A

REAL TIME RESEARCH PROJECT REPORT ON

RHYTHORA : MUSIC PLAYER SYSTEMS USING WEB DEVELOPMENT

Submitted in partial fulfilment of the requirements for award of the degree of

BACHELOR OF TECHNOLOGY

In

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BY

D.MEGHANA	**23BH1A7225**
T.VISHAL	**23BH1A7292**
CH.SHRISHA	**23BH1A7221**
P.PRUDHVIRAJ	**23BH1A7276**
CH.ARUN	**23BH1A7220**

Under the guidance of

Mr. ABHAY KUMAR M.Tech

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

St. Mary's Engineering College

(Approved by AICTE, NEW DELHI. & Affiliated to JNTU-HYDERABAD,Accredited by NAAC)

Deshmukhi (V), Pochampally (M), Yadadri Bhuvanagiri (D), Telangana-50824

[2024-25]

CERTIFICATE

This is to certify that project music player system **"RHYTHORA MUSIC PLAYER SYSTEMS USING WEB DEVELOPMENT"** is bonafide work carried out in the II/II semester by **D.MEGHANA (23BH1A7225), T.VISHAL (23BH1A7292), CH.SHIRISHA (23BH1A7221), P.PRUDHVIRAJ (23BH1A7276), CH.ARUN (23BH1A7220),** in partial fulfilment award of Bachelor of Technology in **(Artificial Intelligence & Data Science)** from **St. Mary's Engineering college** during the academic year 2024-25.

INTERNAL GUIDE
Mr. ABHAY KUMAR, M.Tech
Dept. of AI&DS

HEAD OF THE DEPARTMENT
Mr.B.SRISHAILAM M.Tech(ph.d)
Dept. of AI&DS

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible. We consider it as a privilege to express our gratitude and respect to all those who guided us in the completion of the project.

We are thankful to our internal guide **Mr. ABHAY KUMAR, M.Tech** in **Department of Artificial Intelligence & Data Science, St. Mary's Engineering College** for having been of a source encouragement and for insisting vigour to do this project work.

We are obliged to Mr. B. SRISHAILAM, Head of the Department of Artificial Intelligence & Data Science, St. Mary's Engineering College for his guidance and suggestion throughout project work.

We take this opportunity to express a deep sense of gratitude to **Dr. T.N. Srinivas Rao, Principal of St. Mary's Engineering College** for allowing us to do this project and for this affectionate encouragement in presenting this project work.

We convey our sincere thanks to **Sri Rev. K.V.K RAO, Chairman** of **St. Mary's Engineering College** for giving us learning environment to grow out self personally as well as professionally.

We would like to express our thanks to all staff members who have helped us directly and indirectly in accomplishing this project work. We also extended our sincere thanks to our parents and friends for their moral support throughout the project work. Above all we thank god almighty for his manifold mercies in carrying out this project work successfully.

D.MEGHANA	**-23BH1A7225**
T.VISHAL	**-23BH1A7292**
CH.SHIRISHA	**-23BH1A7221**
P.PRUDHVIRAJ	**-23BH1A7276**
CH.ARUN	**-23BH1A7220**

DECLARATION

This is to certify that the work report in thesis titled**,"RHYTHORA MUSIC PLAYER SYSTEMS USING WEB DEVELOPMENT**", submitted to **the Department of Artificial Intelligence & Data Science, St. Mary's Engineering College** in fulfilment of degree for the award of Bachelor of Technology, is a bonafide work done by us. No part of the thesis is copied from books, journals or internet and wherever the portion is taken, the same has been duly referred in the text. The reported results are based on the project work entirely done by us and not copied from any other sources. Also we declare that the matter embedded in this thesis has not been submitted by us in full or partially there for the award of any degree of any other institution or university previously.

