A

PROJECT REPORT

ON

“WEB MUSIC PLAYER- GET THE BEAT”

Submitted By
Govind Singh Mehta

Submitted To

Astt Prof – Kavita Dangwal

(Roll No) 19224512029

Department of Information Technology

INSTITUTE OF TECHNOLOGY & MANAGEMENT

DEHRADUN

CANDIDATE'S DECLARATION

This is to certify that work, which is being presented in the project entitled "MUSIC PLAYER APP – GET THE BEAT" submitted by undersigned students of BCA 6th sem in partial fulfillment for award of degree of Bachelor in Computer Application is a record of their own work carried out by them under the guidance and supervision of Prof. Kavita Dangwal, Asst Prof, Department of Information Technology.

This work has not been submitted elsewhere for the award of any other degree.

Name of student

- **GOVIND SINGH MEHTA (Roll No: 19224512029)**

Date:

Place: ITM Dehradun

INSTITUTE OF TECHNOLOGY & MANAGEMENT

DEHRADUN

CERTIFICATE

This is to certify that the work, which is being presented in the project entitled "MUSIC PLAYER APP – GET THE BEAT" submitted by Govind Singh Mehta student of final year of BCA in partial fulfillment for award of degree of Bachelor of Computer Application is a record of student's work carried out by them under our guidance and supervision.

This work has not been submitted elsewhere for award of any other degree.

Kavita Dangwal (Asst.Prof)

ACKNOWLEDGEMENT

We the students of BCA [Batch: 2019-22] in the Institute of Technology & Management, Dehradun feel glad and believe that we are very lucky to get the opportunity to develop an in house project. It gives us great pleasure to express our sincere gratitude to Kavita Dangwal, the ASST PROF -IT, who helped us in the progress of our project work.

We express our heartfelt gratitude to all the faculties of IT Deptt in ITM, Dehradun for providing all the facilities to complete and make our project very easy and successful. Our deep knowledge of gratitude is extended to Prof. Kavita Dangwal (Project Guide).

DATE:-

PLACE: - ITM, Dehradun

GOVIND SINGH MEHTA

Title of the Project

“Get the Beat” under the category of “MUSIC PLAYER APP” a Web based project.

Abstract

This project is about the Web music player development . The biggest difference between the music player and existing applications is that it is completely free for users to use. It will integrate the advantages of existing music players on the market, as far as possible to mining out the existing music players' function, and then do the filtering in order to eliminate functions that are not practical or low cost-effective.

Also, it will be kept improved based on user feedback. On the other hand, the existing music players pay less attention to all this. Therefore, the music player will solve the limitation by adding required features to make it more user-friendly and humanity.

In a nutshell, the methodology for developing the mp3 music application used in this project is the agile development cycle. The agile development cycle consists of six phases, which is requirements

analysis, planning, design, implementation or development, testing, and deployment. Due to the iterative and flexible nature of this approach, it is able to effectively adapt to users with changing requirements.

Background Information and Motivation

In modern society, people live a fast-paced life, and pressure is constantly present in their lives. Due to the wide use of mobile phones, music has become the daily essential spiritual food, everyone's mobile phone inside there must be a music player. An application like MP3 music players is used to balance stress and happiness. It accompanies people anytime, anywhere and anyplace such as when people are taking the bus and exercising.

The Web music player is designed to allow users to listen to music in a more convenient and comfortable way without too much restriction. Moreover, it can play the music properly without interference from advertisements and offline.

Since many developers realize that modern urbanites are living in a stressful situation, they have captured the commercial opportunity, therefore many similar applications have emerged in the market. These applications have easy-to-use interfaces and features that make the user experience better.

However, these existing music players blindly pursue fancy appearance and huge features, resulting in the high utilization rate of users' mobile phones, such as CPU and memory. Whereas, for most normal users, these kinds of huge and many features are meaningless. Therefore, this project is designed to dedicate to MP3 music player to optimize performance and simplify to meet user needs.

