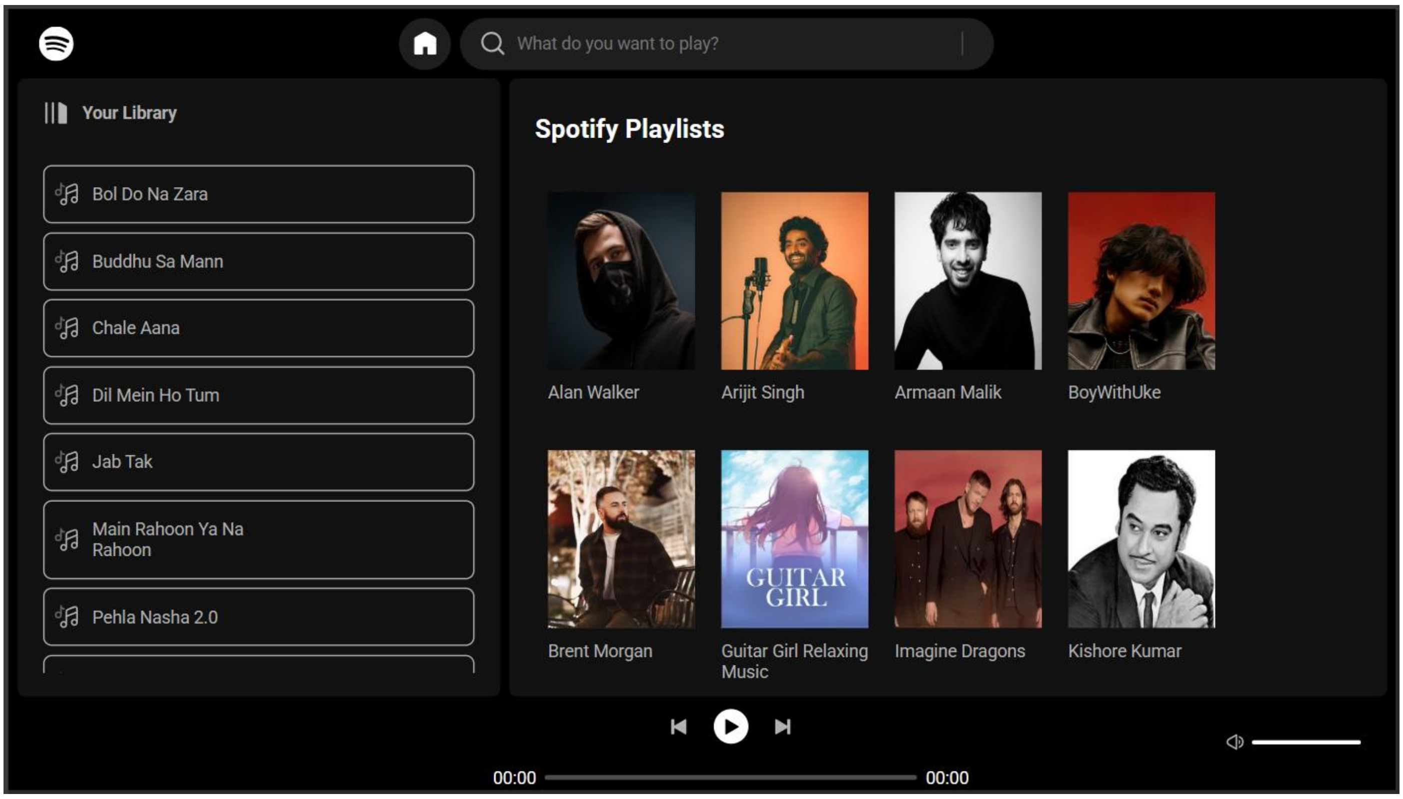
Quad HD - 2k - 2560px X 1440px - 16:9 aspect ratio



