**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

## BELAGAVI, KARNATAKA-590014



**Industry Internship Report**

on

**“MUSIC PLAYER”**

Submitted by

**Ms Minal Fathima. P**

**4DM21CS026**

**UNDER THE GUIDANCE OF**

**Mr Guruprasad G**

Asst Professor,

Department of Computer Science and Engineering

In partial fulfillment for the award of the degree of

# BACHELOR OF ENGINEERING

In

**COMPUTER SCIENCE AND ENGINEERING**



**YENEPOYA INSTITUTE OF TECHNOLOGY**

**N.H.13, THODAR, MOODBIDRI-574225, MANGALORE, D.K.**

**2023-24**

**YENEPOYA INSTITUTE OF TECHNOLOGY**

**THODAR, MIJAR POST, MOODBIDRI**-**574225**

**(Affiliated to Visvesvaraya Technological University, Belagavi)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



CERTIFICATE

This is to certify that the Industry Internship report entitled **“MUSIC PLAYER”** is an authentic record of the work carried out by **Ms MINAL FATHIMA P,** 4DM21CS026, student of 5th semester in partial fulfillmentof requirements for the award of Bachelor’s Degree in **Computer Science & Engineering** prescribed by Visvesvaraya Technological University during the year **2023- 24**

|  |  |  |
| --- | --- | --- |
| Signature of the Guide |  | Signature of the HOD |
| **(Mr Guruprasad G)** |  | **(Dr Manjunath Kamath K)** |
|  | External Viva |  |
| **Name of the examiner** |  | **Signature with Date** |
| 1**.** |  | 1. |
| 2. |  | 2. |

# ACKNOWLEDGEMENT

The successful completion of any work would be incomplete without a mention of the people who made it possible, whose constant guidance and encouragement served as a beacon light and crowned my efforts with success. I owe my gratitude to many people who helped and supported me during my Project **“Music Player ”**.

My deepest thanks to my guide **Mr Guruprasad G**, Asst. Professor, Dept. of CSE, Yenepoya Institute of Technology for his constant support , encouragement and providing me with the necessary advices and help. I am highly indebted to him for taking keen interest in my work, monitoring and providing guidance throughout the completion of my work.

I also thank **Dr Athokpam Bikramjit Singh,** Professor and Industry Internship Coordinator, Department of Computer Science & Engineering for his constant encouragement and support extended throughout.

I express my sincere gratitude to **Dr Manjunath Kamath K,** Professor & Head of the Department of Computer Science & Engineering for his invaluable support and guidance.

I sincerely thank **Dr R G. D’Souza**, Principal, Yenepoya Institute of Technology for his constant support and providing me with all the facilities that were required.

Finally, yet importantly, I express my heartfelt thanks to my family & friends for their wishes, encouragement and providing me moral strength for the successful seminar presentation.

**MINAL FATHIMA P**



# ABSTRACT

In the ever-evolving landscape of web development, the fusion of art and technology has paved the way for innovative digital platforms. This encapsulates the essence of a meticulously designed music player, embodying the principles and practices of Full Stack Web Development.

This music player aspires to redefine the digital music experience by seamlessly blending robust backend architecture with an intuitive and aesthetically pleasing front-end interface. The music player encompasses a comprehensive set of features, including user authentication, personalized playlists, dynamic music playback, and seamless social integration.

This Full Stack Web Development project unfolds as a journey through the intricacies of software development, from conceptualization and design to implementation and deployment. The report details the technologies utilized, challenges faced, and solutions devised to create a web application that not only meets technical standards but also immerses users in a world where the synergy of code and music harmoniously coexists.

