## LIST OF FIGURES

## LIST OF TABLES

# 1. INTRODUCTION

Any music app provides free and unlimited access to music online or download songs, but some features are not easy to work with, like when searching for an artist or a particular song, there are podcast recommendations, which are present only for more retention rate, and this was our main motivation to work on. It has enabled us to understand how popular music streaming apps work and fix those flaws that are not user friendly.

We aim to design a list of songs based on a user's search that could be in the form of a table, which is much easier to work with.

## 1.1 BACKGROUND

Music has turned out to be emotion enhancer and has played a major role in shaping history. Thus, we have developed a web application to provide the best music streaming service to users. Musify provides all types of songs that exist upto today's date and recommends songs further to the user according to their tastes.

The objective is to recommend music to the user based on the type of music and the artist they're listening to, in an organized way in tables. The system provides searching music to the user and also shows songs related to search by recommending it.

## 1.2 MOTIVATION

Musify provides accurate searching and recommendation to the user as it provides proper search results and also avoids showing podcasts and other irrelevant search results . It is a free music streaming service that does not cost anything and it is also add free unlike todays music streaming apps, that require money from customers to give them an add free experience.

In order to provide music uninterrupted and an add free experience to our users we came up with a solution in form of our own system, where it will provide the all songs as well as big hits till todays date.

## 1.3 PROBLEM DEFINITION

The purpose of website is the established fact that music lovers are increasing day by day. One of the main purposes of website is to provide and recommend music to users according to the music they listen to. The music listeners are enjoying a lot but there is the problem of ads which convinces the user to buy premium membership , whereas our system does neither has podcasts nor annoying ads.

## 1.4 SCOPE/ASSUMPTIONS

Musify is a web application with general features like search and recommend. User can search the songs he/she likes as well as listen to songs separated according to genre and artists. It provides songs according to users tastes and recommends relatable songs according to their mood and also doesn’t ruin the groove by interrupting with ads.

## 1.5 ISSUES/LIMITATIONS

There is no real-time updation of recommendations in the application. Also, the number of parameters for recommending songs or artists to the user are less which make it less effective.
There exists no option for multiple users to access the applications with different accounts.

# 2. LITERATURE SURVEY

## 2.1 Related Works

- ### 2.1.1 Music Recommendation System Based on Genre Distance and User Preference Classification

Here, the system analyses the usage patterns and consumption history of the user to recommend related songs to the user. But this may lead to a cold start in the system, leading to irrelevant recommendations. Cold start is a process in which there are no recommendations given to the user and the user has to start over, which leads to a lower retention rate.

A user always needs to give input on which type of song he wants to listen to. Here the app displays songs that are irrelevant to the user's taste.

### 2.1.2 Music and Movie Recommendation System

Here, the system classifies the songs based on the emotions they portray, but this might recommend songs of all possible genres, which the user might not like, making the system less effective.

The system might also recommend songs that the user dislikes.

This music and recommendation system for music streaming services uses primarily relational diagrams, which simply suggest songs from similar users' playlists. This music and movie recommendation system is recommended based on dislikes, and this primitive method does not personalise the suggestions for the user, so we decided to explore an alternative approach that analyses the features of the music. Our goal was to design our own song recommendation algorithm that suggests songs based on the features of the music that users enjoy.

## 2.2 Existing Systems

- ### 2.2.1 Music Player – Audio Player

This music player application, developed by '*Mobile_V5*', provides a great experience through its simple Graphic User Interface. It takes manual input of local files on the device and plays them according to the user's requirement.

Pros-

5 band equalizers

Edit the song details

Notification status support

<u>Cons</u>-

No recommendation system available

Only available on android and iOS devices

### 2.2.2 Custom Music Player

This music player is an open-source web application, developed by Mr. Abhishek Tiwari, using scripting languages like HTML, CSS and JAVASCRIPT. Its GUI provides simple features such as Play, Pause and Skip.