D.MEGHANA -23BH1A7225

T.VISHAL -23BH1A7292

CH.SHIRISHA -23BH1A7221

P.PRUDHVIRAJ -23BH1A7276

CH.ARUN -23BH1A7220

CONTENTS

LIST OF FIGURES

LIST OF PLATES

ABSTRACT

The Music Player System is an application that allows users to play and organize their music files. It supports different audio formats and provides features like creating playlists, searching for songs, adjusting volume, and customizing sound with an equalizer. The user interface is simple and easy to use, available on both desktop and mobile devices. The system also suggests songs based on what users have listened to before. Overall, the Music Player System aims to offer a smooth and enjoyable music experience with easy access and personalized feature. The Music Player System is a digital application designed to manage and play audio files efficiently. The system allows users to browse, select, and play songs from their local storage while offering key features such as play, pause, stop, next, previous, volume control, and playlist creation. It supports multiple audio formats and displays metadata like song title, artist, and duration. Developed using a suitable programming language and framework (e.g., Java, Python, or Android), the project emphasizes a user-friendly interface, smooth navigation, and responsive design. The aim of this project is to provide an enjoyable music listening experience by integrating essential media player functionalities with a simple and attractive user interface.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Music Player System is a multimedia application that allows users browse, manage audio files with features such as playlist creation, shuffle and repeat modes, control, real-time progress The Music Player System is software application designed to enable users to play, organize ,and manage their digital music files efficiently. With the rapid growth of digitalmedia, user need intuitive tools access their music libraries Easily. This user-friendly interface that allows users to browse songs by title , artist or genre , create and manage playlists , and play music with standard controls such as play, pause, stop, skip and volume adjustment.

1.2 FEATURES

The Music Player System is a comprehensive desktop or mobile application designed to provide users with seamless audio playback functionality. It supports a wide range of audio formats (such as MP3, WAV, AAC), allowing users to browse their local music library or stream content. Key features include play/pause, forward/rewind, shuffle, and repeat modes; playlist management with the ability to create, edit, and delete custom playlists; real-time track progress bar; volume control and mute; and audio visualization. Additional enhancements include support for album art display, metadata editing (e.g., title, artist, genre), theme customization (light/dark modes),keyboard short cuts, and background playback.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The Web Music Player System is an interactive and dynamic online music player platform, designed to allow users to play, manage,and discover music directly from their web browsers . With a modern , user - friendly interface , the system will support features like song playback , playlists , favorites , search functionality , and customizable settings. Users will be able to upload their own music or stream tracks from integrated music APIs, creating a seamless and personalized music experience.

- User Accounts and Authentication : Secure user registration and login system , enabling users to create personalized playlists, save favorites, and manage their music preferences.
- Search and Discovery: Powerful search engine to browse through genres , albums , and individual tracks. Integration with music APIs for up-to-date music content.
- Playlists and Library Management: Users can create , edit , and share custom playlists. Ability to save and manage a personal music library.
- Responsive Design Fully responsive interface that work across devices(mobile tablet, desktop) for easy access to music at any time.
- Real-time Music Synchronization : Playlists and currently playing music wil Sync across devices for a consistent experience.

2.2 PROPOSED SYSTEM

The proposed system is a web-based music player application designed to allow users to manage and enjoy their personal music libraries online. This system will provide a streamlined, user-centric platform that supports music file uploads, playback, playlist creation, and personalized management of audio tracks. The goal is to combine efficient music playback with an intuitive user interface that works across devices. This web application will offer essential media player functions—play, pause, skip, shuffle, and repeat, along with advanced features such as playlist management, audio metadata organization (song title, artist, album), and a user authentication system to ensure data privacy and personalized access.

The system architecture follows a client-server model, separating the frontend user interface from backend services and data storage. Users can interact with the system through a modern, responsive web interface developed using React.js or Vue.js, while the backend, built using Node.js and Express, handles user requests, file uploads, and data management. Audio files and playlists will be securely stored in a database and/or cloud-based storage like AWS S3, allowing scalable access and storage. In addition to playback and management features, the system may incorporate search functionality, enabling users to quickly find songs by title, artist, or genre. Additional enhancements could include real-time lyrics integration, playlist sharing, or a recommendation engine powered by basic maching learning or external music API'S.

CHAPTER 3

SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, organization of a computer system. It outlines how system components such as hardware, software, databases, and networks interact with each other to meet the requirements. A good architecture ensures scalability, security, performance and maintainability of the system.