# CONTENTS

1. **Introduction 1**
2. **Literature Survey 2**
3. **Problem Statement and Solution Strategy 3-4**
   1. Problem Statement 3
   2. Solution Strategy 3-4
4. **Proposed System 5**
5. **System Requirements Analysis and Specification 6-9**
   1. Hardware Requirements 6
   2. Software Requirements 7-9
6. **System Design 10-11**
7. **Implementation/Work Done During Internship 12-24**
   1. Weekly Report 12-16
      1. Week1 12-13
      2. Week2 14-15
      3. Week3 15-16
      4. Week4 16-21
   2. Project 17
      1. Overview 17
      2. Features 17
      3. Usage 17
      4. Pseudocode 18-20

Result 21

1. **Reflections/Learnings 22**
2. **Conclusion 23**

**Bibliography**

**References 24**

**LIST OF FIGURES**

* 1. Framework of General Music Player 10
  2. User Interface of Music Player 10
  3. Flow Chart of Music Player 11

# Chapter 1

# INTRODUCTION

Full-stack web development is the comprehensive approach of building web applications, encompassing both front-end and back-end development. On the front end, developers create the user interface and experience using languages like HTML, CSS, and JavaScript. This involves designing responsive and interactive elements that users interact with directly. On the back end, developers focus on server-side logic, databases, and server deployment. Common back-end technologies include Node.js, Python and Java.

In the dynamic realm of Full Stack Web Development, the convergence of creativity and technology has given rise to an array of innovative digital platforms. This report delves into the intricacies of the development process behind a comprehensive music player that seamlessly integrates the art of music with the science of web technologies.

In an era where music has become an integral part of our daily lives, the demand for sophisticated and user-friendly platforms to explore and enjoy music is ever-growing. This music player seeks to address this need by providing a feature-rich, immersive, and intuitive music player experience, encapsulated within a finely crafted web environment.

# Chapter 2

# LITERATURE SURVEY

Music players have undergone a remarkable transformation over the years, adapting to advancements in technology and user preferences. From the classic vinyl records to modern streaming services, each era has brought new possibilities and challenges.

Portability and Convenience: Early portable music players like the Walkman offered the freedom to take music on the go. Modern devices, such as smartphones and MP3 players, further enhanced portability, allowing users to carry extensive music libraries effortlessly.

Accessibility and Variety: Streaming services like Spotify and Apple Music provide vast libraries with millions of songs, offering users an unprecedented variety of music at their fingertips.

Customization and Personalization: Digital music players enable users to create personalized playlists, fostering a unique and tailored listening experience. Features like shuffle play and recommendations algorithms enhance user engagement by introducing them to new artists and genres based on their preferences.

Quality of Sound: Advancements in audio technology have led to high-quality sound reproduction, delivering a more immersive and enjoyable listening experience. Audiophiles appreciate the availability of lossless audio formats and high-fidelity streaming options for superior sound quality.

Impact on Artists and Revenue: The shift to digital music consumption has raised concerns about fair compensation for artists, as streaming platforms often pay minimal royalties.

Privacy and Data Concerns: Streaming services often collect user data to tailor recommendations, raising privacy concerns for some users. The storage and sharing of personal listening habits can lead to targeted advertising and potential breaches of privacy. The evolution of music players has significantly influenced how we experience and interact with music.

# Chapter 3

# PROBLEM STATEMENT AND SOLUTION STRATEGY

* 1. PROBLEM STATEMENT

The current music player faces challenges centered around connectivity, artist compensation, and user privacy. Users encounter disrusptions due to poor internet connectivity, hindering the promised seamless music experience. Simultaneously, artists grapple with inadequate compensation within the digital streaming paradigm, demanding a reevaluation of revenue-sharing models. Privacy concerns escalate as extensive user data collection for personalized recommendations raises ethical questions and compromises user trust. The need for a comprehensive solution is apparent, requiring innovations in offline functionality to ensure accessibility, a fair compensation model to support artists, and stringent privacy measures to safeguard user data. Addressing these challenges is essential for transforming the music player into a user-centric, equitable, and privacy-conscious platform.

* 1. SOLUTION STRATEGY

To address the complex challenges faced by contemporary music players, a multifaceted solution strategy is essential. First and foremost, enhancing offline functionality is crucial to mitigate the impact of unreliable internet connectivity. Implementing robust offline modes that allow users to seamlessly access their music libraries without dependence on an internet connection will ensure uninterrupted enjoyment of music in various settings.

Regarding artist compensation, a fair and transparent revenue-sharing model must be established. Collaboration between music player platforms and artists' unions can lead to the development of a compensation framework that ensures artists receive equitable payment for their work. Exploring innovative revenue streams, such as direct fan support or tiered subscription models that prioritize artist compensation, can contribute to a more sustainable and ethical ecosystem.