Objectives

The objective of this thesis is to propose development of Web that: Make it with a simple feature and run smoothly. Using this mp3 music player will make users feel comfortable and relaxed because it will pay more attention to the features commonly used by users, excluding some rarely used features that occupy a large of system processors, making the music player lightweight, simple, but also has powerful basic features. A user can skip next or previous songs by simply clicking on the skip left and skip right icon.

Highlight of What Have Been Achieved

The main highlight of the project is to make the proposed become a high learnability webplayer without too many complex features, enhance the interaction between the user and the media control so that the user can have a better experience to achieve real pressure relief. In addition, the ability to enhance the interaction between users and media control is that the application can skip songs by just clicking the next button on the phone under the lock screen status of the phone.

Problem Encountered

The main problem encountered in this project is the structure of code which the structure did not build well at first and led to keep modification during its development. Since the layout interface of each activity is not built as uniformly managed and updated as in the beginning, only one layout is updated and the layout of the other remains unchanged. Fortunately, these problems were solved in the final development stage, but the solution is informal and not efficient, as it is achieved by using a lot of duplicate coding to solve.

Project Achievement

Firstly, the proposed music player had achieved its first objective, which is to make the music player becomes a simple, easy-to-use, and well-run . The proposed application had become faster startup, smaller size, and less memory usage by eliminating some unrealistic features. The application also adds some useful features.

Next, the proposed music player achieves a second objective which is to reduce the use of button controls and enhance the way the app interacts with the user, such as using touch controls. Using gestures, the user doesn't have to pay full attention to the phone, but simply click right or left in the playing interface to switch the songs. In addition, if the app is running on the lock screen or in the background, users can successfully switch to the next song by simply clicking on left or right button on the screen of the phone, completely eliminating the use of buttons.

Lastly, the proposed music player achieves a third objective which is a quick search. The application will use the the alphabet fast scroll, allowing users to quickly traverse the song playlist and find the songs they want, which is an efficient way. For example, if a user wants to search the playlist for a song called "Lemon", he or she just scroll down to the letter l and the result appears. This is because the name of the song contains the characters entered in the alphabetical order

Use Case Description

1. Actor: User Description: User able listen to music on this Web Player Trigger:

A. User select any song from the playlist

B. User click the play button to start playing the music Precondition:

C. The user should have the songs downloaded on their mobile device.

Normal flow of events:

D. User select a song from the playlist

E. The song playing until the end Alternate / Exceptional flows:

F. The application will display a message "No local music" if the playlist does not have a song.

2. Actor: User Description: The progress bar shows the progress of the song Trigger:

A. The progress bar becomes active when the user starts playing the song Precondition:

B. The song must be in playing status Normal flow of events: A. Song is playing

C. The progress bar will be finished if the song is finished playing

Alternate / Exceptional flows:

D. The progress bar will reset if user play other song

E. If the user drags the progress bar to left or right, the song's progress will change

F. The progress bar will reset if finished playing.

System Design

Once the user starts the web player, the first screen will be the home page, reads songs from the local device and generates a playlist.

The user will now choose one of the music from the playlist and start playing that. As the song begins to play, the piano image at the top of the screen will be seen and the button on bottom center will be updated to the pause button. The pause button will be updated to the play button if the song stops playing. In addition, the user can control the progress of the song by dragging the progress bar.

Software Requirement Specification

The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to requirements

Introduction of developing environment

The following is the configuration requirement and installation steps of the Web development environment. The required software of the developing environment : Operation system: Windows XP Linux Windows 7 Software：Visual Studio Code, Google Chrome

The design principle of Music Palyer

Twice the result with half the effort will get an overall study of the principles done before the design and follow them in the operation. The principle of software design mainly includes the following points:

(1) Reliability

The reliability of the software design must be determined. The reliability of the software system refers to the ability to avoid fault occurred in the process of system running, as well as the ability to remedy troubles once the fault occurs.

(2) Reusability

Look for commonness of similar codes, and come up with new methods abstractly and reasonably. Pay attention to the generic design.

(3) Understandability

The understandability of software not only require clear and readable document, but the simplified structure of software itself, which requires the designer to possess keen insight and creativity, and know well about the design objects.

(4) Simple program

To keep the program simple and clear, good programmers can use simple programs to solve complex problems.