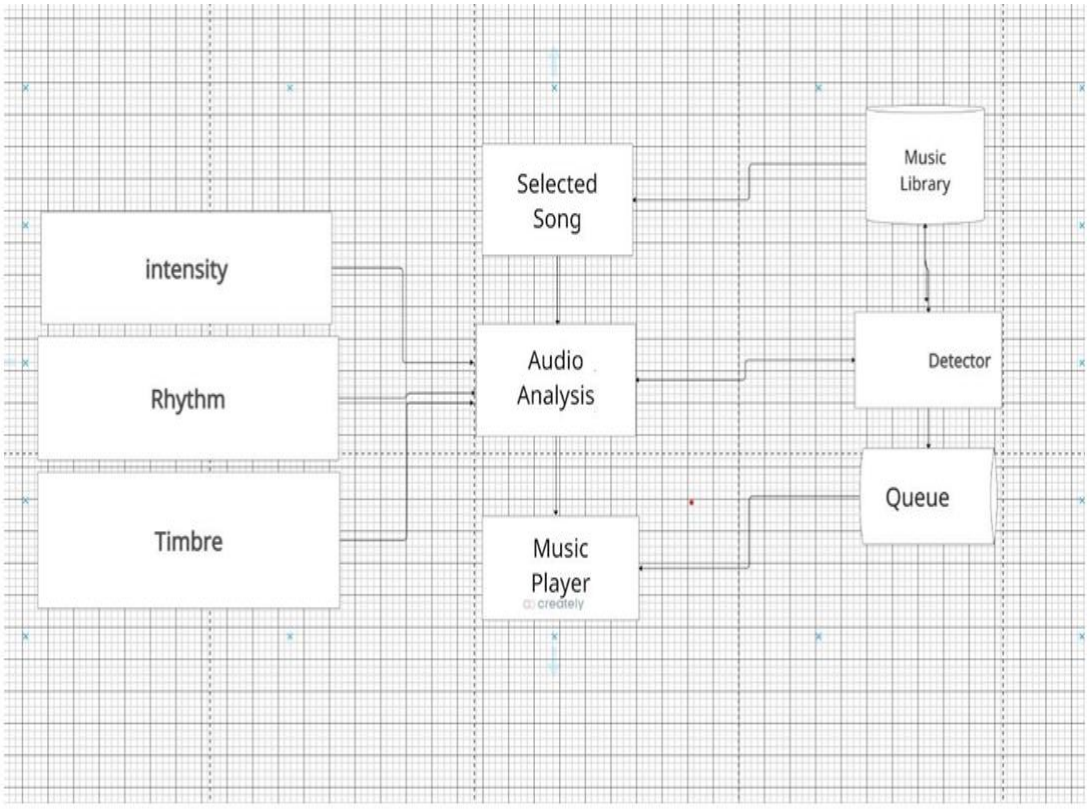

Figure no.3.1*: *system architecture

CHAPTER 4

SYSTEM STUDY

4.1 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must have a high demand for the available technical resources. This will lead to high demand for the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.2 ECOANOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to usethe system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make someconstructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS :

Processor	:	quad-core or higher
RAM	:	4 GB or above
Storage	:	32 GB or above
Input Devices	:	Keyboard Mouse

5.2 SOFTWARE REQUIREMENTS:

Operating System	:	Android , linux or windows
Frontend	:	HTML, CSS, Javascript
Tools	:	Android studio, Xcode, Visual studio

CHAPTER 6

SOFTWARE ENVIRONMENT

6.1 ANDROID

Android is an open-source mobile operating system developed by Google. It supports both native and hybrid application development. While Java and Kotlin are primarily used for native apps, Android also allows developers to embed web content using WebView, which supports HTML, CSS, and JavaScript. Developers can use these web technologies to create responsive UI, dynamic content, and interactive features that run inside an Android app. Frameworks like Apache Cordova, PhoneGap, and Ionic enable hybrid app development using standard web technologies.

Use of HTML, CSS, JavaScript in Android:

HTML structures the app content.

CSS styles the content for better visual presentation.

JavaScript adds interactivity and logic.

6.2 DOWNLOAD THE ANDROID SDK

To begin Android development, including web-based applications within Android, the Android SDK (Software Development Kit) is essential. It provides tools like:Android Debug Bridge (ADB)

Emulator

Build tools

Platform APIs

Developers must download the SDK using Android Studio, the official IDE for Android development. Alternatively, SDK tools can be manually downloaded for integration with other IDEs like Eclipse.For web-based development:

HTML, CSS, and JS files can be loaded into a WebView.

Testing tools in the SDK can simulate different screen sizes and resolutions.

CHAPTER 7

SYSTEM DESIGN

7.1 UML DIAGRAMS

7.1.1 INTRODUCTION

UML (Unified Modeling Language) is a powerful modeling language used to specify, visualize, construct, and document the components of a software system. In the development of a music player system, UML diagrams help teams understand the functional requirements, system structure, workflow, and interactions between various components and users.