Privacy concerns can be addressed through a privacy-centric design approach. Implementing stringent data protection measures, anonymizing user data whenever possible, and providing clear, concise privacy policies will build trust among users. Offering opt-in features for personalized recommendations with granular control over data sharing will empower users to make informed choices about their privacy.

Additionally, fostering an open dialogue with users through feedback mechanisms can aid in continually refining the music player experience. Regular updates based on user input, incorporating new features, and staying ahead of technological advancements will ensure the platform remains relevant and competitive.

The solution strategy involves enhancing offline functionality, establishing a fair artist compensation model, prioritizing user privacy through robust data protection measures, and maintaining a dynamic, user-driven approach to platform development. This holistic strategy aims to transform the music player into a resilient, user-friendly, and socially responsible platform that meets the evolving needs of both users and music industry stakeholders.

# Chapter 4

# PROPOSED SYSTEM

As part of learning to code, I completed a project that involved creating a basic Music player using HTML, CSS, and JavaScript. This project was designed to showcase my skills in front-end web development. It is a responsive and user-friendly web application that allows users to play music. I used HTML and CSS to design and style the interface, and I implemented JavaScript to handle user input and interaction such as basic playback controls -play, pause, and next. I also gained experience working with version control tools such as Git and GitHub.

A music player aims to provide users with an immersive and user-friendly platform for listening to and discovering music. The website typically features a sleek and intuitive user interface that allows users to play a vast library of songs. Key features include user customization options. Responsive design ensures a seamless experience across various devices. Users can explore different genres, create personalized playlists, and enjoy a continuous playback experience. Additionally, integration with social media platforms may enhance the sharing and discovery of music. The music player combines functionality with an engaging design to create a compelling digital space for music enthusiasts to enjoy and explore diverse musical content.

# Chapter 5

# SYSTEM REQUIREMENTS ANALYSIS AND

# SPECIFICATION

## **HARDWARE REQUIREMENTS**

* **Intel core i5 and above:** Intel Core i5 and above are a series of processors developed by Intel, known for their performance and versatility. These processors are part of the Intel Core family and are commonly used in a variety of computing devices, from laptops to desktops. When choosing a specific Intel Core i5 or higher model, consider your computing needs, budget, and the requirements of the applications you intend to run. Always check for the latest generation and compare specifications to make an informed decision based on your preferences and requirements.
* **RAM:** A 4GB RAM size is considered minimal for modern computing needs. While it can still handle basic tasks like web browsing, document editing, and lightweight applications, it may struggle with more resource-intensive activities such as gaming, video editing, or running multiple demanding applications simultaneously. This allows for better multitasking and improved performance across a broader range of applications.
* **Hard disk:** A 250GB hard disk drive is relatively small by modern standards. While it can still be sufficient for basic computing needs, it may fill up quickly, especially as file sizes and the demand for storage space increase over time. 250GB is suitable for basic tasks like storing documents, photos, and a moderate number of applications. However, if you work with large media files, such as videos or high-resolution images, you may find the storage space limited. The space required for the operating system and installed applications should be taken into account. As software updates and new applications are introduced, they often require more storage.

## **SOFTWARE REQUIREMENTS**

* + - **VS Code:** Visual Studio Code (VS Code) can be a valuable tool for developing a music player website by providing a feature-rich and efficient development environment.

VS Code offers a powerful code editor with features like syntax highlighting, autocompletion, and code snippets. This aids developers in writing clean and error-free code for the frontend (HTML, CSS, JavaScript) and backend. It includes an integrated terminal, allowing developers to run commands directly within the editor, it seamlessly integrates with version control systems like Git, it supports a vast library of extensions, it comes with built-in debugging tools for various languages. Developers can set breakpoints, inspect variables, and debug JavaScript code, which is essential for troubleshooting and optimizing the functionality of a music player website. The Live Server extension in VS Code allows developers to launch a local development server with live reload capabilities.