(5) Testability

Testability means that the created system has a proper data collection to conduct a comprehensive test of the entire system.

(6) The Open-Closed Principle

Module is extensible but cannot be modified. That is to say, extension is open to the existing code in order to adapt to the new requirements. While modify is closed to the categories. Once the design is completed, the categories cannot be modified

Function and structure design ..

This system adopts the modularized program design, and system function is correspondingly divided into function modules, the main modules include:

(1)UI function module design of terminal: the index screen, play screen, music adding page, file management page are realized.

(2) Backstage function module design of terminal: the specific function, music file data storage function and other functions are implemented.

Design Specifications

1. Methodology
In this project, the agile development cycle will be used to guide the development process.
The agile development cycle contains 6 phrase which is requirement analysis, planning, design, implementation or development, testing, and deployment.

A.Requirement analysis

At this stage, we will review existing MP3 music players on the market. After the review, we will find out what current users need and idea to improve the existing music players and collect their comments and suggestions for further analysis.

B. Planning

In the planning stage, we should first try to explore out the features that the music player can have. Next, we will eliminate the features that users feel no really useful or low cost-effective. Finally, each feature is prioritized and assigned to an iteration.

C. Design

The design stage is prepared according to the requirements of users. Since there are many details and problems encountered during development to be considered for each feature. Therefore, we will discuss and formulate solutions and test strategies to verify the product at this stage.

D.Implementation or Development

During the development phase, we will iteratively implement each of the features listed during the planning phase. At this stage, there will be many setbacks and obstacle, so the team needs to constantly overcome these obstacles. Moreover, we will prioritize the most important features and need to make intelligent trade-offs between the depth of completeness of a single feature and the breadth of implementation of multiple features.

E. Testing

In this stage, we will test the performance of each feature in order to check whether it meets the requirements of users. For example, we will test whether the application can be properly installed and run on a real device, and check whether any errors occur in the running process and each feature is up to standard.

F. Deployment

In this final phase, we will begin to deliver this application to the customer. For instance, we will upload this MP3 music player application in the Google Play Store, or posting download links on Utar Confession which is on Facebook in order to allow students to use it. In addition, we will anticipate that users will encounter unpredictable problems when using the player in this process, so we will solve these problems in a future version.

Tool to use

1. Software Requirement
    A. Visual Studio Code
    B. Google Chrome Browser
2. Hardware Requirement

A.Laptop

- Processor: Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40 GHz
- RAM: 8.00 GB
- Graphic Card: NVIDIA GeForce GT 740M
- Hard Disk storage: 1TB

Functional Requirements:

1. Able to play audio files smoothly
2. Able to listen to the song.
4. Songs in playlists can be quickly searched and filtered through the alphabet.

Non-functional Requirements:

1. Simplify user interface
2. Optimize the design to display the information in a better way.

Implementation

The proposed player completed the debugging task during the testing phase, then it should enter the deployment phase.

due to the number of users are limit so far and the proposed web player is not in the final public version, there are still many modules that should be improved and updated. Therefore, it will be uploaded to the relevant platform to promote to users after the final public version is released.

Below are the steps to describe how a new user will execute the proposed music player:

1. The user first executes the player, he or she needs to give the proposed application the permissions it needs to read local songs on the phone and load them into the song playlist.
2. Users can play a song by clicking on one of the songs on the playlist.
3. In the song playback interface, the user is allowed to drag the progress bar, as well as to perform media control through the icon buttons.

Testing

Unit Testing 1: Music Player
Test Objective: To ensure that the song selected by the user can be played normally, the selected song information is displayed normally, and the song playlist can be import and show properly
Actual Output: Pass

Unit Testing 2: Media Icon Playback Control

Test Objective: To ensure that all playback control icon buttons under playing song interface can work and perform properly

Input: Click the pause button

Expected Output: Stop play the song, the pause button is updated to the play button

Actual Output: Pass

Input: Click the play button

Expected Output: Start play the song and update play the button to pause button

Actual Output: Pass

Input: Click the next button to switch to the next song

Expected Output: Switch to the next song, the song name, album name, and artist name are also successfully updated to the next song's information

