An Industry-Oriented Mini Project

"MUSIC PLAYER SYSTEM"

Submitted in Partial Fulfillment of the Academic Requirement for the Award of Degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted

By

K. ANUSHA	**(17R01A0535)**
NAMPALLY PALLAVI VARMA	**(17R01A0539)**
R PRANEETHA	**(17R01A0545)**
SK.BAHIJAH	**(17R01A0549)**

Under the esteemed guidance of

MR.A.PRAKASH

(Head of Department)

CMR INSTITUTE OF TECHNOLOGY

(Approved by AICTE, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, R.R. Dist., Hyderabad.

2020-21

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad) Kandlakoya, Medchal Road, Hyderabad.

CERTIFICATE

This is to certify that an Industry oriented Mini Project entitled with: "DOPPELGANGER BASED APPRAISAL FOR REAL ESTATE PROPERTIES" is being

Submitted By

K. ANUSHA	(17R01A0535)
NAMPALLY PALLAVI VARMA	(17R01A0539)
R PRANEETHA	(17R01A0545)
SK.BAHIJAH	(17R01A0549)

In partial fulfillment of the requirement for award of the degree of B. Tech in CSE to the JNTUH, Hyderabad is a record of a bonafide work carried out under our guidance and supervision.

The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University for award of any other degree or diploma.

Signature of Guide	**Signature of Coordinator**	**Signature of HOD**
Mr.A.Prakash	**Dr.S.Alagumuthukrishnan**	
(Head of Department)	(Associate Professor)	

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, **Dr. B. Satyanarayana**, **Principal** and **A.Prakash**, **Head of Department**, Dept of Computer Science and Engineering, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to **Mr.A. Prakash**, **Mini Project Coordinator** and Dept of Computer Science and Engineering, CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

K. ANUSHA (17R01A0535)
NAMPALLY PALLAVI VARMA (17R01A0539)
R PRANEETHA (17R01A0545)
SK.BAHIJAH (17R01A0549)

ABSTRACT

In order to solve the problem of complex functions and large required memory of mobile phone music player on the current market, a new music player of simple, convenient, less required memory as well as user-friendly is developed. In this paper, we propose a mini music player with a clean user interface purely implemented using HTML, CSS and JavaScript which lead to design and coding of music player. The new design mainly realizes limited core functions including play, pause, volume control with big bold texts highlighted which are used to show the title track playing.It supports audio files and allow users to enjoy music in any browser.

TABLE OF CONTENTS

ACKNOWLEDGEMENT **I**

ABSTRACT **II**

LIST OF FIGURES **III**

LIST OF TABLES **IV**

LIST OF SCREENSHOTS **IV**

1.INTRODUCTION

1.1 OVERVIEW 1

1.1.1 HTML 1

1.1.2 CSS 2

1.1.3 JAVA SCRIPT 4

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM 5

2.2 PROPOSED SYSTEM 5

3. REQUIREMENTS

3.1 REQUIREMENT ANALYSIS 6

3.2 REQUIREMENT ANALYSIS IN THIS PROJECT 7

3.2.1 PROBLEM RECOGNITION 7

3.3 SPECIFICATION PRINCIPLES 7

3.3.1 FUNCTIONAL REQUIREMENTS 7

3.3.2 SOFTWARE REQUIREMENTS 7

3.3.3 HARDWARE REQUIREMENTS 7

4. SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM 8
4.2 DATA FLOW DIAGRAM 8
5. MODULES
5.1 PLAY MUSIC 9
5.2 PAUSE MUSIC 9
5.3 CHANGE AND ALTER SONG 10
5.4 VOLUME CONTROL 10
6. IMPLEMENTATION
6.1 HTML LAYOUT 11
6.2 CSS STYLING 11
6.3 JAVASCRIPT LOGIC OF THE PLAYER 13
7. SCREENSHOTS
8. SYSTEM TEST
8.1 UNIT TEST 18
8.2 INTEGRATION TEST 18
8.3 FUNCTIONAL TEST 18
8.4 WHITEBOX TESTING 19
8.5 BLACKBOX TESTING 19
8.6 TESTING STRATEGY AND APPROACH 19
8.7 TEST REPORT 20
9. CONCLUSION 21
10 BIBLIOGRAPHY 22