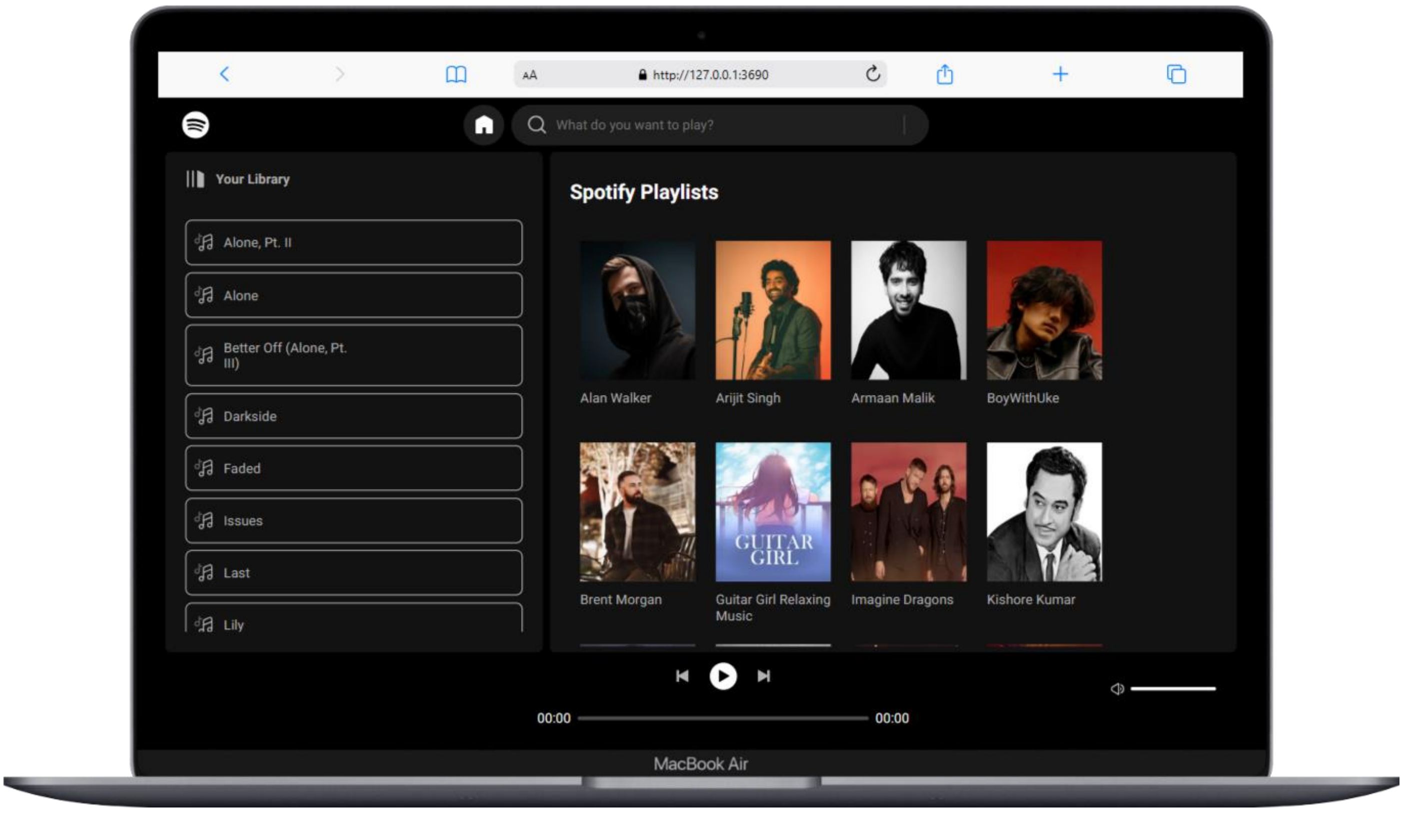
Full HD - 1920px X 1080px - 16:9 aspect ratio