* + - **HTML:** HTML (Hyper Text Markup Language) plays a fundamental role in building the structure and content of a music player website. HTML defines the overall structure of the web pages, including the layout, headers, footers, and main content area. It establishes the foundation for organizing various components of the music player website. Semantic HTML elements, such as <header>, <nav>, <main>, <footer>, and others, provide a meaningful structure to the content. HTML5 introduced the <audio> element, which is crucial for embedding audio content, such as music tracks, directly into web pages. It allows specifying the audio file source, controls, and other attributes for a seamless music playback experience. If the music player website includes a playlist feature, HTML is used to structure the list of tracks. This could involve using ordered or unordered lists (<ol>, <ul>) and list items (<li>) to present the playlist in an organized manner. In summary, HTML is the backbone of a music player website, defining the structure, content, and interactive elements that contribute to a seamless user experience.
    - **JavaScript:** JavaScript is crucial in a music player website for enhancing interactivity, managing dynamic content, and creating a smooth user experience. JS is used to control

the playback of audio elements. This involves functions to play, pause, stop, adjust volume, and skip tracks. JS can be utilized to dynamically load content, such as playlists or additional tracks, without requiring a full page reload. JS is involved in handling user authentication and authorization processes. Local Storage in JavaScript allows the website to store small amounts of data on the user's device. In summary, JavaScript enhances the functionality and user experience of a music player website by enabling dynamic content updates, managing audio playback, and facilitating seamless interactions between the user and the application.

* + - **CSS:** CSS (Cascading Style Sheets) is crucial in a music player website for styling and enhancing the visual presentation of the user interface.CSS is used to define the visual styling of various elements, such as buttons, navigation bars, backgrounds, and fonts. CSS is employed to create a responsive design that adapts to different screen sizes and devices. For the audio player interface, CSS is used to style and customize the appearance of playback controls, progress bars, volume sliders, and other elements to ensure a cohesive and user-friendly design.CSS is instrumental in defining color schemes and theming for the website. In summary, CSS in a music player website is essential for creating an attractive and user-friendly interface, ensuring responsive design, and customizing the appearance of various elements to align with the website's overall theme and branding.
    - **Node js:** Node.js is commonly used in the backend of a music player website to handle server-side logic, manage data, and communicate with databases or external APIs.Node.js allows developers to build a server that handles HTTP requests and responses. Node.js is well-suited for developing RESTful APIs that the frontend can interact with. Node.js can handle file uploads and downloads, which is important for managing music files.Node.js can handle user authentication and authorization processes. This includes verifying user credentials, managing user sessions, and ensuring that only authorized users can access certain functionalities.Node.js, with its non-blocking and event-driven architecture, is well-suited for building scalable and performant backend services for music player websites**.**
    - **Firebase :** Integrating Firebase into a music player website can enhance functionality, offering features like real-time data synchronization, user authentication, and cloud storage. Use Firebase Realtime Database can be used to store and synchronize music- related data in real time. This includes playlists, track information, and user-specific data. Firebase Authentication can be used to handle user registration and login. This provides secure access to features like personalized playlists and user-specific data. Firebase Cloud Storage is ideal for storing audio files associated with the music tracks. This allows users to upload, retrieve, and stream audio content from the cloud. Firebase Firestore can be used as an alternative to the Realtime Database for structured data storage. Firestore offers powerful querying capabilities and supports more complex data structures. Firebase Hosting can be used to deploy and host the music player website. It provides a secure and scalable hosting solution with features like automatic SSL, CDN, and easy deployment.By integrating Firebase into your music player website, you can leverage a serverless architecture with real-time capabilities, user authentication, and cloud storage, simplifying the development process and enhancing the user experience.

# Chapter 6

# SYSTEM DESIGN

FRONT-ENDBACK-END

PLAYER INTERFACE

(ACTIVITY)

PLAYBACK

(SERVICE)

MUSIC LIST

(ACTIVITY)

Fig 6.1 Framework of General Music Player

**TRENDING NOW!**

**SONG 1**

**SONG 2**

**SONG 3**

**SONG 4**

**SONG 5**

Progress Bar

Currently Control Options

Playing

Fig 6.2 User Interface of Music Player

RUN THE SOURCE CODE

PLAY NEXT SONG

BROWSE THROUGH THE SONGS

SELECT THE SONG OF CHOICE

PLAY PREVIOUS SONG

PLAY NEXT SONG

RESUME THE SONG

WHAT TO DO NEXT?