Actual Output: Pass

Input: Click the previous button to return to the previous song

Expected Output: Switch to the previous song, the song name, album name, and artist name are also successfully updated to the previous song's information

Actual Output: Pass

CODING OF SOME IMPORTANT PAGES

1.Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Music Player | Get the Beat</title>
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Materi
al+Icons">
</head>
<body>
  <div class="wrapper">
    <div class="top-bar">
      <i class="material-icons">expand_more</i>
      <span>Now Playing</span>
      <i class="material-icons">more_horiz</i>
    </div>
    <div class="img-area">
      <img src="" alt="">
    </div>
    <div class="song-details">
      <p class="name"></p>
      <p class="artist"></p>
    </div>
    <div class="progress-area">
      <div class="progress-bar">
        <audio id="main-audio" src=""></audio>
      </div>
      <div class="song-timer">
        <span class="current-time">0:00</span>
        <span class="max-duration">0:00</span>
      </div>
    </div>
    <div class="controls">
      <i id="repeat-plist" class="material-icons"
title="Playlist looped">repeat</i>
      <i id="prev" class="material-
icons">skip_previous</i>
      <div class="play-pause">
        <i class="material-icons play">play_arrow</i>
      </div>
      <i id="next" class="material-
icons">skip_next</i>
      <i id="more-music" class="material-
icons">queue_music</i>
    </div>
    <div class="music-list">
      <div class="header">
        <div class="row">
          <i class= "list material-
icons">queue_music</i>
          <span>Music list</span>
        </div>
        <i id="close" class="material-
icons">close</i>
      </div>
      <ul>
        <!-- here li list are coming from js -->
      </ul>
    </div>
  </div>
  <script src="js/music-list.js"></script>
  <script src="js/script.js"></script>
</body>
</html>
```

2.style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins
:wght@200;300;400;500;600;700&display=swap');
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
}
*::before, *::after{
  padding: 0;
  margin: 0;
}
:root{
  --pink: #00ecfd;
  --violet: #000000;
  --lightblack: #515C6F;
  --white: #ffffff;
  --darkwhite: #cecaca;
  --pinkshadow: #ffcbdd;
  --lightbshadow: rgba(0,0,0,0.15);
}
body{
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background: linear-gradient(var(--pink) 0%, var(--
violet) 100%);
}
.wrapper{
  width: 380px;
  padding: 25px 30px;
  overflow: hidden;
  position: relative;
  border-radius: 15px;
  background: var(--white);
  box-shadow: 0px 6px 15px var(--lightbshadow);
}
.wrapper i{
  cursor: pointer;
}
.top-bar, .progress-area .song-timer,
.controls, .music-list .header, .music-list ul li{
  display: flex;
```

```
    align-items: center;
    justify-content: space-between;
  }
  .top-bar i{
    font-size: 30px;
    color: var(--lightblack);
  }
  .top-bar i:first-child{
    margin-left: -7px;
  }
  .top-bar span{
    font-size: 18px;
    margin-left: -3px;
    color: var(--lightblack);
  }
  .img-area{
    width: 100%;
    height: 256px;
    overflow: hidden;
    margin-top: 25px;
    border-radius: 15px;
    box-shadow: 0px 6px 12px var(--lightbshadow);
  }
  .img-area img{
    width: 100%;
    height: 100%;
    object-fit: cover;
  }
  .song-details{
    text-align: center;
    margin: 30px 0;
  }
  .song-details p{
    color: var(--lightblack);