Music player systems typically involve user interaction, media handling, streaming services, authentication, and data management — making them ideal candidates for comprehensive UML modeling.

7.1.1 USE CASE DIAGRAM

A **Use Case Diagram** for a Music Player System Project is a Unified Modeling Language (UML) diagram that visually represents the functional requirements of the system by showing the interactions between users (actors) and the system's use cases (functions).It is used during the requirements analysis phase of software development to define what the system should do from the user's perspective, not how it is done. Play Music User selects a song to play. Pause Music Temporarily stops the music without resetting its position. Stop Music Stops the music and resets the position to the beginning. Next Track Skips to the next song in the list or playlist. Previous Track Returns to the previous song. Search Music User searches for songs using keywords like title or artist. Create Playlist Allows user to make custom playlists. Add/Remove Songs from Playlist Modify songs in a playlist.

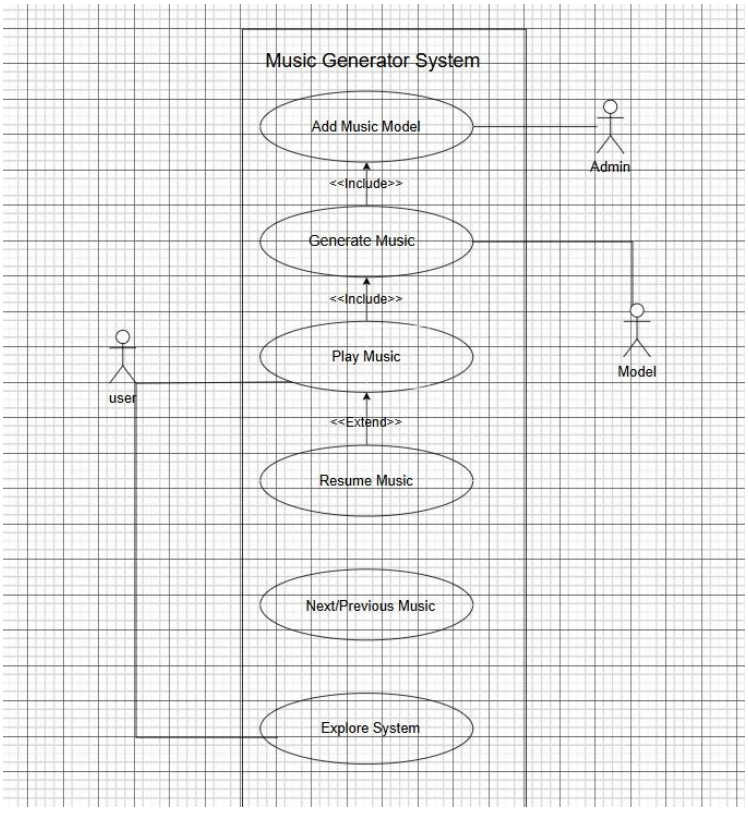

Figure7.1.1: *represents Usecase Diagram*

7.1.2 CLASS DIAGRAM

A class diagram is a type of static structure diagram in UML (Unified Modeling Language) that shows the structure of a system by illustrating its classes, attributes, methods, and the relationships among objects. It helps visualize the blueprint of a software system, including inheritance, associations, and dependencies between classes. Class diagrams are commonly used in object-oriented programming to model and design software before implement.