PAUSE THE SONG

PLAY THE SELECTED SONG

Fig 6.3 Flow chart of Music Player

# Chapter 7

# IMPLEMENTATION/WORK DONE DURING INTERNSHIP

The initial week of the Full stack Development internship commenced with a warm welcome and thorough orientation. The team provided an overview of the project's objectives, fostering a sense of inclusion and purpose. Participants engaged in sessions exploring the diverse career opportunities within Full stack Development. The discussions shed light on the significance of being well-versed in both frontend and backend technologies, emphasizing the role of Full stack Developers in creating cohesive and efficient software solutions. With a foundational understanding of the internship context and career prospects, hands-on coding sessions commenced. Participants delved into the basics of web development, starting with HTML, CSS, and JavaScript.

## WEEKLY REPORT

* + 1. WEEK 1

Introduction, Career Opportunities, Basic Web Development, and Environment Setup

Introduction to the Internship:

The initial week of the Full stack Development internship commenced with a warm welcome and thorough orientation. The team provided an overview of the project's objectives, fostering a sense of inclusion and purpose.

Career Opportunities in Full stack Development:

Participants engaged in sessions exploring the diverse career opportunities within Full stack Development. The discussions shed light on the significance of being well-versed in both frontend and backend technologies, emphasizing the role of Full stack Developers in creating cohesive and efficient software solutions.

Basic HTML,CSS and JavaScript:

With a foundational understanding of the internship context and career prospects, hands-on coding sessions commenced. Participants delved into the basics of web development, starting with HTML, CSS, and JavaScript.

Setting up the Development Environment:

An essential aspect of Week 1 was setting up the development environment. Visual Studio Code (VSCode) was chosen as the integrated development environment (IDE) for its versatility and extensive plugin support.

Downloading Visual Studio Code:

Participants were guided through the process of downloading and installing Visual Studio Code from the official website. The step-by-step instructions ensured a smooth setup for both Windows and macOS users.

Visit the [Visual Studio Code website] (https://code.visualstudio.com/). Click on the "Download" button.

Follow the on-screen instructions for your operating system. Once the installation is complete, launch Visual Studio Code.

Configuring Visual Studio Code:

To enhance the development experience, participants were introduced to essential VSCode extensions. The "Live Server" extension, for instance, was recommended for quickly launching and previewing HTML files.

* Open Visual Studio Code.
* Navigate to the Extensions view by clicking on the Extensions icon in the Activity Bar on the side of the window or using the keyboard shortcut `Ctrl+Shift+X`.
* Search for "Live Server" and click on the "Install" button.
* Once installed, click on the "Reload" button to activate the extension.
* Open an HTML file, right-click, and select "Open with Live Server" to launch a live preview of the webpage.
  + 1. WEEK 2

React Fundamentals and Node.js Installation

Frontend Exploration with React:

The second week of the Fullstack Development internship centered around a comprehensive exploration of React, a JavaScript library for building user interfaces.

Understanding React Components:

Participants delved into the core concept of React components. They grasped the difference between functional and class components and gained a deep understanding of JSX, the syntax extension for JavaScript recommended by React.

State Management in React:

The week extended to the essential topic of state management in React. Participants learned the significance of state in creating dynamic and interactive components and explored the usage of the useState hook.

Hands-On Coding Tasks:

Practical coding tasks were integral to solidifying React concepts. Participants engaged in creating interactive components, handling user input, and managing component state, fostering a hands-on approach to learning.

Introduction to Node.js:

Complementing the frontend exploration, the week introduced participants to the backend aspect of development using Node.js. The event-driven, non-blocking I/O model of Node.js was emphasized, highlighting its suitability for scalable applications.

Node.js Installation:

To enable participants to seamlessly navigate both frontend and backend development, the installation of Node.js was a key focus. Clear instructions were provided for installing Node.js and npm (Node Package Manager) on various operating systems.