LIST OF FIGURES

FIG.NO	PARTICULARS	PAGE NO.
4.1.1	Architecture Diagram	8
4.2.1	Data Flow Diagram	8
5.1.1	Play Music	9
5.2.1	Pause Music	10
5.3.1	Seek Slider	10
5.4.1	Volume Slider	10

LIST OF TABLES

TABLE NO	TABLE PARTICULARS	PAGE NO
8.7	Test Report	20

LIST OF SCREENSHOTS

SCREENSHOT NO.	PARTICULARS	PAGE NO.
7.1	Music Player	15
7.2	Playing Song	15
7.3	Playing next song	16
7.4	Mobile Screenshots	17

1. INTRODUCTION

As streaming is increasingly being adopted by users, online media players have become essential for consuming media on the internet. Music players allow one to enjoy music in any browser and supports a lot of the features of an offline music player.

We will be creating a music player with a clean user interface that can be used to play music in the browser. We will also implement features like seeking and volume control. HTML has several methods in the HTMLMediaElement interface that can be used to play audio files and control its playback without using any other library.

We will start by creating the HTML layout first that defines the structure of the player, make it look good by styling using CSS and then write the player logic for all the functions in JavaScript.

1.1 OVERVIEW

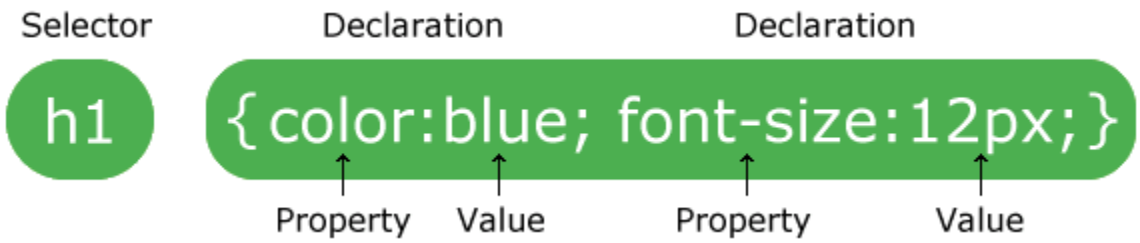
Music player is a system which can run over browsers directly rather than downloading an application.

1.1.1 HTML

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.
- The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.
- It must only appear once, at the top of the page (before any HTML tags).
- The <!DOCTYPE> declaration is not case sensitive.
- The tag defines a division or a section in an HTML document.
- The tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.
- The tag is easily styled by using the class or id attribute.

1.1.2 CSS - CASCADING STYLE SHEETS

- CSS is the language we use to style an HTML document.
- CSS describes how HTML elements should be displayed.
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files.**
- CSS SYNTAX: A CSS rule-set consists of a selector and a declaration block:



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.
- CSS selectors are used to "find" (or select) the HTML elements you want to style.
- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)
- There are three ways of inserting a style sheet:
- External CSS
- Internal CSS

- Inline CSS

1.1.3 JAVASCRIPT

- JavaScript is the world's most popular programming language.
- JavaScript is the programming language of the Web.
- JavaScript is easy to learn.
- JavaScript is one of the **3 languages** all web developers **must** learn:
- 1. **HTML** to define the content of web pages
- 2. **CSS** to specify the layout of web pages
- 3. **JavaScript** to program the behavior of web page.
- JavaScript can "display" data in different ways:
- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().
- To access an HTML element, JavaScript can use the document.getElementById(id) method.
- The id attribute defines the HTML element. The innerHTML property defines the HTML content.
- A **computer program** is a list of "instructions" to be "executed" by a computer.
- In a programming language, these programming instructions are called **statements**.
- A **JavaScript program** is a list of programming **statements**.
- In HTML, JavaScript programs are executed by the web browser.
- JavaScript statements are composed of:
- Values, Operators, Expressions, Keywords, and Comments.
- This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":
- JavaScript programs (and JavaScript statements) are often called JavaScript code.

- JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- The purpose of code blocks is to define statements to be executed together.
- The JavaScript syntax defines two types of values:
- Fixed values
- Variable values
- Fixed values are called **Literals**
- Variable values are called **Variables**.
- In a programming language, **variables** are used to **store** data values.
- JavaScript uses the var keyword to **declare** variables.
- An **equal sign** is used to **assign values** to variables.