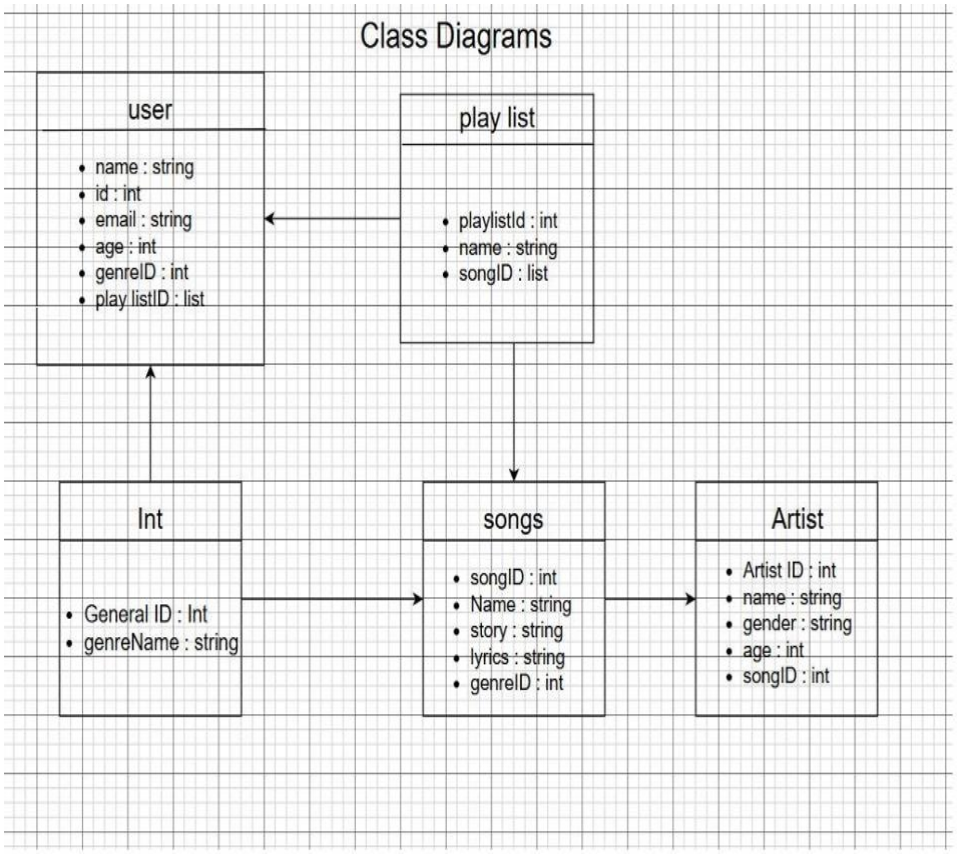

Figure 7.1.2: *represents Class Diagram*

7.1.3 SEQUENCE DIAGRAM

A sequence diagram is a type of UML (Unified Modeling Language) diagram that shows how objects interact in a particular sequence over time. It illustrates the flow of messages betwee n objects to carry out a specific function or process in a system. Sequence diagrams are useful for understanding and designing system behavior, especially in scenarios like user actions or system events. A sequence diagram for a music player system shows how objects in the system interact over time to perform specific functions such as playing, pausing, or stopping music.