  }
  .song-details .name{
    font-size: 21px;
  }
  .song-details .artist{
    font-size: 18px;
    opacity: 0.9;
    line-height: 35px;
  }
  .progress-area{
    height: 6px;
    width: 100%;
    border-radius: 50px;
    background: #f0f0f0;
    cursor: pointer;
  }
```

```
.progress-area .progress-bar{
  height: inherit;
  width: 0%;
  position: relative;
  border-radius: inherit;
  background: linear-gradient(90deg, var(--pink) 0%,
var(--violet) 100%);
}
.progress-bar::before{
  content: "";
  position: absolute;
  height: 12px;
  width: 12px;
  border-radius: 50%;
  top: 50%;
  right: -5px;
  z-index: 2;
  opacity: 0;
  pointer-events: none;
  transform: translateY(-50%);
  background: inherit;
  transition: opacity 0.2s ease;
}
.progress-area:hover .progress-bar::before{
  opacity: 1;
```

```
    pointer-events: auto;
  }
  .progress-area .song-timer{
    margin-top: 2px;
  }
  .song-timer span{
    font-size: 13px;
    color: var(--lightblack);
  }
  .controls{
    margin: 40px 0 5px 0;
  }
  .controls i{
    font-size: 28px;
    user-select: none;
    background: linear-gradient(var(--pink) 0%, var(--
  violet) 100%);
    background-clip: text;
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
  }
  .controls i:nth-child(2),
  .controls i:nth-child(4){
    font-size: 43px;
  }
```

```
.controls #prev{
  margin-right: -13px;
}
.controls #next{
  margin-left: -13px;
}
.controls .play-pause{
  height: 54px;
  width: 54px;
  display: flex;
  cursor: pointer;
  align-items: center;
  justify-content: center;
  border-radius: 50%;
  background: linear-gradient(var(--white) 0%, var(--
darkwhite) 100%);
  box-shadow: 0px 0px 5px var(--pink);
}
.play-pause::before{
  position: absolute;
  content: "";
  height: 43px;
  width: 43px;
  border-radius: inherit;
  background: linear-gradient(var(--pink) 0%, var(--
violet) 100%);
}
.play-pause i{
  height: 43px;
  width: 43px;
  line-height: 43px;
  text-align: center;
  background: inherit;
  background-clip: text;
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
  position: absolute;
}

.music-list{
  position: absolute;
  background: var(--white);
  width: 100%;
  left: 0;
  bottom: -55%;
  opacity: 0;
  pointer-events: none;
  z-index: 5;
  padding: 15px 30px;
```

```
    border-radius: 15px;
    box-shadow: 0px -5px 10px rgba(0,0,0,0.1);
    transition: all 0.15s ease-out;
  }
  .music-list.show{
    bottom: 0;
    opacity: 1;
    pointer-events: auto;
  }
  .header .row{
    display: flex;
    align-items: center;
    font-size: 19px;
    color: var(--lightblack);
  }
  .header .row i{
    cursor: default;
  }
  .header .row span{
    margin-left: 5px;
  }
  .header #close{
    font-size: 22px;
    color: var(--lightblack);
  }
```

```
    .music-list ul{
      margin: 10px 0;
      max-height: 260px;
      overflow: auto;
    }
    .music-list ul::-webkit-scrollbar{
      width: 0px;
    }
    .music-list ul li{
      list-style: none;
      display: flex;
      cursor: pointer;
      padding-bottom: 10px;
      margin-bottom: 5px;
      color: var(--lightblack);
      border-bottom: 1px solid #E5E5E5;
    }
    .music-list ul li:last-child{
      border-bottom: 0px;
    }
    .music-list ul li .row span{
      font-size: 17px;
    }
    .music-list ul li .row p{
      opacity: 0.9;
```

```
}
ul li .audio-duration{
  font-size: 16px;
}
ul li.playing{
  pointer-events: none;
  color: var(--violet);
}
```