2.SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

As streaming is increasingly being adopted by users, online media players have become essential for consuming media on the internet.

Music players allow one to enjoy music in any browser and supports a lot of the features of an music player.

Usually if we want to listen to some music, we need to download an app and then get the songs what we wanted.

It works only in online and if we want to listen in offline we need to download the required songs.

2.2 PROPOSED SYSTEM

Creating a music player with a clean user interface that can be used to play music in the browser. We will also implement features like seeking and volume control.
The songs are directly available in browser and we can listen to them easily. It does not require any app.

Advantages:

- It offers low data size.
- The smaller file size enables the user to rip a large amount of music files.
- The compression ratio is not fixed.

Disadvantages:

- There is no proper sorting of songs.
- Searching of a song is not possible here.

3. REQUIREMENTS

The requirement phase basically consists of three phases:

- Requirement Analysis
- Requirement Specification
- Requirement Validation

3.1 REQUIREMENT ANALYSIS

Requirement Analysis is a software engineering task that bridges the gap between system level software allocation and software design. It provides the system engineer to specify software function and performance, indicate software's interface with the other system elements and establish constraints that software must meet.

The basic aim of this stage is to obtain a clear picture of the needs and requirements of the end-user and also the organization. Analysis involves interaction between the clients and the analysis. Usually analysts research a problem by asking questions and reading existing documents. The analysts have to uncover the real needs of the user even if they don't know them clearly. During analysis it is essential that a complete and consistent set of specifications emerge for the system. Here it is essential to resolve the contradictions that could emerge from the information got from various parties. This is essential to ensure that the final specifications are consistent.

It maybe be divided into 5 areas of effort.

- Problem recognition
- Evaluation and synthesis
- Modeling
- Specification
- Review

Each requirement analysis method has a unique point of view. However, all analysis methods are related by a set of operational principles.
They are

- The information domain of the problem must be represented understood.
- The functions that the software is to perform must be defined.
- The behavior of the software as a consequence of external events must be defined.
- The models that depict information function and behavior must be partitioned in a hierarchical or layered fashion.
- The analysis process must move from essential information to Implementation detail.

3.2 REQUIREMENT ANALYSIS IN THIS PROJECT

The main aim in this stage is to assess what kind of a system would be suitable for a problem and how to build it. The requirements of this system can be defined by going through the existing system and its problems. This helps in building new system and their expectations from it. The steps involved would be

3.2.1. PROBLEM RECOGNITION

This stage is used to determine the problem.

3.3 SPECIFICATION PRINCIPLES

Software Requirements Specification plays an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation.
Requirements may be specified in a variety of ways. However, there are some guidelines worth following:

- Representation format and content should be relevant to the problem.
- Information contained within the specification should be nested.
- Diagrams and other notational forms should be restricted in number and consistent in use.
- Representations should be revisable.

3.3.1 FUNCTIONAL REQUIREMENTS

- Graphical User Interface with the user.

3.3.2 SOFTWARE REQUIREMENTS

- HTML & CSS
- JavaScript

OS supported:

- Windows 7
- Windows 8
- Windows 10

3.3.3 HARDWARE REQUIREMENTS

- System : PentiumIV 2.4
- Hard Disk : 40 GB.
- RAM : 512 Mb.

4. SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM

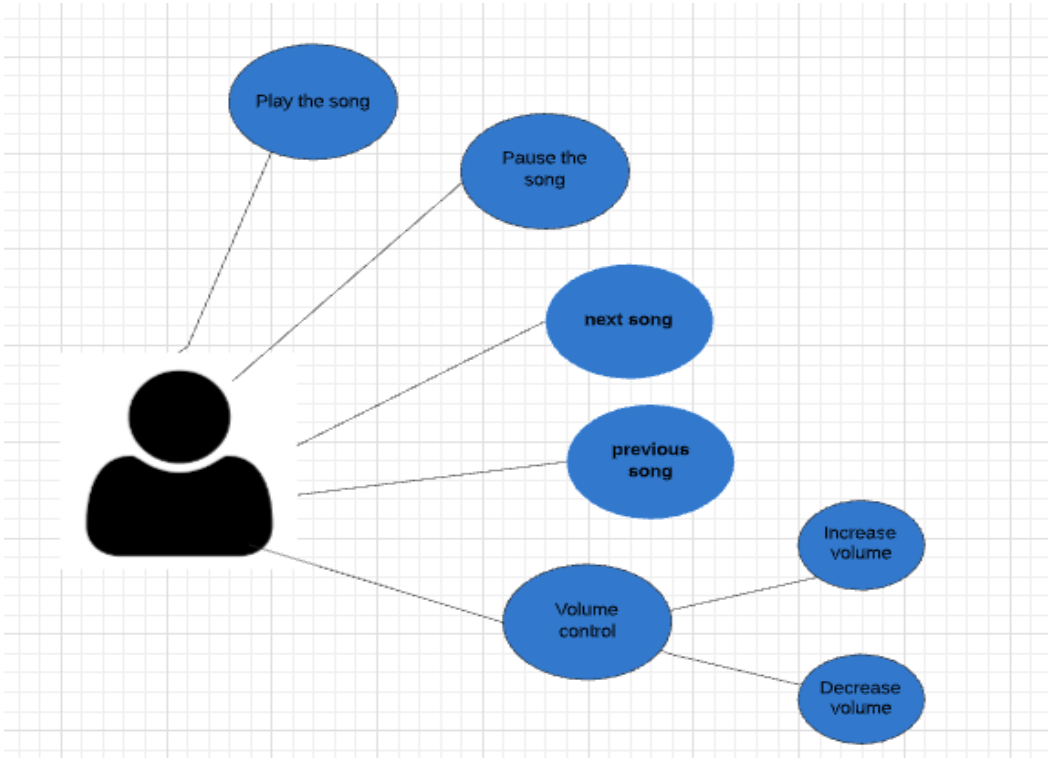


FIG 4.1.1 Architecture diagram

4.2 DATA FLOW DIAGRAM

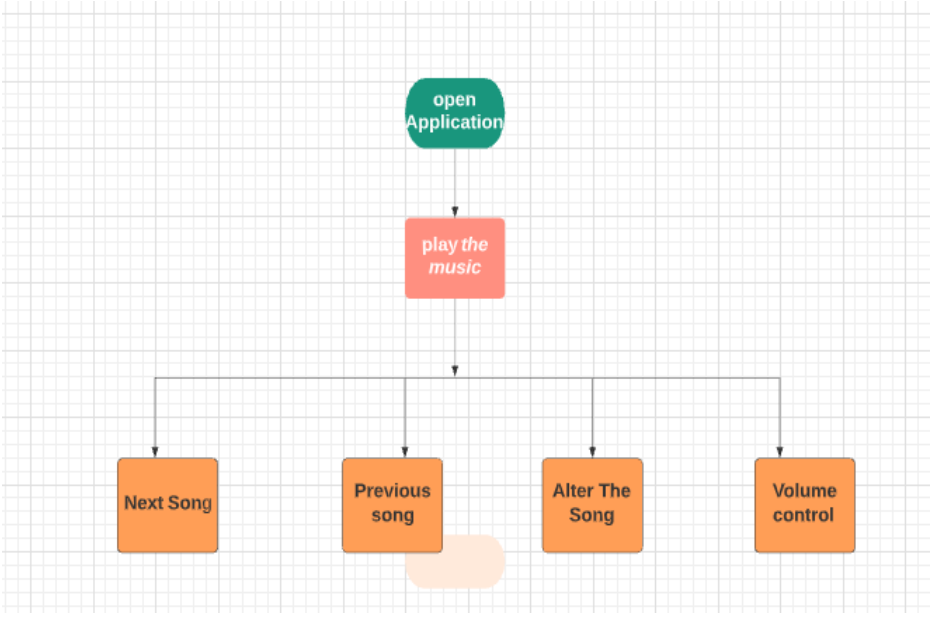


FIG 4.2.1 Flow Chart

5. MODULES

There are four modules:

- Play music
- Pause music
- Change and Alter song
- Volume control

MODULES DESCRIPTION:

5.1 PLAY MUSIC

All the tracks that have to be played are defined in the tracklist as objects. These objects contain properties like the name, artist, image and path to the track. Each of the tracks can then be accessed using its track index

Loading the track:

The audio element is assigned a new source using its src property. It may be given any path from the filesystem or a URL. The load() method is then used on the audio element to get the track ready.A function playTrack() handles the playing of the currently loaded track.
The play() method of the HTMLMediaElement API is used for this function. The icon of the button also changes to the pause icon. This is done by using one of the icons from the FontAwesome library and inserting it using innerHTML.