* + - 1. Visit the [Node.js website](https://nodejs.org/).
      2. Download the recommended LTS version for your operating system.
      3. Follow the installation instructions provided on the website.
      4. Once installed, open a terminal or command prompt and verify the installation using the following commands:
         1. `node -v` (to check Node.js version)
         2. `npm -v` (to check npm version)
    1. WEEK 3

Login/Signup Page Development and Firebase Integration

Login/Signup Page Development:

Building upon the foundation laid in previous weeks, participants immersed themselves in the development of a login/signup page. The focus was on creating an interactive and user-friendly authentication experience.

Enhancing the Login Page:

The login page structure, introduced in Week 2, was refined and extended to accommodate both login and signup functionalities. Participants implemented input validation and designed a seamless user interface for a streamlined authentication process.

CSS Styling for Authentication Page:

CSS styling was applied to enhance the visual appeal and responsiveness of the login/signup page. Participants fine-tuned the styles to create an engaging user interface.

JavaScript for Authentication Logic:

JavaScript played a pivotal role in implementing the authentication logic. Participants created an auth.js script that handled form submissions, validated user input, and interacted with Firebase for authentication.

Firebase Integration:

The second major aspect of Week 3 was the integration of Firebase for authentication services. Firebase provided a seamless and scalable solution for user authentication.

Setting up Firebase:

Participants were guided through the process of setting up a Firebase project, obtaining API keys, and configuring Firebase Authentication.

Connecting Firebase to the Project:

The Firebase JavaScript SDK was added to the project, enabling seamless integration with Firebase services. Participants learned how to initialize Firebase and access authentication functionalities.

Firebase Configuration:

The configuration details, including the Firebase project's API key, were added to the JavaScript file to establish a connection with Firebase.

* + 1. WEEK 4

Working on project

In Week 4 of the Full stack Development internship, participants embarked on an exciting projects to develop different real-time websites.

Project Objectives:

React Components:

Participants created reusable React components to efficiently manage the UI, ensuring a dynamic and responsive user experience.

API Integration:

The project involved integrating a Movie Database API to fetch and display relevant movie information. Axios, a promise-based HTTP client, facilitated seamless communication with the API.

## PROJECT

* + 1. OVERVIEW

I completed a project that involved creating a basic music player using HTML, CSS, and JavaScript. This project was designed to showcase my skills in front-end web development. It is a responsive and user-friendly web application that allows users to play music. I used HTML and CSS to design and style the interface, and I implemented JavaScript to handle user input and interaction such as basic playback controls -play, pause, and next. I also gained experience working with version control tools such as Git and GitHub.

* + 1. FEATURES
* Play/Pause: Allows the user to start or pause the playback of the currently selected track.
* Next/Previous: Enables the user to skip to the next or previous track in the playlist or album.
* Seek: Allows the user to manually move forward or backward within a track by dragging a slider or clicking on a progress bar.
* Display the currently playing song and album artwork.
  + 1. USAGE

Once the Music player is open in your browser, You can then click the play button next to the song to start playing it, and use the playback controls at the bottom of the page to control the song playback. The currently playing song and album artwork are displayed in the "Now Playing" section at the bottom of the page.

## PSEUDO CODE

// Declare variables

let songIndex = 0;

let audioElement = new Audio('songs/1.mp3');

let masterPlay = document.getElementById('masterPlay');

let myProgressBar = document.getElementById('myProgressBar');

let gif = document.getElementById('gif');

let masterSongName = document.getElementById('masterSongName');

let songItems = Array.from(document.getElementsByClassName('songItem'));

// Define the list of songs

let songs = [

{ songName: "Ayyo Vayye", filePath: "songs/1.mp3", coverPath: "covers/1.jpeg" },

// ... (similar entries for other songs)

];

// Initialize song items with cover images and names

songItems.forEach((element, i) => {

element.getElementsByTagName("img")[0].src = songs[i].coverPath;

element.getElementsByClassName("songName")[0].innerText = songs[i].songName;

});

// Event listener for master play button

masterPlay.addEventListener('click', () => {

if (audioElement.paused || audioElement.currentTime <= 0) {

// If paused or at the beginning, play the audio

audioElement.play();

masterPlay.classList.remove('fa-play-circle');

masterPlay.classList.add('fa-pause-circle');

gif.style.opacity = 1;