3.script.js

```
const wrapper = document.querySelector(".wrapper"),
musicImg = wrapper.querySelector(".img-area img"),
musicName = wrapper.querySelector(".song-details
.name"),
musicArtist = wrapper.querySelector(".song-details
.artist"),
playPauseBtn = wrapper.querySelector(".play-pause"),
prevBtn = wrapper.querySelector("#prev"),
nextBtn = wrapper.querySelector("#next"),
mainAudio = wrapper.querySelector("#main-audio"),
progressArea = wrapper.querySelector(".progress-
area"),
progressBar = progressArea.querySelector(".progress-
bar"),
musicList = wrapper.querySelector(".music-list"),
moreMusicBtn = wrapper.querySelector("#more-music"),
closemoreMusic = musicList.querySelector("#close");

let musicIndex = Math.floor((Math.random() *
allMusic.length) + 1);
isMusicPaused = true;

window.addEventListener("load", ()=>{
  loadMusic(musicIndex);
  playingSong();
});

function loadMusic(indexNumb){
  musicName.innerText = allMusic[indexNumb - 1].name;
  musicArtist.innerText = allMusic[indexNumb -
1].artist;
  musicImg.src = `images/${allMusic[indexNumb -
1].src}.jpg`;
  mainAudio.src = `songs/${allMusic[indexNumb -
1].src}.mp3`;
}
```

```
//play music function
function playMusic(){
  wrapper.classList.add("paused");
  playPauseBtn.querySelector("i").innerText =
"pause";
  mainAudio.play();
}

//pause music function
function pauseMusic(){
  wrapper.classList.remove("paused");
  playPauseBtn.querySelector("i").innerText =
"play_arrow";
  mainAudio.pause();
}

//prev music function
function prevMusic(){
  musicIndex--; //decrement of musicIndex by 1
  //if musicIndex is less than 1 then musicIndex will
be the array length so the last music play
  musicIndex < 1 ? musicIndex = allMusic.length :
musicIndex = musicIndex;
  loadMusic(musicIndex);
  playMusic();
  playingSong();
}

//next music function
function nextMusic(){
  musicIndex++; //increment of musicIndex by 1
  //if musicIndex is greater than array length then
musicIndex will be 1 so the first music play
  musicIndex > allMusic.length ? musicIndex = 1 :
musicIndex = musicIndex;
  loadMusic(musicIndex);
  playMusic();
  playingSong();
}

// play or pause button event
playPauseBtn.addEventListener("click", ()=>{
  const isMusicPlay =
wrapper.classList.contains("paused");
  //if isPlayMusic is true then call pauseMusic else
call playMusic
  isMusicPlay ? pauseMusic() : playMusic();
  playingSong();
});
```

```
//prev music button event
prevBtn.addEventListener("click", ()=>{
  prevMusic();
});

//next music button event
nextBtn.addEventListener("click", ()=>{
  nextMusic();
});

// update progress bar width according to music
current time
mainAudio.addEventListener("timeupdate", (e)=>{
  const currentTime = e.target.currentTime; //getting
playing song currentTime
  const duration = e.target.duration; //getting
playing song total duration
  let progressWidth = (currentTime / duration) * 100;
  progressBar.style.width = `${progressWidth}%`;

  let musicCurrentTime =
wrapper.querySelector(".current-time"),
  musicDuartion = wrapper.querySelector(".max-
duration");
  mainAudio.addEventListener("loadeddata", ()=>{
    // update song total duration
    let mainAdDuration = mainAudio.duration;
    let totalMin = Math.floor(mainAdDuration / 60);
    let totalSec = Math.floor(mainAdDuration % 60);
    if(totalSec < 10){ //if sec is less than 10 then
add 0 before it
      totalSec = `0${totalSec}`;
    }
    musicDuartion.innerText =
`${totalMin}:${totalSec}`;
  });
  // update playing song current time
  let currentMin = Math.floor(currentTime / 60);
  let currentSec = Math.floor(currentTime % 60);
  if(currentSec < 10){ //if sec is less than 10 then
add 0 before it
    currentSec = `0${currentSec}`;
  }
  musicCurrentTime.innerText =
`${currentMin}:${currentSec}`;
});

// update playing song currentTime on according to
the progress bar width
progressArea.addEventListener("click", (e)=>{
  let progressWidth = progressArea.clientWidth;
//getting width of progress bar
  let clickedOffsetX = e.offsetX; //getting offset x
value
  let songDuration = mainAudio.duration; //getting
song total duration

  mainAudio.currentTime = (clickedOffsetX /
progressWidth) * songDuration;
  playMusic(); //calling playMusic function
  playingSong();
});

//change loop, shuffle, repeat icon onclick