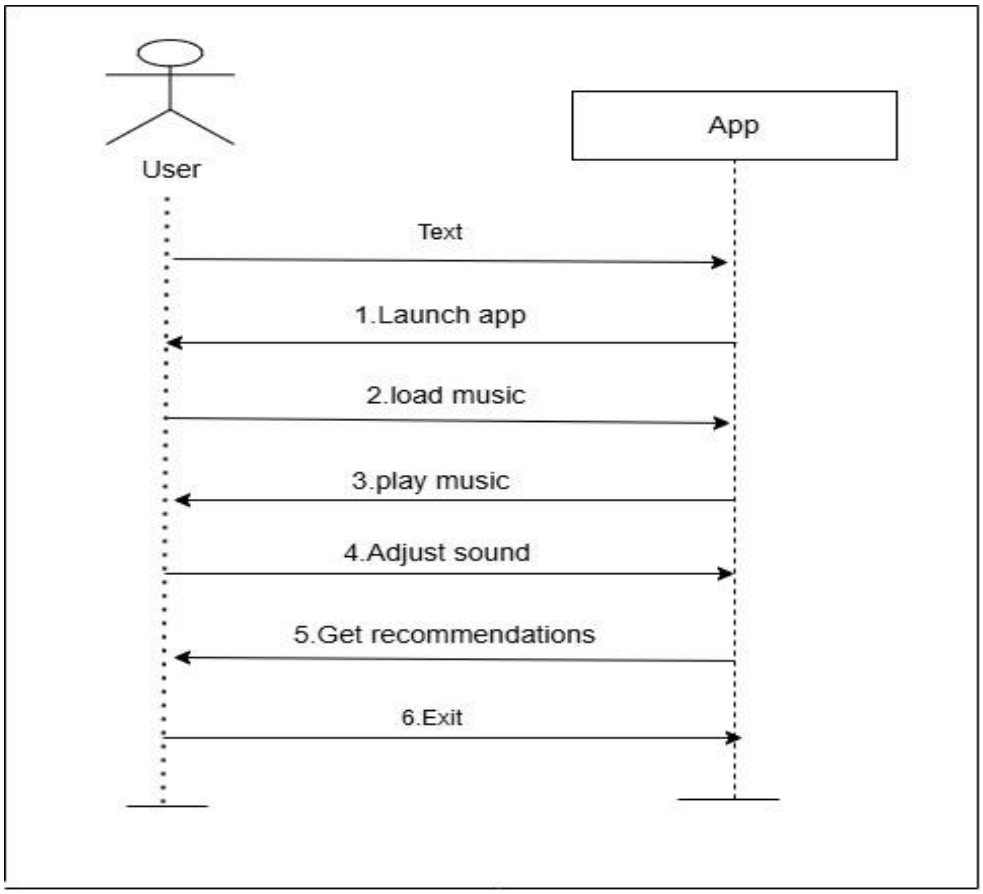

Figure 7.1.3*: Sequence Diagram*

7.1.4 ACTIVITY DIAGRAM

An activity diagram for a music player system represents the flow of activities involved in using the player, such as selecting a song, playing it, pausing, or stopping. It visualizes the sequence of actions, decisions, and parallel processes in the system. For example, it may start with the user opening the app, browsing the library, selecting a track, and then choosing actions like play, pause, next or stop. The activity diagram provides a visual representation of the system's functionality and user interactions, helping to identify requirements and design the system'

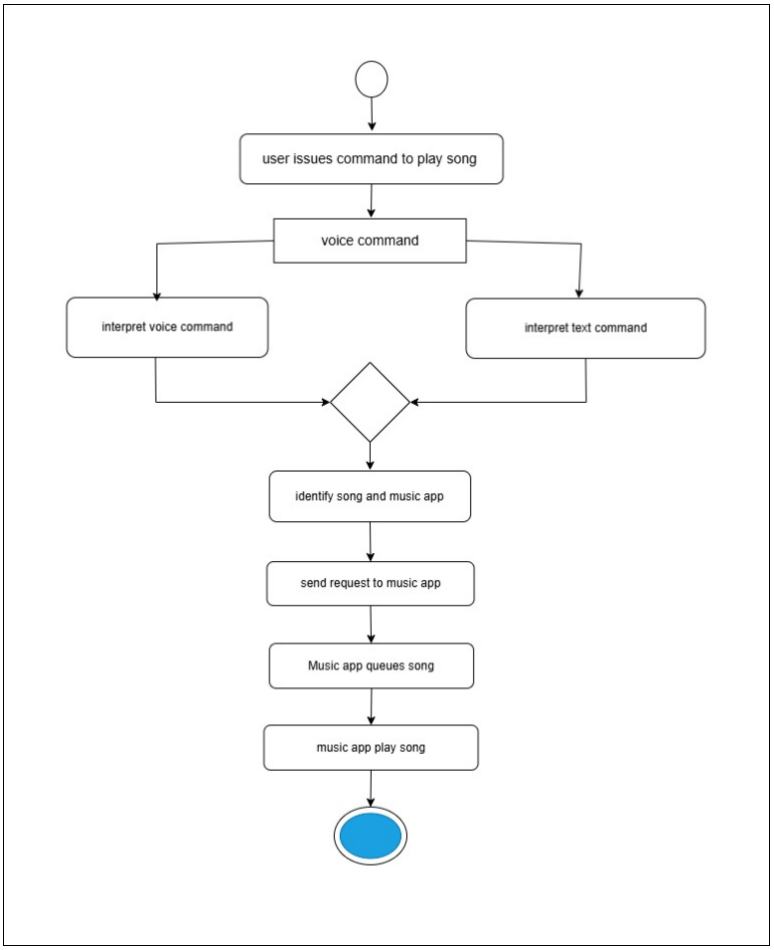

Figure7.1.4: *represents Activity Diagram*

CHAPTER 8

SOURCE CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Music Player</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }
        body {
            background-color: #121212;
            color: #fff;
            display: flex;
            height: 100vh;
        }
        .sidebar {
            width: 230px;
            background-color: #000;
            padding: 20px;
            display: flex;
            flex-direction: column;
        }
        .logo {
           color: #1DB954;
            font-size: 24px;
```

```
    font-weight: bold;
    margin-bottom: 30px;
}
.nav-links {
    margin-bottom: 30px;
}
.nav-link {
    display: flex;
    align-items: center;
    color: #b3b3b3;
    padding: 10px 0;
    cursor: pointer;
    transition: color 0.2s;
    font-weight: 600;
}
.nav-link:hover, .nav-link.active {
    color: #fff;
}
.nav-link i {
    margin-right: 15px;
    font-size: 20px;
}
.library {
    flex-grow: 1;
}
.library-header {
    display: flex;
 justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}
.library-title {
```