<u>Pros</u>-

Simple and minimalistic GUI

<u>Cons</u>-

Very small music database

Fewer features

No recommendation system available

## 2.3 Analysis

| **Existing System** | **Year** | **Publisher** | **Limitations** | **Our advantage over the existing system** |
|---|---|---|---|---|
| Music Player – Audio Player [2.2.1] | 2013 | Mobile_VS | No recommendation system is available | Music Recommendation based on Listening activity |
| Custom Music Player [2.2.2] | 2021 | Abhishek Tiwari | Very less features and small music data<br>Lack of recommendation system | Searching the database for songs is possible |

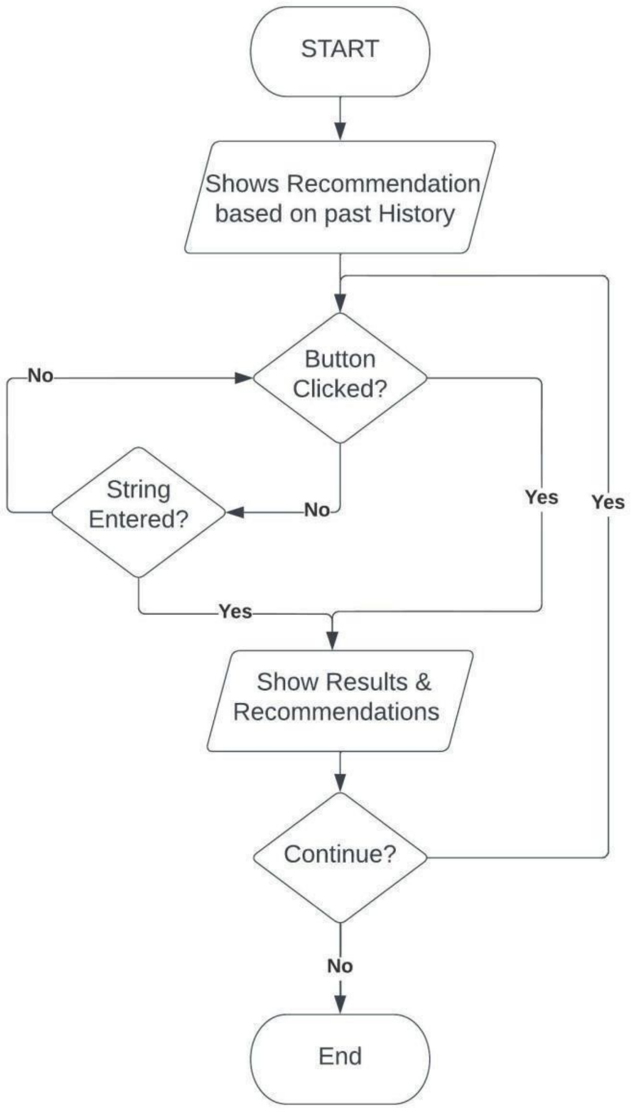
## 2.4 Gap Identified

The applications compared above didn’t have a very interactive Graphical User Interface and didn’t have a recommendation system. Also Searching for a particular song or artist or songs of particular language is not available.

These features are expected to be present in any of the music applications, but the absence of these features makes these apps less appealing.

# 3. SYSTEM DESIGN

## 3.1 Flow Chart


START
Shows Recommendation based on past History
Button Clicked?
No
String Entered?
No
Yes
Yes
Yes
Show Results & Recommendations
Continue?
No
End

## Flowchart Explanation:

The app starts by showing recommendations based on the user's listening activity and popularity of songs. Then the application checks if a button is clicked or a string is entered.
Then it shows the related results and recommendations based on the recommendation algorithm.
After this cycle is completed, is the user enters a string in the search bar again or clicks a button, the cycle starts again by showing search results and recommendations.

## 3.2 Recommendation Algorithm

```
Step 1: Start
Step 2: Fetch songs based on listening history and popularity
            If( popularity of songs > threshold)
            Display the related results
Step 3: User Enters Search String or Clicks a button  Step
4: IF String is Entered
            show Search Results
                        Fetch related results from database
            If (popularity of songs > threshold)
                                    Display the results
           ELSE IF Button is Clicked
           show Search Results
                        Fetch related results from database
                        If (popularity of songs > threshold)
                                    Display the results
Step 5: End
```

## Algorithm Explanation:

As soon as the web application is launched, the application gets the songs from databases based on the user's listening activity and popularity. If The popularity of the songs passes the threshold, then the songs are displayed.
If the user clicks a button or uses the search bar to search a specific category, the application displays the search results exactly according to the search string and displays recommendations based on listening activity and popularity

# 4. IMPLEMENTATION DETAILS

## 4.1 System Requirements

- Memory: Minimum 2GB Required
- Hardware: Keyboard, mouse, 4GB ram, monitor.
- Software: Web Browser, Eclipse IDE, Apache Tomcat Server

### Apache tomcat server:

Apache Tomcat is a web container. It allows the users to run Servlet and JSP (Java Server Pages) that are based on the web application. It can be used as an HTTP server. It can be used as separate product with its own internal web server.