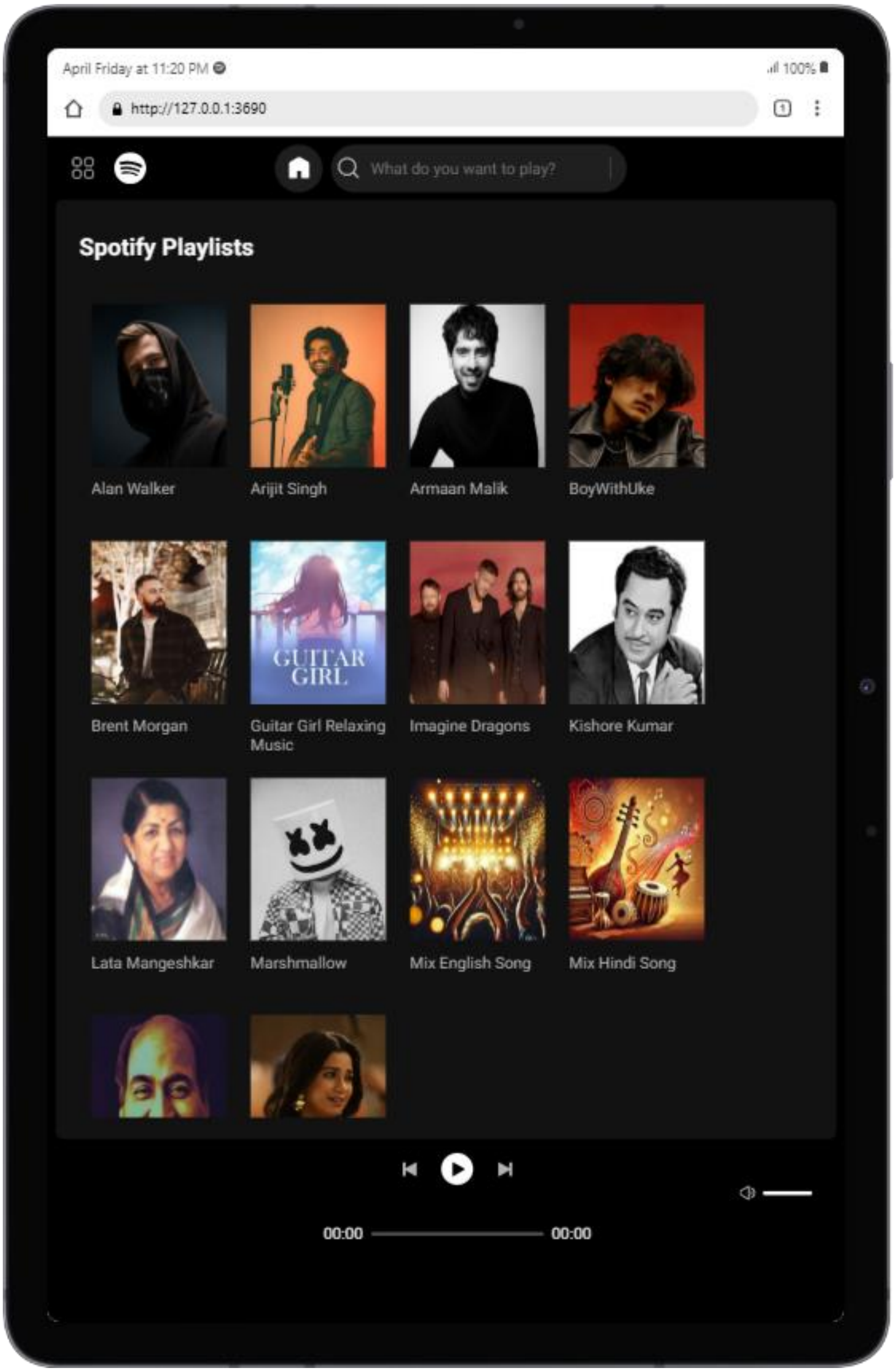
HD - 1280px X 720px - 16:9 aspect ratio



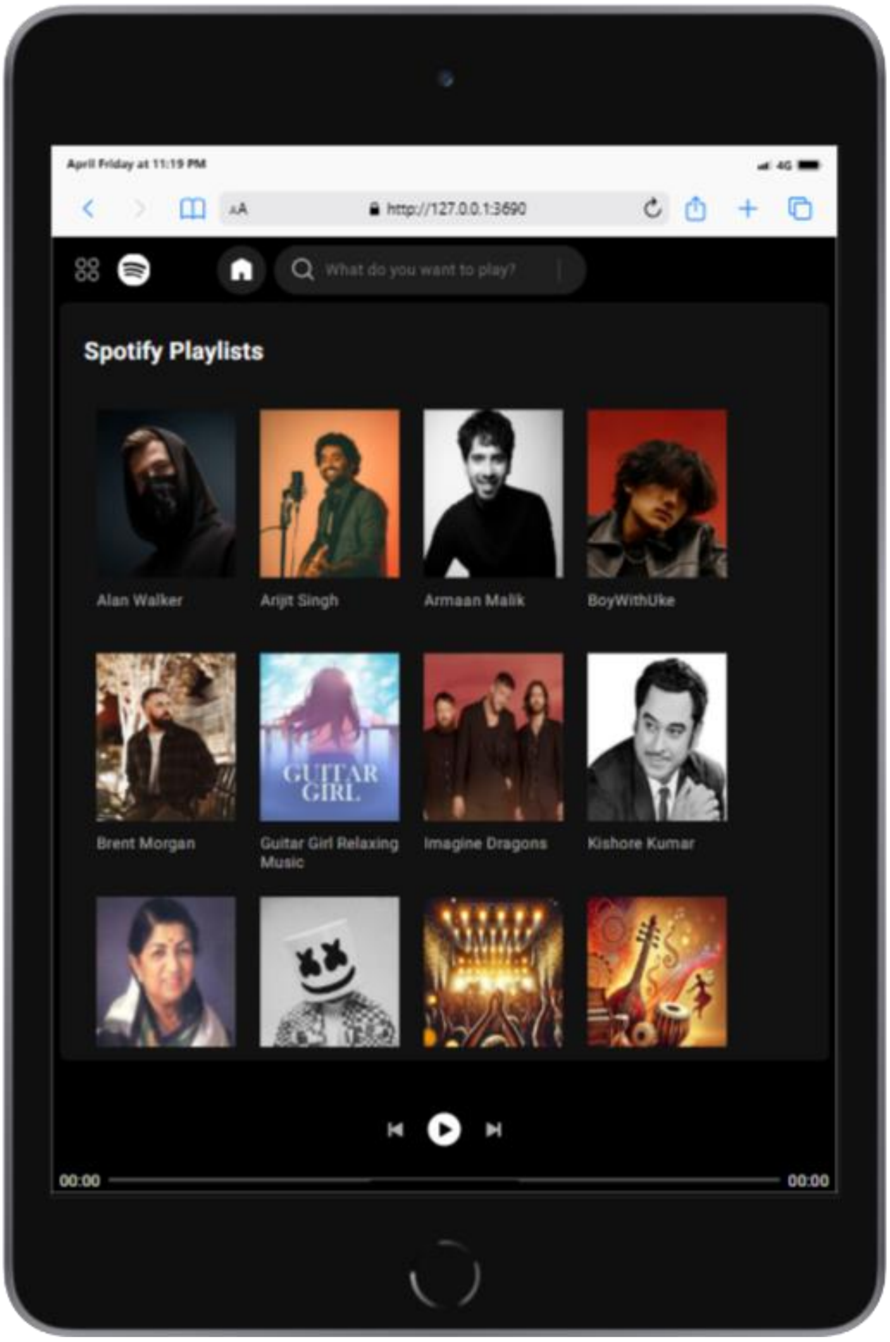
MacBook air – size: 1280px X 800px



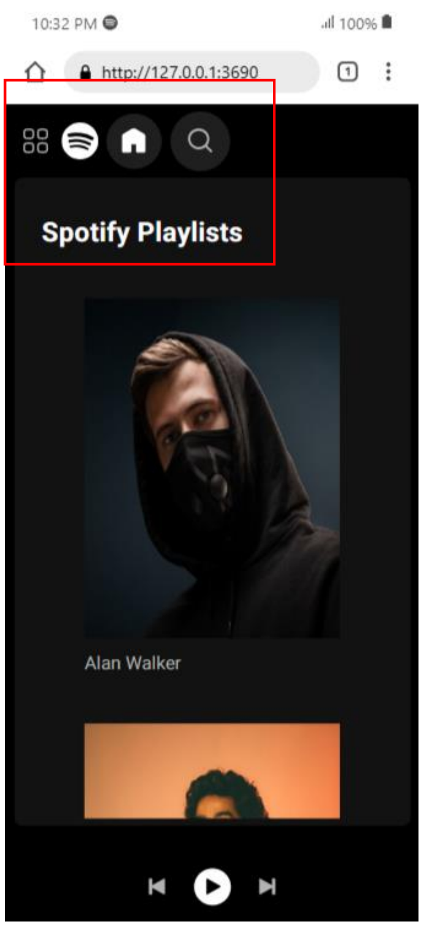