```
    color: #b3b3b3;
    font-size: 14px;
    font-weight: bold;
}
.playlist {
    background-color: #242424;
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 15px;
    cursor: pointer;
    transition: background-color 0.2s;
}
.playlist:hover {
    background-color: #2a2a2a;
}
.playlist-title {
    font-weight: bold;
    margin-bottom: 5px;
}
.playlist-songs {
    color: #b3b3b3;
    font-size: 14px;
}
.main-content {
    flex-grow: 1;
    padding: 20px;
    overflow-y: auto;
}

.content-header {
    display: flex;
```

```
    align-items: center;
    margin-bottom: 30px;
}

.content-title {
    font-size: 24px;
    font-weight: bold;
}

.song-list {
    width: 100%;
    border-collapse: collapse;
}
.song-list th {
    text-align: left;
    padding: 10px;
    color: #b3b3b3;
    border-bottom: 1px solid #282828;
}

.song-list tr:hover {
    background-color: rgba(255, 255, 255, 0.1);
}

.song-list td {
    padding: 10px;
}
.song-number {
    width: 40px;
    color: #b3b3b3;
}
.song-info {
```

```
        display: flex;
        align-items: center;
    }
    .song-thumbnail {
        width: 40px;
        height: 40px;
        border-radius: 4px;
        margin-right: 15px;
        object-fit: cover;
    }

    .song-title {
        font-weight: 500;
    }

    .song-artist {
        color: #b3b3b3;
        font-size: 14px;
    }

    .song-album {
        color: #b3b3b3;
    }
    .song-duration {
        color: #b3b3b3;
    }
player {
        position: fixed;
        bottom: 0;
        left: 0;
        right: 0;
        height: 90px;
```

```
    background-color: #181818;
    border-top: 1px solid #282828;
    display: flex;
    align-items: center;
    padding: 0 20px;
}
.current-song {
    width: 30%;
    display: flex;
    align-items: center;
}
.current-thumbnail {
    width: 56px;
    height: 56px;
    border-radius: 4px;
    margin-right: 15px;
    object-fit: cover;
}
.controls {
    width: 40%;
    display: flex;
    flex-direction: column;
    align-items: center;
}
.control-buttons {
    display: flex;
    align-items: center;
    margin-bottom: 10px;
}
.control-button {
    background: none;
    border: none;
```

```
    color: #b3b3b3;
    font-size: 16px;
    cursor: pointer;
    margin: 0 10px;
    transition: color 0.2s;
}
.control-button:hover {
    color: #fff;
}
.play-pause {
    width: 32px;
    height: 32px;
    border-radius: 50%;
    background-color: #fff;
    color: #000;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 14px;
}
.progress {
    width: 100%;
    display: flex;
    align-items: center;
}
.progress-time {
    color: #b3b3b3;
    font-size: 12px;
    min-width: 40px;
}

.progress-bar {
```

```
    flex-grow: 1;
    height: 4px;
    background-color: #535353;
    border-radius: 2px;
    margin: 0 10px;
    cursor: pointer;
    position: relative;
}
.progress-current {
    height: 100%;
    background-color: #b3b3b3;
    border-radius: 2px;
    width: 30%;
}
.progress-bar:hover .progress-current {
    background-color: #1DB954;
}
.volume {
    width: 30%;
    display: flex;
    align-items: center;
    justify-content: flex-end;
}
.volume-icon {
    color: #b3b3b3;
    margin-right: 10px;
}
.volume-bar {
    width: 100px;
    height: 4px;
    background-color: #535353;
    border-radius: 2px;
```

```
        cursor: pointer;
        position: relative;
    }
    .volume-current {
        height: 100%;
        background-color: #b3b3b3;
        border-radius: 2px;
        width: 70%;
    }

    .volume-bar:hover .volume-current {
        background-color: #1DB954;
    }
    .liked {
        color: #1DB954;
    }
    .song-list tr.playing {
        background-color: rgba(29, 185, 84, 0.2);
    }
    .song-list tr.playing .song-number,
.song-list tr.playing .song-title,
    .song-list tr.playing .song-artist {
        color: #1DB954;
    }
    .hidden {
        display: none;
    }
    @media (max-width: 768px) {
        .sidebar {
            width: 80px;
            padding: 10px;
        }
```