### Web Browser:

Any web browser with its latest version will work for this application.
The Apache Tomcat Server uses the browser to launch the application.

### Eclipse IDE for Enterprise Java and Web developers:

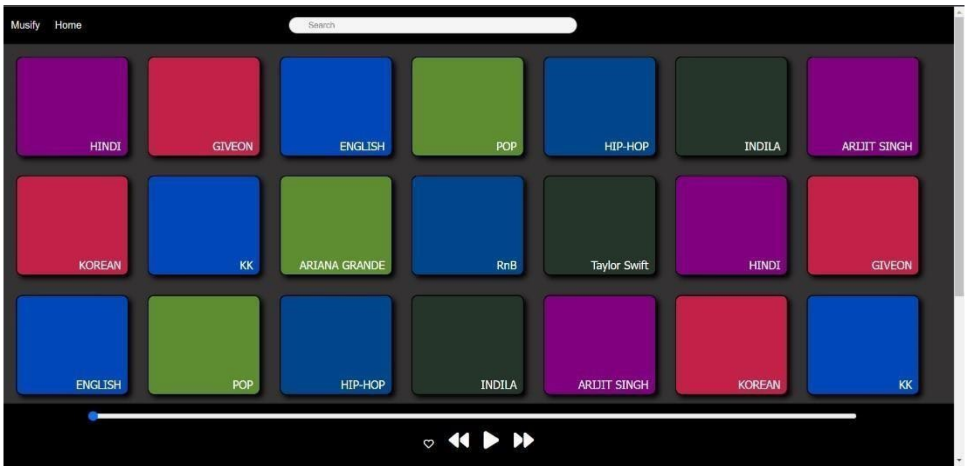
This Eclipse IDE provides tools for developers to work with Java and Web application, using a Java IDE, tools or JavaScript, TypeScript, Java Server Pages (JSP), Java Servlet and many more.

### Implementation:

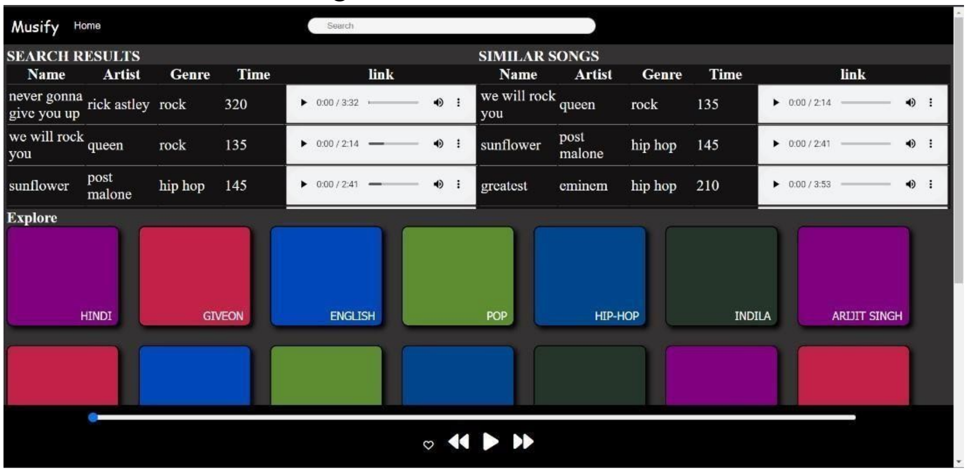
The designing of the Graphical User Interface was done using HTML and CSS. The database was created locally on the PC using MySQL. The database and Front-end of the application were integrated using Java Server Pages (JSP). The Web-application was then hosted on the personal server using Apache Tomcat Server.

# 5. EXPERIMENTAL RESULTS – GUI

The Landing page:
As soon as the web application is launched, the following web page loads which give recommendations to the user based on the past listening history and the popularity of songs. The User can either click on a button to access the particular artist or language or genre, or they can directly search for it using the Search Bar.