} else {

// If playing, pause the audio

audioElement.pause();

masterPlay.classList.remove('fa-pause-circle');

masterPlay.classList.add('fa-play-circle');

gif.style.opacity = 0;

}

});

// Event listener for updating progress bar during audio playback

audioElement.addEventListener('timeupdate', () => {

progress = parseInt((audioElement.currentTime / audioElement.duration) \* 100);

myProgressBar.value = progress;

});

// Event listener for manually changing the progress bar

myProgressBar.addEventListener('change', () => {

audioElement.currentTime = myProgressBar.value \* audioElement.duration / 100;

});

// Function to reset play icon for all song items

const makeAllPlays = () => {

Array.from(document.getElementsByClassName('songItemPlay')).forEach((element) => {

element.classList.remove('fa-pause-circle');

element.classList.add('fa-play-circle');

});

};

// Event listeners for individual song item play buttons

Array.from(document.getElementsByClassName('songItemPlay')).forEach((element) => {

element.addEventListener('click', (e) => {

makeAllPlays();

songIndex = parseInt(e.target.id);

e.target.classList.remove('fa-play-circle');

e.target.classList.add('fa-pause-circle');

audioElement.src = `songs/${songIndex + 1}.mp3`;

masterSongName.innerText = songs[songIndex].songName;

audioElement.currentTime = 0;

audioElement.play();

gif.style.opacity = 1;

masterPlay.classList.remove('fa-play-circle');

masterPlay.classList.add('fa-pause-circle');

});

});

// Event listener for the "Next" button

document.getElementById('next').addEventListener('click', () => {

if (songIndex >= 9) {

songIndex = 0;

} else {

songIndex += 1;

}

audioElement.src = `songs/${songIndex + 1}.mp3`;

masterSongName.innerText = songs[songIndex].songName;

audioElement.currentTime = 0;

audioElement.play();

masterPlay.classList.remove('fa-play-circle');

masterPlay.classList.add('fa-pause-circle');

});

// Event listener for the "Previous" button

document.getElementById('previous').addEventListener('click', () => {

if (songIndex <= 0) {

songIndex = 0;

} else {

songIndex -= 1;

}

audioElement.src = `songs/${songIndex + 1}.mp3`;

masterSongName.innerText = songs[songIndex].songName;

audioElement.currentTime = 0;

audioElement.play();