FIG 5.1.1 Play music

5.2 PAUSE MUSIC

A function pauseTrack() handles the playing of the currently loaded track.
The pause() method of the HTMLMediaElement API is used for this function. The icon of the button also changes back to the play icon. This is done by using one of the icons from the FontAwesome library and inserting it using innerHTML.These two functions are invoked depending on whether the track is currently playing or not. The playpause() function handles the actual play/pause control of the track.

FIG 5.2.1 Pause Music

5.3 CHANGE AND ALTER SONG

A function prevTrack() handles the loading of the previous track and moving the index backward. The index is reset to the last track when the index reaches the first track. The loadTrack() method defined above is used for loading the new track. A function nextTrack() handles the loading of the next track and moving the index forward. The index is reset to the first track when the index reaches the last track. The loadTrack() method defined above is used for loading the new track.

Seek slider

The seek slider shows the current playback position on a slider by updating it with the current time of the track. A new function is created seekUpdate() which handles the updating of the seek slider relative to the current time of the track. The seek slider position is calculated and set using the value property.

FIG 5.3.1 seek slider

5.4VOLUME CONTROLLER

To increase and decrease the volume of the song being played we use volume slider which helps to control the volume.

The volume slider:

The volume slider is used to display an set the current volume of the track. A new function created setVolume() which handles the setting of the volume slider whenever the user changes it.

FIG 5.4.1 Volume Slider

6. IMPLEMENTATION

Technologies used in these project are

- HTML
- CSS
- JavaScript

6.1 HTML Layout

The HTML layout defines the element structure that would be shown on the page. The player can be divided into the following portions:

Details Portion: This section shows the details of the current track being played. It includes the track number, track album, track name and track artist.

Buttons Portion: This section shows the buttons that are used to control the playback of the track. It includes the play/pause button, the previous and next track buttons. They would have an onclick() method that calls a specific function defined in the JavaScript file.

Sliders Portion: This section contains the seek slider and volume slider that can be used to control the playback and volume.

We will be using FontAwesome icons to get the icons for all the buttons used on the page. The custom CSS and JavaScript we will write later is also linked in the file.

6.2 CSS STYLING

Using CSS we can style the different portions to make it more visually appealing:

The flex layout is used to arrange the various elements of the player and align them to the middle of the page.

The track art image is given a fixed dimension and made rounded using the border-radius property.

The two sliders have been modified from their default look by using the appearance property. The height and background are changed to suit the color scheme. They are also given slight transparency that smoothly transitions to the full opacity using the transition property.

All the playback controls have their cursor property set so that it changes to a pointer whenever the mouse hovers over it.filter_none.

6.3 JAVASCRIPT LOGIC OF THE PLAYER:

The logic of the player is defined in the JavaScript file. There are several functions that work together to handle all the functions of the player.

Step 1: Defining all the variables and accessing the HTML elements
The required elements in the HTML layout that are to be dynamically changed are first selected using the querySelector() method. They are then assigned variable names so that they could be accessed and modified. Other variables that would be accessed throughout the program are also defined.

Step 2: Loading a new track from the tracklist

All the tracks that have to be played are defined in the tracklist as objects. These objects contain properties like the name, artist, image and path to the track. Each of the tracks can then be accessed using its track index.

To load a track, a function loadTrack() is defined which handles the following things:

Reset all the values of the previous track
A resetValues() function is created which handles the resetting of the duration value and the slider to their initial values before a new track starts. This prevents the jumping of the seek slider while the new track loads.

Loading the track
The audio element is assigned a new source using its src property. It may be given any path from the filesystem or a URL. The load() method is then used on the audio element to get the track ready.

Updating the track art to be shown
The track art is fetched from the array and assigned with the help of
the backgroundImage property.

Updating the track details to be shown
The track details are fetched from the array and assigned with the help of the textContent property.

Adding event listeners to the track
The media element has two event listeners added to it, the first one to update the current seek position and the second one to load the next track when the current track finishes.

Setting a random colored background
A coloured background is generated by randomising the red, green and blue values used and setting it as a color. The effect is animated by using the transition property on the background-color.