On Clicking a button (Search Resut and Recommendation):
When the user clicks on a button, it leads to another page which displays the search results based on the entered string in the form of tables with recommendations in the form of buttons.



SEARCH RESULTS

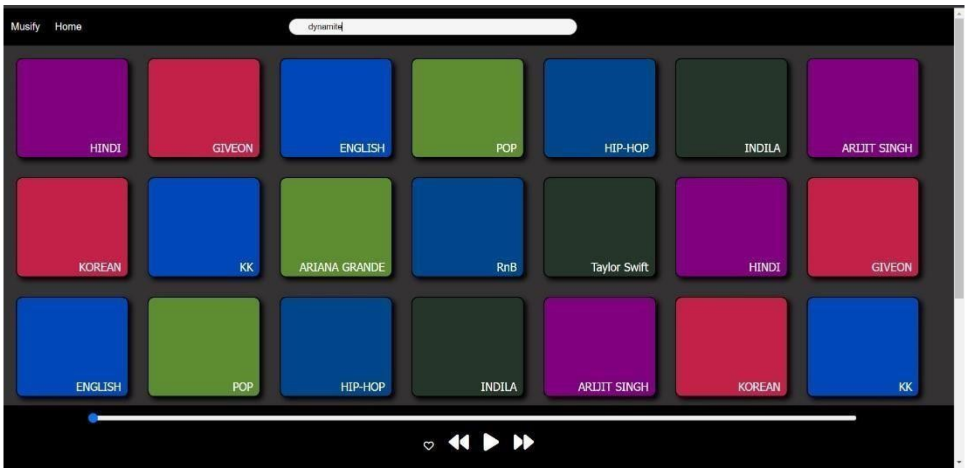
| Name | Artist | Genre | Time | link |
|---|---|---|---|---|
| never gonna give you up | rick astley | rock | 320 | 0:00 / 3:32 |
| we will rock you | queen | rock | 135 | 0:00 / 2:14 |
| sunflower | post malone | hip hop | 145 | 0:00 / 2:41 |

SIMILAR SONGS

| Name | Artist | Genre | Time | link |
|---|---|---|---|---|
| we will rock you | queen | rock | 135 | 0:00 / 2:14 |
| sunflower | post malone | hip hop | 145 | 0:00 / 2:41 |
| greatest | eminem | hip hop | 210 | 0:00 / 3:53 |

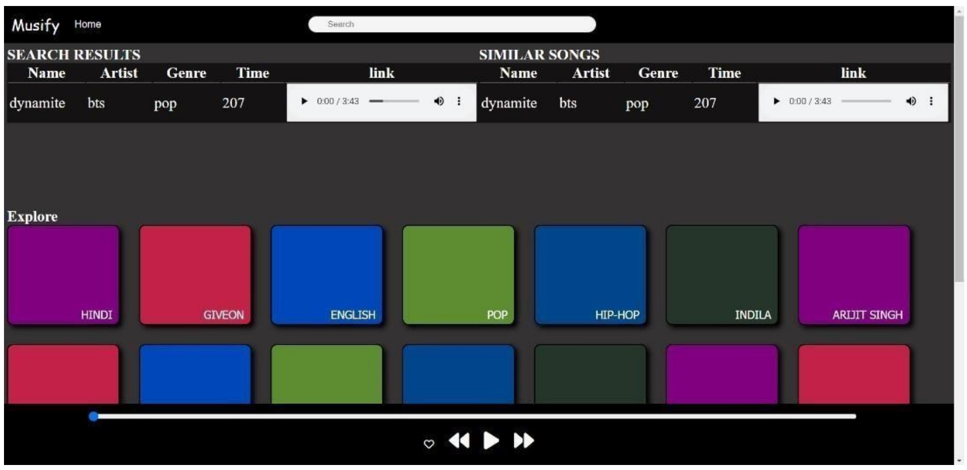
Explore

Entering a String:
When a user enters a string in the search bar, the application compares it to all contents in the database and gives the possible outcomes related to the artist, genre or language.



Search Result & Recommendation

When the user searches for the string, the applications displays all possible results for the search operation and displays them in the form of tables and shows recommendations in the form of tables.