const repeatBtn = wrapper.querySelector("#repeat-
plist");
repeatBtn.addEventListener("click", ()=>{
  let getText = repeatBtn.innerText; //getting this
tag innerText
  switch(getText){
    case "repeat":
      repeatBtn.innerText = "repeat_one";
      repeatBtn.setAttribute("title", "Song looped");
      break;
    case "repeat_one":
      repeatBtn.innerText = "shuffle";
      repeatBtn.setAttribute("title", "Playback
shuffled");
      break;
    case "shuffle":
      repeatBtn.innerText = "repeat";
      repeatBtn.setAttribute("title", "Playlist
looped");
      break;
  }
});

//code for what to do after song ended
mainAudio.addEventListener("ended", ()=>{
  // we'll do according to the icon means if user has
set icon to
  // loop song then we'll repeat the current song and
will do accordingly
  let getText = repeatBtn.innerText; //getting this
tag innerText
  switch(getText){
    case "repeat":
      nextMusic(); //calling nextMusic function
      break;
    case "repeat_one":
      mainAudio.currentTime = 0; //setting audio
current time to 0
      loadMusic(musicIndex); //calling loadMusic
function with argument, in the argument there is a
index of current song
      playMusic(); //calling playMusic function
      break;
    case "shuffle":
      let randIndex = Math.floor((Math.random() *
allMusic.length) + 1); //genereting random index/numb
with max range of array length
      do{
        randIndex = Math.floor((Math.random() *
allMusic.length) + 1);
      }while(musicIndex == randIndex); //this loop
run until the next random number won't be the same of
current musicIndex
      musicIndex = randIndex; //passing randomIndex
to musicIndex
      loadMusic(musicIndex);
      playMusic();
      playingSong();
      break;
  }
});
```

```
//show music list onclick of music icon
moreMusicBtn.addEventListener("click", ()=>{
  musicList.classList.toggle("show");
});
closemoreMusic.addEventListener("click", ()=>{
  moreMusicBtn.click();
});

const ulTag = wrapper.querySelector("ul");
// let create li tags according to array length for
list
for (let i = 0; i < allMusic.length; i++) {
  //let's pass the song name, artist from the array
  let liTag = `<li li-index="${i + 1}">
                <div class="row">
                  <span>${allMusic[i].name}</span>
                  <p>${allMusic[i].artist}</p>
                </div>
                <span id="${allMusic[i].src}"
class="audio-duration">3:40</span>
                <audio class="${allMusic[i].src}"
src="songs/${allMusic[i].src}.mp3"></audio>
              </li>`;
  ulTag.insertAdjacentHTML("beforeend", liTag);
//inserting the li inside ul tag

  let liAudioDuartionTag =
ulTag.querySelector(`#${allMusic[i].src}`);
  let liAudioTag =
ulTag.querySelector(`.${allMusic[i].src}`);
  liAudioTag.addEventListener("loadeddata", ()=>{
    let duration = liAudioTag.duration;
    let totalMin = Math.floor(duration / 60);
    let totalSec = Math.floor(duration % 60);
    if(totalSec < 10){ //if sec is less than 10 then
add 0 before it
      totalSec = `0${totalSec}`;
    };
    liAudioDuartionTag.innerText =
`${totalMin}:${totalSec}`; //passing total duation of
song
    liAudioDuartionTag.setAttribute("t-duration",
`${totalMin}:${totalSec}`); //adding t-duration
attribute with total duration value
  });
}
```

```
//play particular song from the list onclick of li
tag
function playingSong(){
  const allLiTag = ulTag.querySelectorAll("li");

  for (let j = 0; j < allLiTag.length; j++) {
    let audioTag = allLiTag[j].querySelector(".audio-
duration");

    if(allLiTag[j].classList.contains("playing")){
      allLiTag[j].classList.remove("playing");
      let adDuration = audioTag.getAttribute("t-
duration");
      audioTag.innerText = adDuration;
    }

    //if the li tag index is equal to the musicIndex
then add playing class in it
    if(allLiTag[j].getAttribute("li-index") ==
musicIndex){
      allLiTag[j].classList.add("playing");
      audioTag.innerText = "Playing";
    }
    allLiTag[j].setAttribute("onclick",
"clicked(this)");
  }
}

//particular li clicked function
function clicked(element){
  let getLiIndex = element.getAttribute("li-index");
  musicIndex = getLiIndex; //updating current song
index with clicked li index
  loadMusic(musicIndex);
  playMusic();
  playingSong();
}
```