Step 3: Configuring the player buttons

A function playTrack() handles the playing of the currently loaded track. The play() method of the HTMLMediaElement API is used for this function. The icon of the button also changes to the pause icon. This is done by using one of the icons from the FontAwesome library and inserting it using innerHTML.

A function pauseTrack() handles the playing of the currently loaded track. The pause() method of the HTMLMediaElement API is used for this function. The icon of the button also changes back to the play icon. This is done by using one of the icons from the FontAwesome library and inserting it using innerHTML.

These two functions are invoked depending on whether the track is currently playing or not. The playpause() function handles the actual play/pause control of the track.

A function prevTrack() handles the loading of the previous track and moving the index backward. The index is reset to the last track when the index reaches the first track. The loadTrack() method defined above is used for loading the new track.

Similarly, a function nextTrack() handles the loading of the next track and moving the index forward. The index is reset to the first track when the index reaches the last track. The loadTrack() method defined above is used for loading the new track.

Step 4: Configuring the sliders portion

We will be setting up two sliders that control the seek slider and the volume slider.

- **The seek slider**
  The seek slider shows the current playback position on a slider by updating it with the current time of the track. A new function is created seekUpdate() which handles the updating of the seek slider relative to the current time of the track. The seek slider position is calculated and set using the value property.

Now, this function has to be called every time the track progresses further. This can be done by scheduling it to be updated every second. This can be done using the setInterval() method with an interval of 1000 milliseconds. This timer is cleared every time a new track is loaded.

- This function also handles the changing of the time elapsed and the total duration of the track, which is updated every time this function fires. The minutes and the seconds are separately calculated and properly formatted to be displayed.

- **The volume slider**
  The volume slider is used to display an set the current volume of the track. A new function is created setVolume() which handles the setting of the volume slider whenever the user changes it.

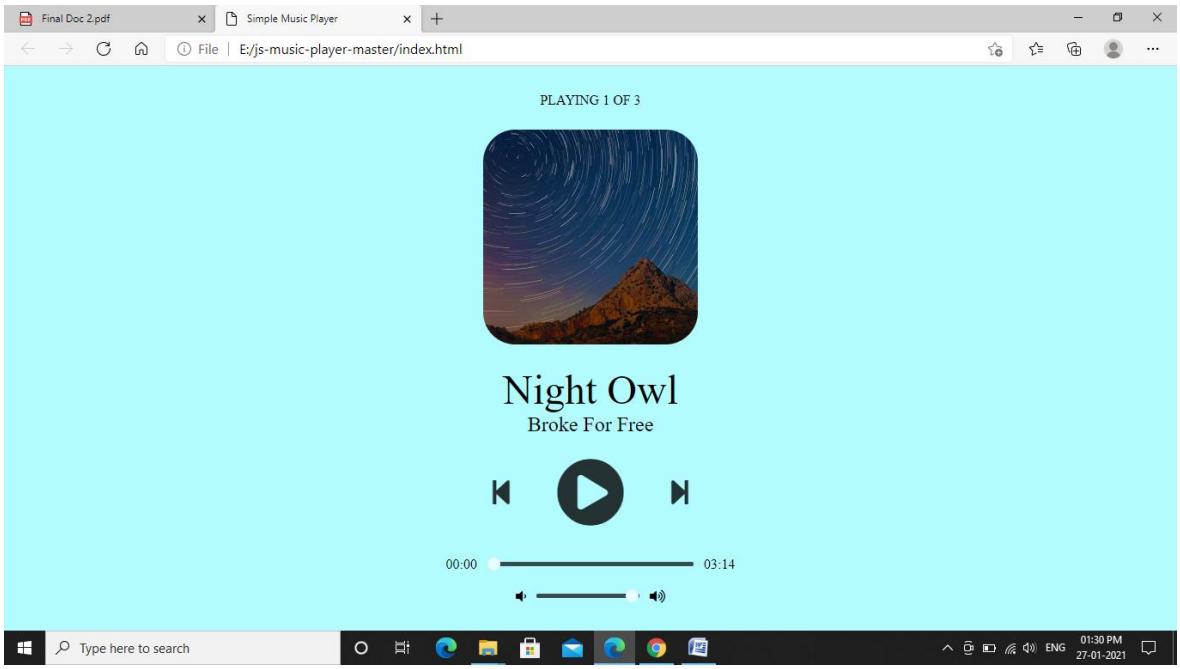
Step 5: Starting the player

- The first track is loaded by calling the loadTrack() function. This will load the first track from the tracklist and update all the details of the track. The user can then start playing the track using the play button. The previous and next track button would load the previous and next track respectively and start playing them.