Galaxy Tab S7 – size: 800px X 1280px



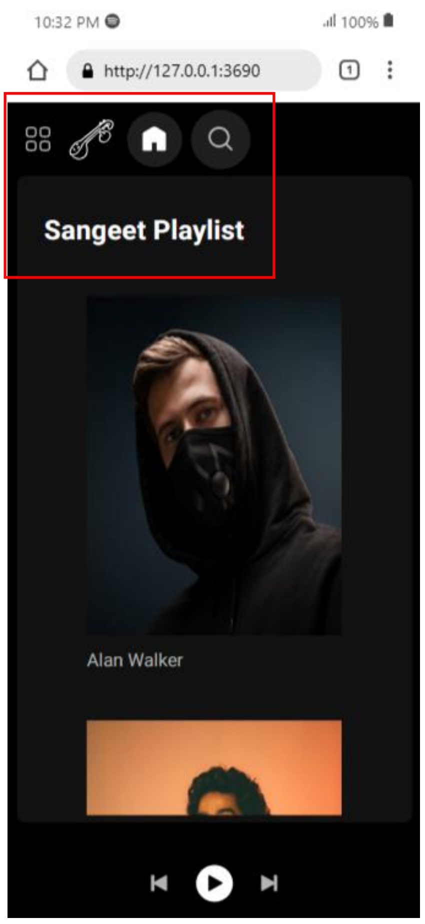
iPad Mini– size: 768px X 1024px



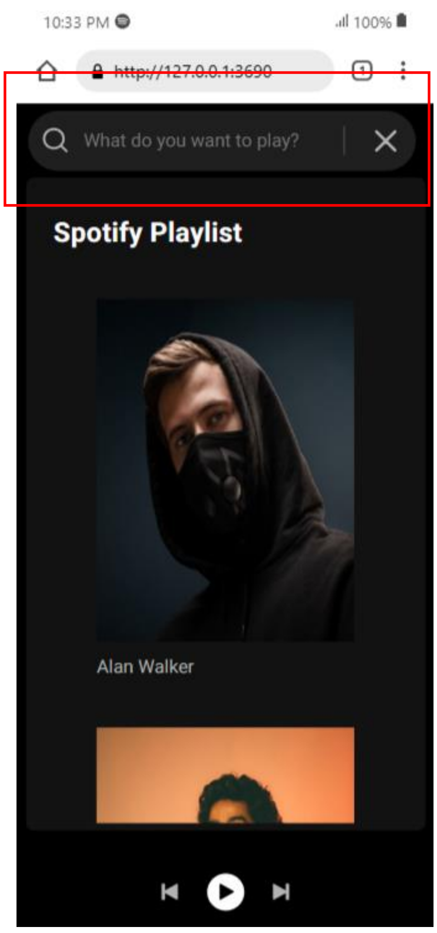
❖ Responsive Design Screenshots based on mobile