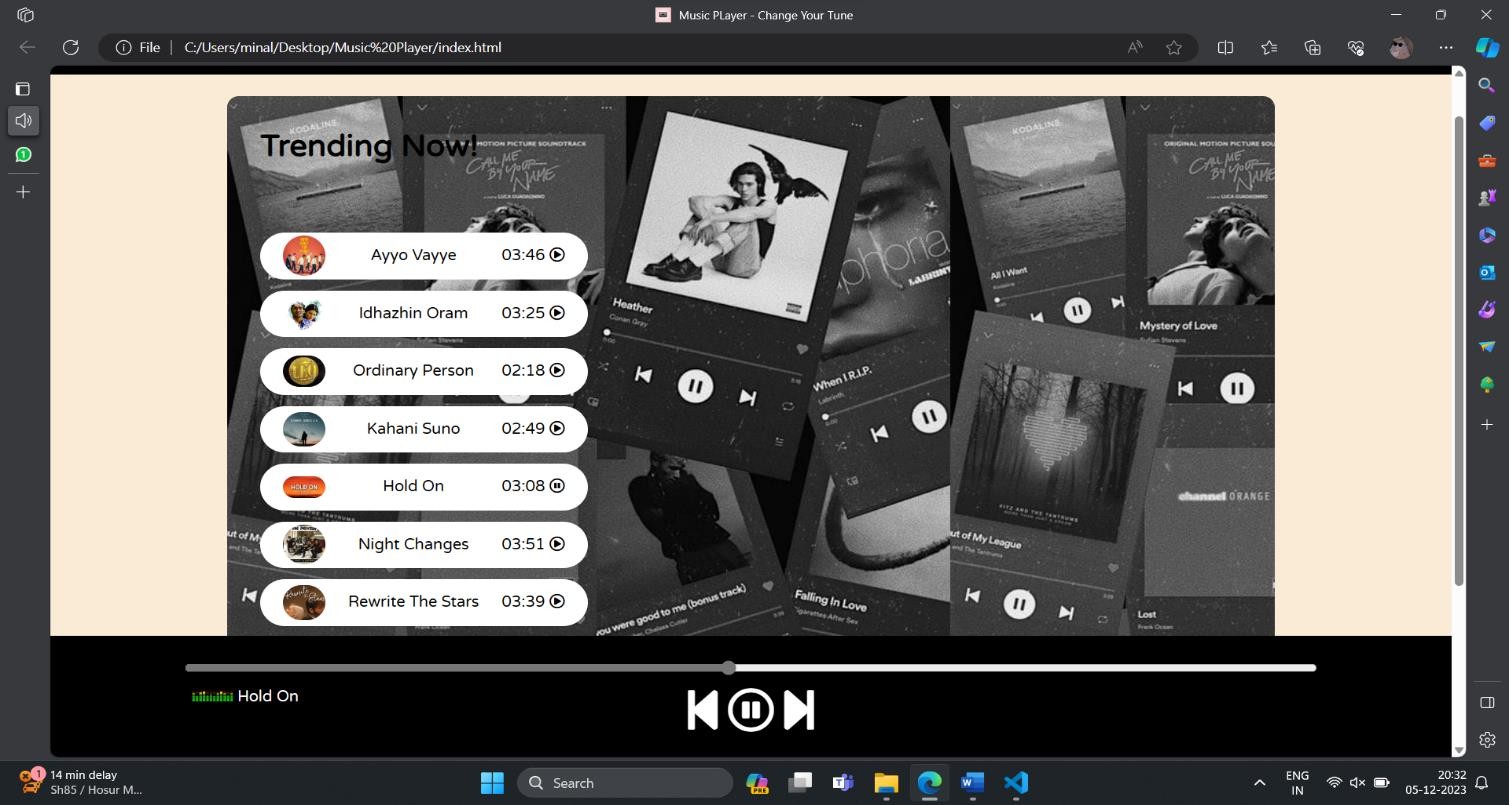
masterPlay.classList.remove('fa-play-circle');

masterPlay.classList.add('fa-pause-circle');

});

**RESULT**

Snapshot 1



# Chapter 8

# REFLECTION/LEARNINGS

The culmination of the Full stack Development internship marks a journey filled with growth, challenges, and profound learning experiences. Throughout the internship, participants engaged in a progressive exploration of frontend and backend technologies, honing their skills to become adept developers.

Key Learnings:

Frontend Mastery:

Proficiency in creating dynamic and visually appealing user interfaces using HTML, CSS, and JavaScript, In-depth understanding and application of React for building interactive components.

Backend Proficiency:

Introduction to backend development with Node.js, providing a comprehensive Full stack skill set, Integration of external services, such as Firebase, for robust authentication.

API Integration and Project Development:

Practical experience in working with external APIs, demonstrated in the creation of an image searching website. Development of real-world projects, translating theoretical knowledge into tangible applications.

# Chapter 9

# CONCLUSION

The music player aims to provide a seamless and enjoyable music listening experience. With its user-friendly interface, diverse and customizable playlist, it caters to a wide audience. Additionally, features like personalized recommendations and offline listening enhance user satisfaction, making it a comprehensive platform for music enthusiasts. Constant updates and user feedback integration ensure the website stays dynamic and aligned with evolving preferences in the ever-changing world of music.

The significance of full-stack expertise lies in its versatility, allowing developers to adapt to evolving industry trends and demands. By mastering both client and server-side aspects, developers can contribute holistically to project lifecycles, fostering efficient collaboration and faster development cycles. This integrated skill set is particularly valuable in startups and smaller teams where individuals wear multiple hats.

Moreover, the continuous evolution of full-stack technologies underscores its enduring relevance. The demand for full-stack developers remains high as businesses seek agile and comprehensive solutions. In conclusion, embracing full-stack web development empowers professionals to build end-to-end solutions, stay adaptable in a dynamic tech landscape, and play pivotal roles in shaping the digital experiences of tomorrow.

# BIBLIOGRAPHY

**REFERENCES:**

1. Wikipedia
2. Geeksforgeeks
3. Youtube