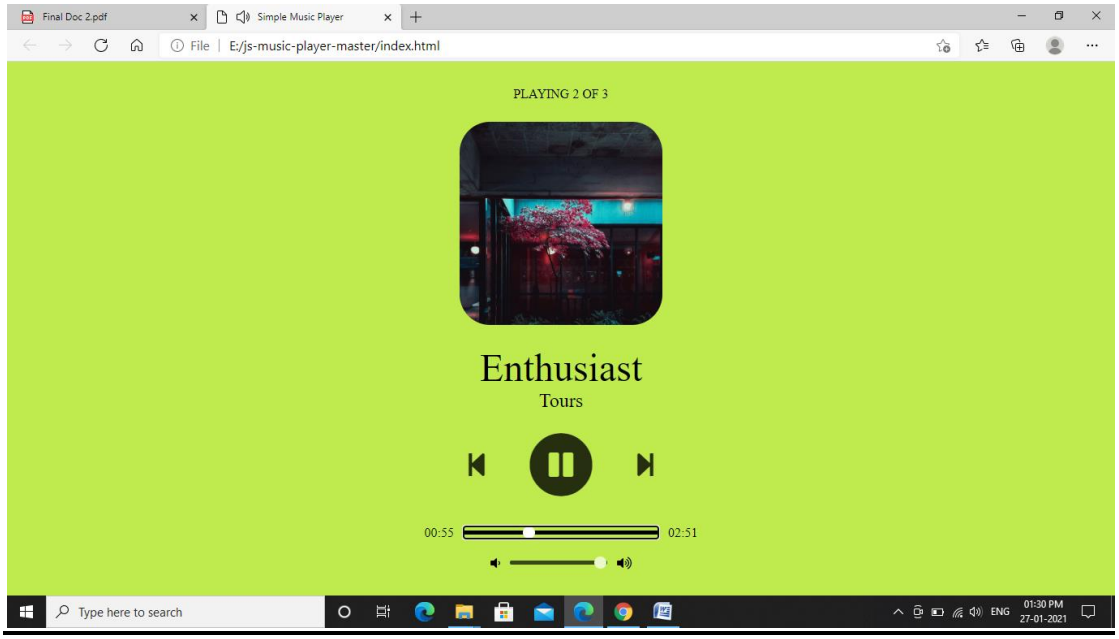
- The next track is automatically loaded when a track finishes playing. The user can seek to a position in the track using the seek slider. The volume can also be adjusted using the volume slider.