1



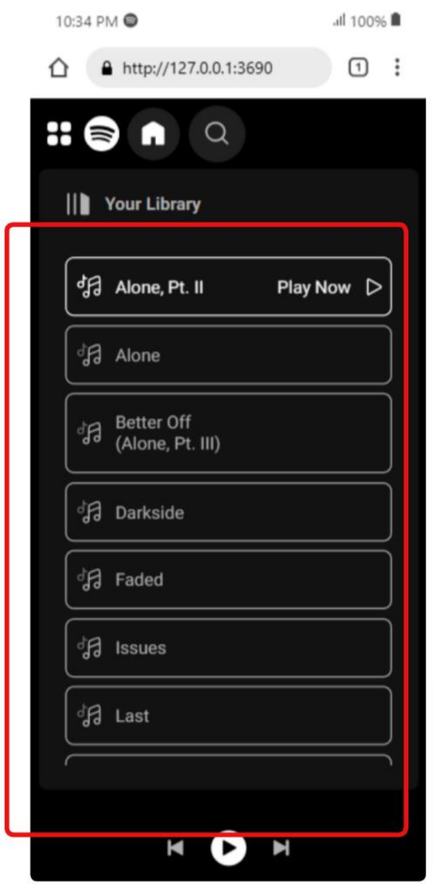
2



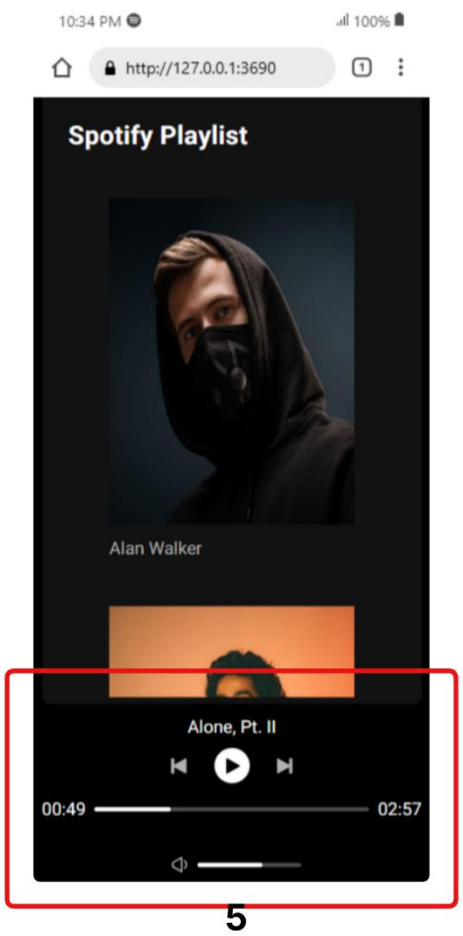
3

In above image no 1, When click on web application logo, **it changes logo and text "spotify playlists" into "sangeet playlists"** and we hover cursor on **specific letter it changes text color white to green**. (similar to pc version)

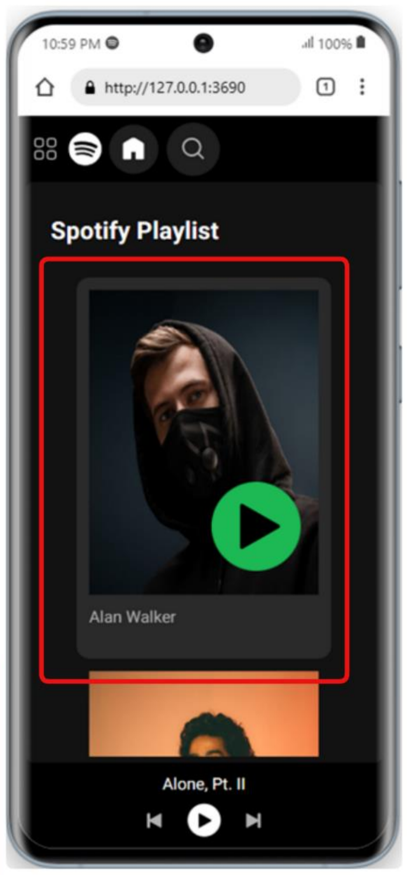
In above image no 1 or 2, when we **click on search icon** then search bar will be **opened** and in image no 3, when **we click on cross** icon in search bar then **search bar will be close**.