SEARCH RESULTS

| Name | Artist | Genre | Time | link |
|---|---|---|---|---|
| dynamite | bts | pop | 207 | 0:00 / 3:43 |

SIMILAR SONGS

| Name | Artist | Genre | Time | link |
|---|---|---|---|---|
| dynamite | bts | pop | 207 | 0:00 / 3:43 |

Explore

# 6. CONCLUSION AND FUTURE SCOPE

**Conclusion:**

Many people complain that they don’t enjoy songs that music streaming services recommend. Our group decided to explore this problem by designing and implementing an algorithm to recommend music to users based on the songs played on the app . We extracted data from the created our own dataset and added new features from other datasets, removed features that were poorly collected, and synthesized new features from related features. Because artists’ given genre labels do not align well with features of the music, we created our own genres that were based on the users’ needs. Our final recommendations took users’ listening histories and returned songs from the clusters that were closest to the most concentrated areas of songs from those histories. We evaluated the accuracy of our recommendations by excluding some songs from the users’ listening histories that were input to the model and determining whether the model recommended songs from the holdout set. Ultimately, our final model was fairly accurate based on the metrics we defined for measuring success.

**Future Scope:**

The app can be used for recreational purposes as music can help reduce stress and improve memory To continue and improve upon this project, we can collect more informative data about the songs, such as instrumentation, form, and Fourier Transform Spectrums. Additionally, adding more “current” songs would help make the project’s final model more relevant to today’s trends and popular songs. Knowing how long a user listened to a song or whether they replayed it would provide extremely useful information about weighting input songs. For instance, if a user listens to a song for only 30 seconds before deciding that they did not enjoy it, the song still appears in their listening history in the same capacity as a song that they love. Finally, adding a front-end interface or API to make recommendation requests from would make our project deployable for users and developers to work with.

# 7. REFERENCES

1. Jongseol Lee, Kyoungro Yoon, Dalwon Jang, Sei-Jin Jang, Saim Shin, Ji-Hwan Kim. "Music

Recommendation System Based on Genre Distance And User Preference Classification" from the *Journal of Theoretical And Applied Information Technology*. (2018)

2. Puja Deshmukh and Geetanjali Kale. "Music and Movie Recommendation System" from the *International Journal of Engineering Trends and Technology.* (2018)

3. https://play.google.com/store/apps/details?id=media.music.musicplayer (2013) 4. https://github.com/abhishekptiwari/Custom-music-player (2021)

# 8. APPENDIX A – Code Sample

## index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"     pageEncoding="UTF8"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>

<%
            String database = "projectdata";
            String userid = "root";
            String password = "Shansql@007";
            String name = request.getParameter("SearchBar");
            String driver = "com.mysql.jdbc.Driver";
            String connectionUrl = "jdbc:mysql://localhost:3306/";%>


            <%

            try {

                        Class.forName(driver);
                        } catch (ClassNotFoundException e) {
                                    e.printStackTrace();
                        }
                        Connection connection = null;
                        Statement statement = null;
                        ResultSet resultSet = null;
   %>

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Musify</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <nav>
      <ul>
         <li style="font-family:cursive;font-size:x-large;">Musify</li>              <!--navigation bar-->
         <li><a style="text-decoration:none;color:white;" href="/Music_Player/index.html">Home</a></li>
                        <li>
                        <form action="index.jsp">
                       <div class="wrapper" >
                         <input class="search" placeholder="Search" type="text" name="SearchBar">
                      </div>
         </form>
         </li>
      </ul>
   </nav>
```

```
      <div class="containernew">
      <div class="containerTop">
<div class="containerLeft">
          <table class="table">
          <h2>SEARCH RESULTS</h2>
                              <tr>
                              <th>Name</th>
                              <th>Artist</th>
                              <th>Genre</th>
                              <th>Time</th>
                              <th>link</th>
                              </tr>
                              <%
                              try{
                                        connection = DriverManager.getConnection(connectionUrl+database, userid,
                                        password);
                                        statement=connection.createStatement();
                                        String sql ="SELECT * FROM onlytable WHERE songName='"+name+"'
OR artistName = '"+name+"' OR genre= '"+name+"' OR language = '"+name+"'";
          resultSet = statement.executeQuery(sql);
                                        while(resultSet.next()){ %>
                                                  <tr>
                                                  <td><%=resultSet.getString("songName")%></td>
                              <td><%=resultSet.getString("artistName")%></td>
                                                  <td><%=resultSet.getString("genre")%></td>
                                                  <td><%=resultSet.getString("time")%></td>
                                                  <td><button  value="<%=resultSet.getString("songname")%>"
><audio controls ><source src ="<%=resultSet.getString("songName")%>.mp3" type="audio/mp3"
/></audio></button></td>
                                                  </tr>