7.SCREENSHOTS

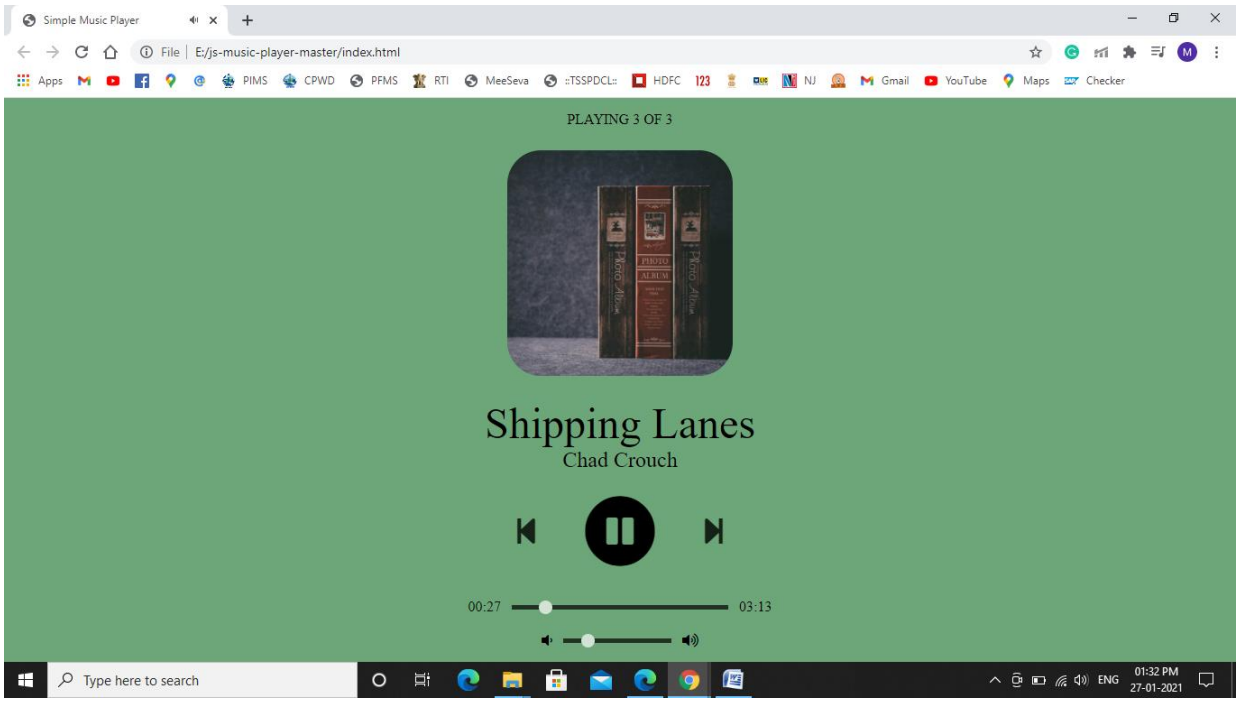
7.1MUSIC PLAYER



7.2 PLAYING SONG



7.3 PLAYING NEXT SONG



7.4 MOBILE SCREEN SHOTS

8 SYSTEM TESTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail. There are various types of tests. Each test type addresses a specific testing requirement.

8.1 UNIT TEST

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.2 INTEGRATION TEST

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event based and is more concerned with the basic outcome of console or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Procedures	: interfacing systems or procedures must be invoked.

Systematic coverage pertaining to identify Business process flows, data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.4 WHITEBOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.5 BLACKBOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

8.6 TESTING STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

Test Objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page

8.7 Test report

Tests have been generated and report is in the following Page.

TABLE 8.7.1 TEST REPORT

Test Case ID	Test Case	Expected Behavior	Exhibiting Behavior	Result
1	Play the song	To start playing the first song in tracklist	The first song in tracklist is played	Pass
2	Pause the song	To pause the song which is playing currently	Currently playing song paused	Pass
3	Play the next song	To play the next song in the tracklist	Playing the next song in the list	Pass
4	Play the previous song	To play the previous song in the tracklist	Playing the previous song in the list	Pass
5	Increase the volume	Volume should be increased according to the slider	Volume is increased	Pass
6	Decrease the volume	Volume should be decreased according to the slider	Volume is decreased	Pass
7	Alter the song	To play the song from point in slider	Played song from given point in slider	Pass

9.CONCLUSION

From the above discussion we conclude that the project Music Player System has successfully satisfied the requirements of the users. This project helps the user to play the music without requirement of any application or downloading the songs. This project can run over browser and user can listen to music anywhere and anytime.

10.BIBLIOGRAPHY

Used GOOGLE Search engine for all searches

- t,The Complete ReferenceHerbertSchild Java2,5thEdition, Herbert Schildt(2002), Tata McGrw-HillPublishing Company Ltd
- HTML5 W3C Recommendation 28 October 201422 (@ http://www.w3.org/TR/html5).
- CSS 2.1 Specification W3C Recommendation Revised 17 December 2014 (@ http://www.w3.org/TR/CSS21/).
- Matthew MacDonald, "Creating a Website - The Missing Manual", 3rd ed, 2011, O'Reilly.
  (A good introductory book on HTML/CSS. A new version is expected in July 2015.)
- Matthew MacDonald, "HTML 5 - The Missing Manual", 2nd ed, 2014, O'Reilly.
- David Sawyer McFarland, "CSS 3 - The Missing Manual", 3rd ed, 2013, O'Reilly.
- ECMAScript (JavaScript) Specification: "Standard ECMA-262 ECMAScript Language Specification 5.1", (same as "ISO/IEC 16262" 3rd eds).
- David Sawyer McFarland, "JavaScript and jQuery - The missing manual", 3rd ed, 2014, O'Reilly.
- Jonathan Chaffer and Karl Swedbery, "Learning jQuery", 4th ed, 2013, Packt Publishing.