4



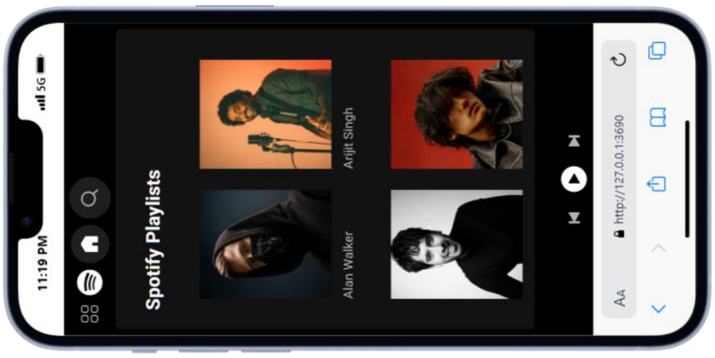
5



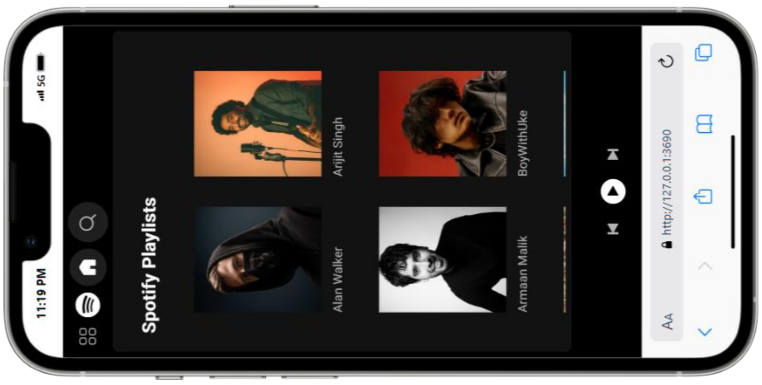
6

In above image no.4, when we **click on menu icon** then **song list will be opened** and click again to close. In above image no.5, we can **see** how the **play controller looks like in mobile**. And in above image no.6, when we hover cursor on any album card, **card around background will change** and play icon shows up like this that's show in a red box.

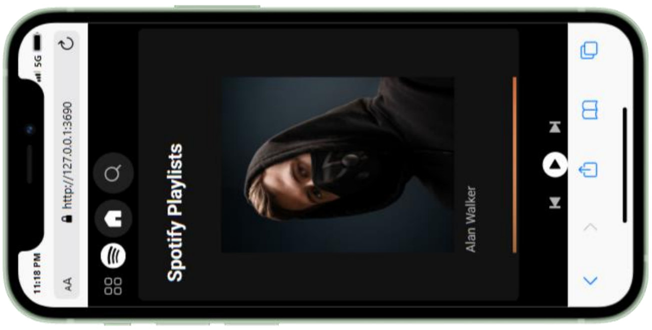
iPhone 14 (iOS 16) – size: 844px X 390px