                                        <%
                                        }
                                        connection.close();
                                        }
                                        catch (Exception e) {
          e.printStackTrace();
                                        }
                                        %>
                              </table>
      </div>
      <div class="containerRight">
          <table class="table">
          <h2>SIMILAR SONGS</h2>
                              <tr>
                              <th>Name</th>
                              <th>Artist</th>
                              <th>Genre</th>
                              <th>Time</th>
                              <th>link</th>
                              </tr>
                              <%
                              try{
                                        connection = DriverManager.getConnection(connectionUrl+database, userid,
password);
                                        statement=connection.createStatement();
                                        String sql ="SELECT * FROM onlytable WHERE songName='"+name+"'
OR artistName = '"+name+"' OR genre= '"+name+"' OR language = '"+name+"' AND like_dislike > 100";
                              resultSet = statement.executeQuery(sql);
                                        while(resultSet.next()){ %>
```

```
                                                    <tr>
                                                    <td><%=resultSet.getString("songName")%></td>
                                                    <td><%=resultSet.getString("artistName")%></td>
                                                    <td><%=resultSet.getString("genre")%></td>
                                                    <td><%=resultSet.getString("time")%></td>
                                                    <td><button  value="<%=resultSet.getString("songname")%>"
><audio controls ><source src ="<%=resultSet.getString("songName")%>.mp3" type="audio/mp3"
/></audio></button></td>
                                                    </tr>

                                            <%
                                            }
                                            connection.close();
                                            }
                                            catch (Exception e) {
        e.printStackTrace();
                                            }
                                            %>
                                    </table>
    </div>
    </div>
    <h2>Explore</h2>
    <div class="containerBottom">
        <form action="displaylang.jsp">
     <div><button style="background-color:purple;" name="langButton" value="Hindi" type="submit"
class="button2">
          <span class="buttontext">HINDI</span>
          </button>
      </div>
      </form>
      <form action="displayartist.jsp">
      <div><button style="background-color:#C32148;" name="artistButton" value="Giveon" type="submit"
class="button2">
          <span class="buttontext">GIVEON</span>
          </button>
      </div>
      </form>
      <form action="displaylang.jsp">
      <div><button style="background-color:#0048BA;" name="langButton" value="English" type="submit"
class="button2">
          <span class="buttontext">ENGLISH</span>
          </button>
      </div>
      </form>
      <form action="displaygenre.jsp">
      <div><button style="background-color:#5E8C31;" name="genreButton" value="Pop" type="submit"
class="button2">
          <span class="buttontext">POP</span>
          </button>
      </div>
      </form>
      <form action="displaygenre.jsp">
      <div><button style="background-color:#00468C;" name="genreButton" value="Hip-hop" type="submit"
class="button2">
          <span class="buttontext">HIP-HOP</span>
          </button>
      </div>
      </form>
      <form action="displayartist.jsp">
```