5.music-list.js

```
let allMusic = [
    {
      name: "Harley Bird - Home",
      artist: "Jordan Schor",
      img: "music-1",
      src: "music-1"
    },
    {
```

```
    name: "Ikson Anywhere – Ikson",
    artist: "Audio Library",
    img: "music-2",
    src: "music-2"
  },
  {
    name: "Beauz & Jvna - Crazy",
    artist: "Beauz & Jvna",
    img: "music-3",
    src: "music-3"
  },
  {
    name: "Hardwind - Want Me",
    artist: "Mike Archangelo",
    img: "music-4",
    src: "music-4"
  },
  {
    name: "Jim - Sun Goes Down",
    artist: "Jim Yosef x Roy",
    img: "music-5",
    src: "music-5"
  },
  {
    name: "Lost Sky - Vision NCS",
    artist: "NCS Release",
    img: "music-6",
    src: "music-6"
  },
];
```

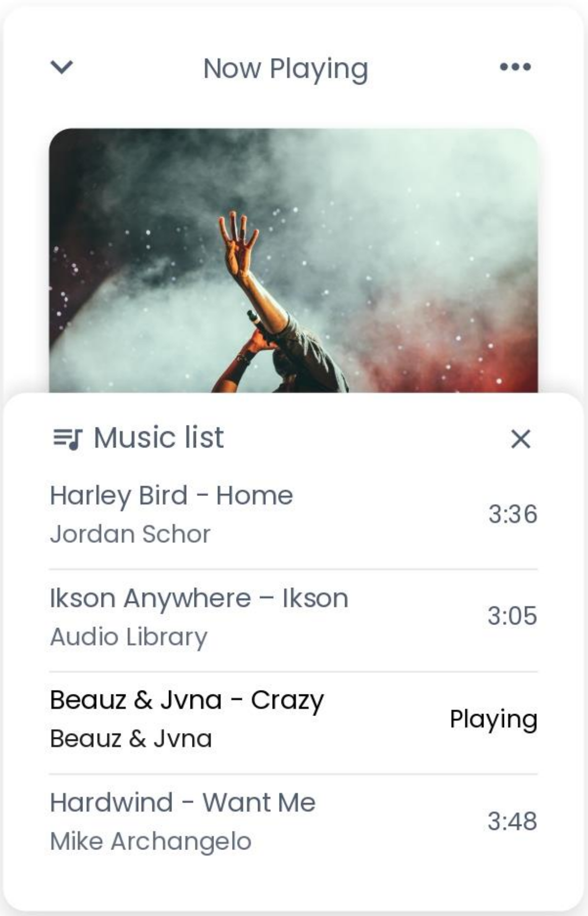
SCREENSHOT

Now Playing

Beauz & Jvna - Crazy

Beauz & Jvna

2:02 3:08



Harley Bird - Home Jordan Schor	3:36
Ikson Anywhere – Ikson Audio Library	3:05
Beauz & Jvna - Crazy Beauz & Jvna	Playing
Hardwind - Want Me Mike Archangelo	3:48

Conclusion

In a nutshell, when users hold the mentality of venting and relaxation to expect the music player to bring them relief pressure, in result the application with a dazzling and complex interface, a variety of multifarious functions, from time to time prompt out of the advertising, as well as the function that requires be a members to use, which will only make users feel more depressed and feel the pressure. Moreover, most people who use a music player, usually don't leave the music player open in the foreground, but start playing music and then go on to do something else at hand such as take a break, read a book and news, or play a game. As a result, they can't focus on the various functions and buttons in the app's interface. For instance, users who are lying down to take a break and try to switch to the next song but they need lots of action like unlocking the phone, opening the app again in the background and looking for the switch button. In addition, the specific song is overwhelmed by a large number of songs and causes information overload, users can only spend more energy and time to find it. For example, searching for a book in the library, and realizing that there is no library catalog is mean to looking for a needle in a haystack. In short, the proposed application will combine the strengths of most music players on the existing market and eliminate some unrealistic features, allowing users to focus on listening to music rather than store, communities or various VIP packages or features. The proposed MP3 music player will focus on improving the experience of users of the music player experience