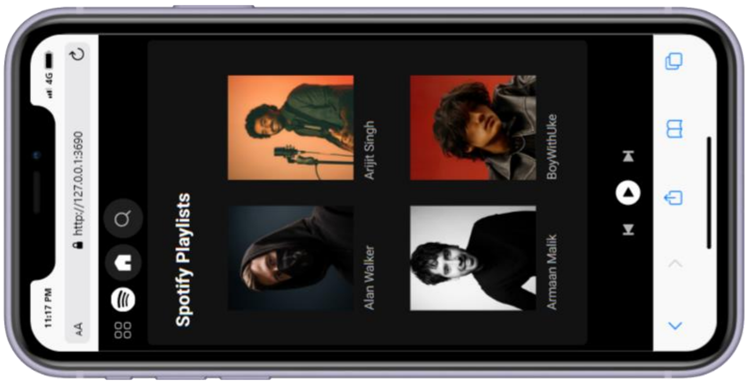
iPhone 13 PRO MAX – size: 926px X 428px



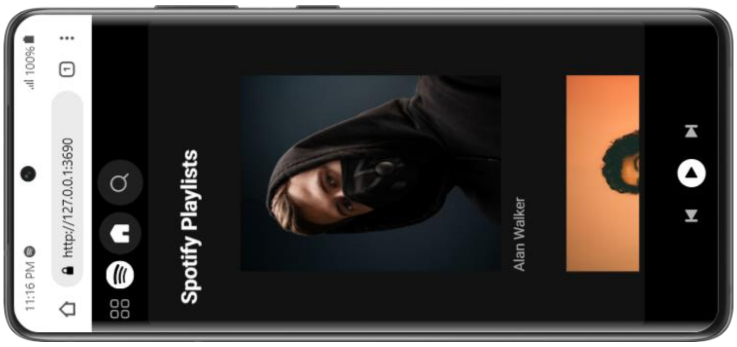
iPhone 12 Mini – size: 780px X 360px



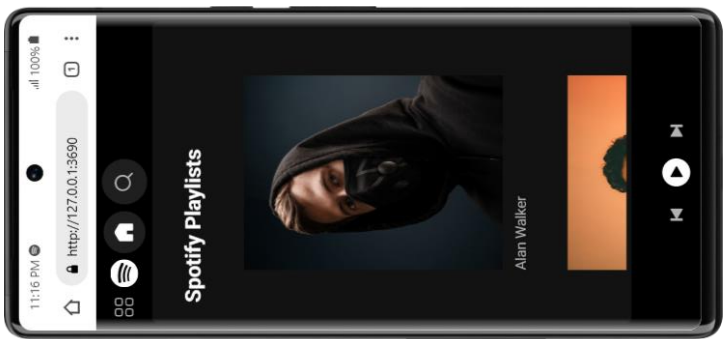
iPhone 11 – size: 896px X 414px



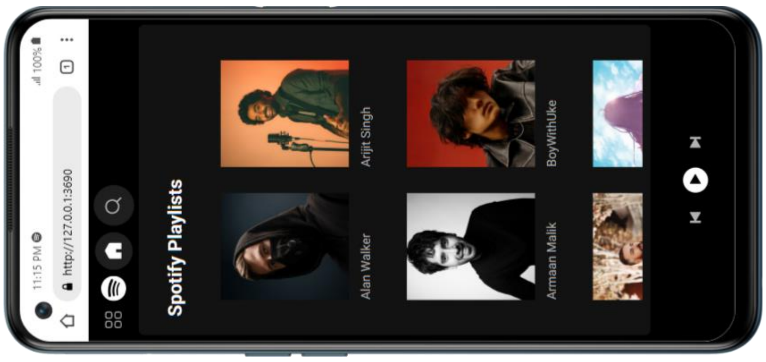
Galaxy S21 Ultra – size: 800px X 360px



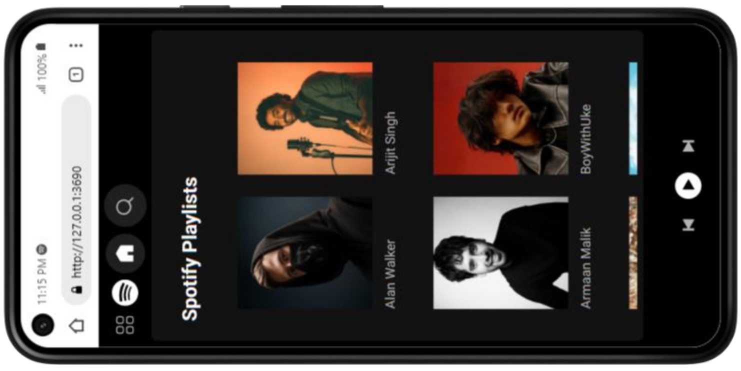
Google Pixel 6 PRO – size: 780px X 360px