```
          <div><button style="background-color:#253529;" name="artistButton" value="Indila" type="submit"
class="button2">
              <span class="buttontext">INDILA</span>
             </button>
         </div>
         </form>
         <form action="displayartist.jsp">
          <div><button style="background-color:purple;" name="artistButton" value="Arijit Singh" type="submit"
class="button2">
              <span class="buttontext">ARIJIT SINGH</span>
             </button>
         </div>
         </form>
         <form action="displaylang.jsp">
          <div><button style="background-color:#C32148;" name="langButton" value="Korean" type="submit"
class="button2">
              <span class="buttontext">KOREAN</span>
             </button>
         </div>
         </form>
         <form action="displayartist.jsp">
          <div><button style="background-color:#0048BA;" name="artistButton" value="KK" type="submit"
class="button2">
              <span class="buttontext">KK</span>
             </button>
         </div>
         </form>
         <form action="displayartist.jsp">
         <div><button style="background-color:#5E8C31;" name="artistButton" value="Ariana Grande"
type="submit" class="button2">
              <span class="buttontext">ARIANA GRANDE</span>
             </button>
         </div>
         </form>
         <form action="displaygenre.jsp">
          <div><button style="background-color:#00468C;" name="genreButton" value="R&B" type="submit"
class="button2">
              <span class="buttontext">RnB</span>
             </button>
         </div>
         </form>
         <form action="displayartist.jsp">
          <div><button style="background-color:#253529;" name="artistButton" value="Taylor Swift" type="submit"
class="button2">
              <span class="buttontext">Taylor Swift</span>                  </button>
         </div>
         </form>
         <form action="displaylang.jsp">
          <div><button style="background-color:purple;" name="langButton" value="Hindi" type="submit"
class="button2">
              <span class="buttontext">HINDI</span>
             </button>
         </div>
         </form>
         <form action="displayartist.jsp">
          <div><button style="background-color:#C32148;" name="artistButton" value="Giveon" type="submit"
class="button2">
              <span class="buttontext">GIVEON</span>
             </button>
         </div>
```

```
        </form>
        <form action="displaylang.jsp">
        <div><button style="background-color:#0048BA;" name="langButton" value="English" type="submit"
class="button2">
            <span class="buttontext">ENGLISH</span>
            </button>
        </div>
        </form>
        <form action="displaygenre.jsp">
        <div><button style="background-color:#5E8C31;" name="genreButton" value="Pop" type="submit"
class="button2">
            <span class="buttontext">POP</span>
            </button>
        </div>
        </form>
        <form action="displaygenre.jsp">
        <div><button style="background-color:#00468C;" name="genreButton" value="Hip-hop" type="submit"
class="button2">
            <span class="buttontext">HIP-HOP</span>
            </button>
        </div>
        </form>
        <form action="displayartist.jsp">
        <div><button style="background-color:#253529;" name="artistButton" value="Indila" type="submit"
class="button2">
            <span class="buttontext">INDILA</span>
            </button>
        </div>
        </form>
        <form action="displayartist.jsp">
        <div><button style="background-color:purple;" name="artistButton" value="Arijit Singh" type="submit"
class="button2">
            <span class="buttontext">ARIJIT SINGH</span>
            </button>
        </div>
        </form>
        <form action="displaylang.jsp">
        <div><button style="background-color:#C32148;" name="langButton" value="Korean" type="submit"
class="button2">
            <span class="buttontext">KOREAN</span>
            </button>
        </div>
        </form>
        <form action="displayartist.jsp">
        <div><button style="background-color:#0048BA;" name="artistButton" value="KK" type="submit"
class="button2">
            <span class="buttontext">KK</span>
            </button>
        </div>
        </form>
        <form action="displayartist.jsp">
        <div><button style="background-color:#5E8C31;" name="artistButton" value="Ariana Grande"
type="submit" class="button2">
            <span class="buttontext">ARIANA GRANDE</span>
            </button>
        </div>
        </form>
        <form action="displaygenre.jsp">
        <div><button style="background-color:#00468C;" name="genreButton" value="R&B" type="submit"
class="button2">
```

```
                <span class="buttontext">RnB</span>
                </button>
            </div>
            </form>
            <form action="displayartist.jsp">
            <div><button style="background-color:#253529;" name="artistButton" value="Taylor Swift" type="submit"
class="button2">
                <span class="buttontext">Taylor Swift</span>
                </button>
            </div>
            </form>
        </div>
    </div>

    <div class="bottom">
        <input type="range" name="range" id="myProgressBar" min="0" value="0" max="100">        <!-progress bar-->
        <div class="icons">
            <!--fontawesome icons-->
            <i class="fa-regular fa-heart"></i>
            <i class="fa-2x fa-solid fa-backward" id = "previous"></i>
            <i class="fa-2x fa-solid fa-play" id="masterPlay"></i>                                  <!--buttons-->
            <i class="fa-2x fa-solid fa-forward" id="next"></i>
        </div>
    </div>
    <script src="https://kit.fontawesome.com/31e8ce7dbf.js" ></script>    <script
src="script.js"></script>
</body>

</html>
```