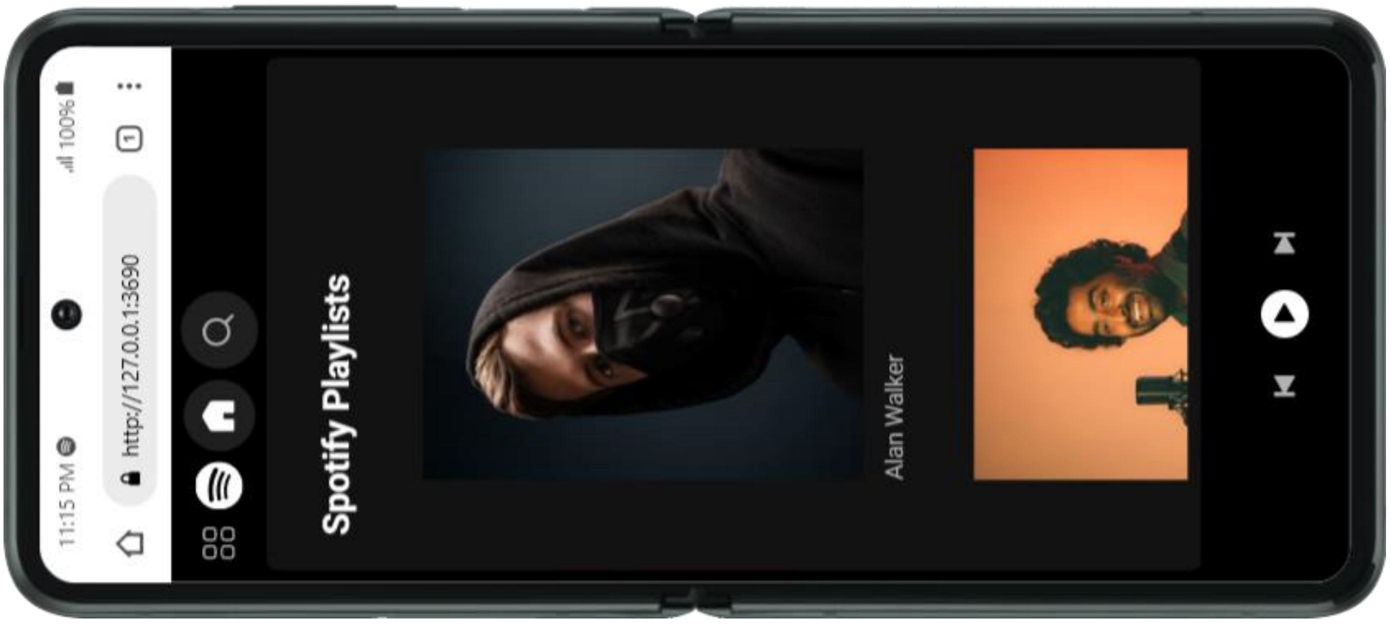
Oneplus Nord 2 – size: 915px X 412px



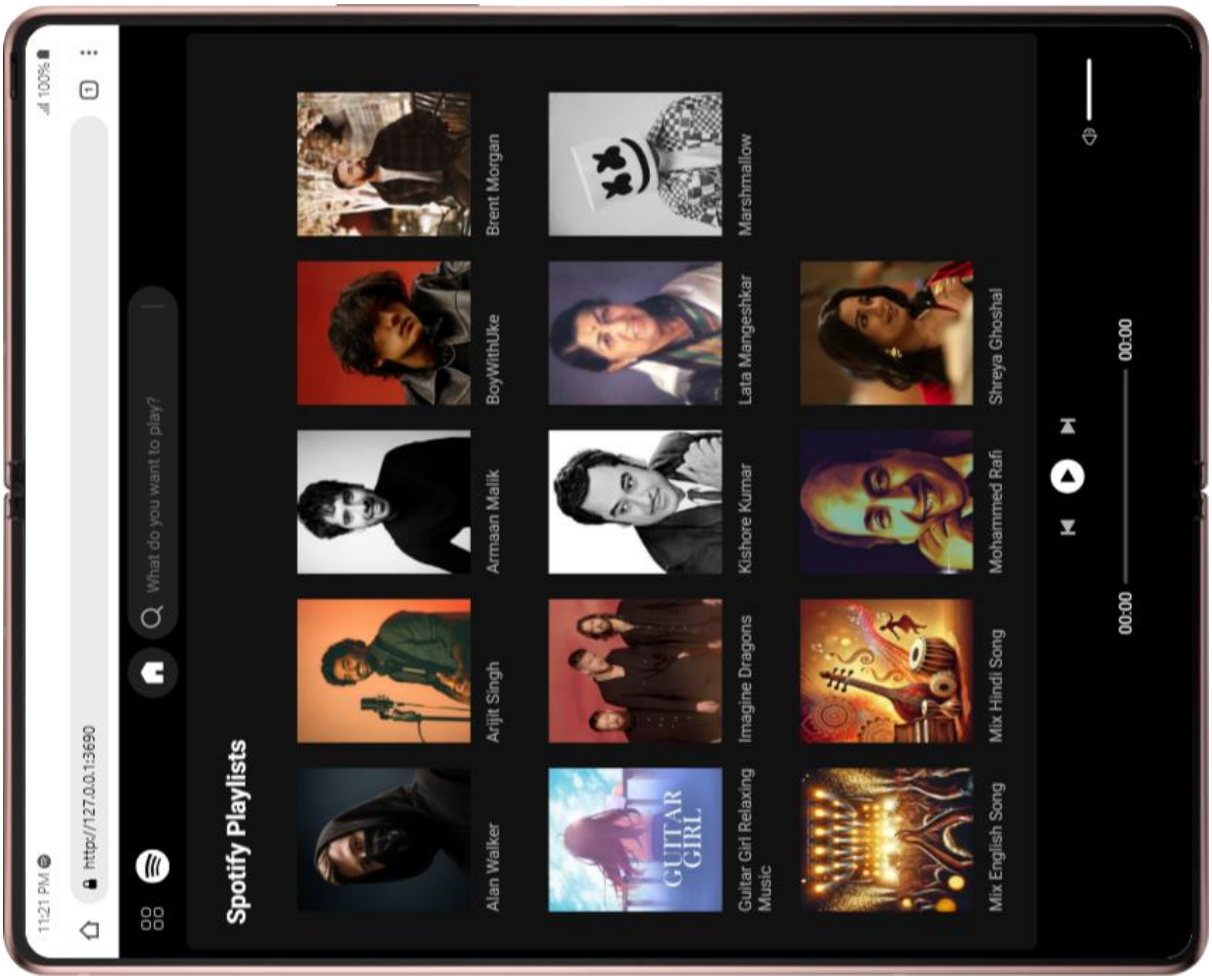
Google Pixel 5 – size: 851px X 393px



Galaxy Z Flip 3– size: 880px X 360px



Galaxy Fold 2 – size: 884px X 1104px



❖ Conclusion

The **Sangeet: Music Streaming Platform** is a web-based project inspired by Spotify, designed primarily for learning and skill development in front-end web technologies. Built using **HTML, CSS, and JavaScript**, it provides a functional and interactive music player with essential playback controls like **play, pause, next, previous, volume adjustment, and progress tracking**.

This project serves as a stepping stone in understanding web development concepts, offering a **responsive, visually appealing user interface** with smooth navigation. Although it lacks backend functionality, future enhancements such as **user authentication, personalized playlists, AI-based song recommendations, and cloud-based streaming** can be implemented to extend its capabilities.

The feasibility study confirms that the project is technically, economically, and operationally viable. It can be hosted for free on platforms like **GitHub Pages or Netlify**, making it accessible for users. Despite some limitations—such as the inability to upload songs or stream live music—the project provides valuable hands-on experience and a solid foundation for future development.

In conclusion, **Sangeet** is an excellent educational project that enhances front-end development skills and serves as a prototype for a more advanced music streaming platform in the future.

❖ References / Bibliography

- Artificial intelligence (AI): ChatGPT, Gemini.
- YouTube
- MDN Web Docs:
  https://developer.mozilla.org/en-US/docs/Web/HTML
  https://developer.mozilla.org/en-US/docs/Web/CSS
  https://developer.mozilla.org/en-US/docs/Web/JavaScript