```
            .nav-link span,
            .library-title,
            .playlist-songs {
                display: none;
            }
            .current-song .song-artist {
                display: none;
            }
            .volume {
                display: none;
            }
        }
    </style>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
</head>
<body>
    <div class="sidebar">
<div class="logo">Music Player</div>
        <div class="nav-links">
            <div class="nav-link active" data-section="home">
                <i class="fas fa-home"></i>
                <span>Home</span>
            </div>
            <div class="nav-link" data-section="search">
                <i class="fas fa-search"></i>
                <span>Search</span>
            </div>
        </div>
        <div class="library">
            <div class="library-header">
                <div class="library-title">YOUR LIBRARY</div>
```

```
        </div>
        <div class="nav-link" data-section="liked">
            <i class="fas fa-heart"></i>
            <span>Liked Songs</span>
        </div>
        <div class="playlist" data-playlist="1">
            <div class="playlist-title">My Playlist #1</div>
            <div class="playlist-songs">10 songs</div>
        </div>
        <div class="playlist" data-playlist="2">
            <div class="playlist-title">Chill Vibes</div>
            <div class="playlist-songs">8 songs</div>
        </div>
        <div class="playlist" data-playlist="3">
            <div class="playlist-title">Workout Mix</div>
            <div class="playlist-songs">12 songs</div>
        </div>
    </div>
  </div>  <div class="main-content">
<div id="home-section">
        <div class="content-header">
            <div class="content-title">Welcome Back!</div>
        </div>
        <div class="playlist" data-playlist="1">
            <div class="playlist-title">My Playlist #1</div>
            <div class="playlist-songs">10 songs</div>
        </div>
        <div class="playlist" data-playlist="2">
            <div class="playlist-title">Chill Vibes</div>
            <div class="playlist-songs">8 songs</div>
        </div>
        <div class="playlist" data-playlist="3">
```

```
            <div class="playlist-title">Workout Mix</div>
            <div class="playlist-songs">12 songs</div>
        </div>
    </div>

    <div id="search-section" class="hidden">
        <div class="content-header">
            const allSongs = [
        {
            id: 1,
            title: "Sunny Day",
            artist: "The Soundwaves",
            album: "Summer Vibes",
            duration: "3:45",
            thumbnail: "/api/placeholder/40/40?text=SD",
            audio: "https://samplelib.com/lib/preview/mp3/sample-3s.mp3",
            liked: false
        },
        {
            id: 2,
            title: "Ocean Breeze",
album: "Beach Life",
            duration: "4:12",
            thumbnail: "/api/placeholder/40/40?text=OB",
            audio: "https://samplelib.com/lib/preview/mp3/sample-6s.mp3",
            liked: true
        },
        {
            id: 3,
            title: "City Lights",
            artist: "Urban Soul",
            album: "Midnight Drive",
```

```
            duration: "3:30",
            thumbnail: "/api/placeholder/40/40?text=CL",
            audio: "https://samplelib.com/lib/preview/mp3/sample-9s.mp3",
            liked: false
        },
        {
            id: 4,
            title: "Mountain Echo",
            artist: "Nature's Call",
            album: "Wilderness",
            duration: "5:18",
                let shuffle = false;
    let repeat = false;

    // Initialize app
    function init() {
        // Set up event listeners
        playPauseBtn.addEventListener('click', togglePlayPause);
        prevBtn.addEventListener('click', playPrevious);
        nextBtn.addEventListener('click', playNext);
        shuffleBtn.addEventListener('click', toggleShuffle);
        repeatBtn.addEventListener('click', toggleRepeat);
        audioPlayer.addEventListener('timeupdate', updateProgress);
audioPlayer.addEventListener('ended', handleSongEnd);
        progressBar.addEventListener('click', setProgress);
        volumeBar.addEventListener('click', setVolume);

        // Navigation
        navLinks.forEach(link => {
            link.addEventListener('click', () => {
                // Remove active class from all links
                navLinks.forEach(l => l.classList.remove('active'));
```

```
            // Add active class to clicked link
            link.classList.add('active');

            // Hide all sections
            sections.forEach(section => section.classList.add('hidden'));

            // Show selected section
            const section = link.getAttribute('data-section');
            document.getElementById(${section}-section).classList.remove('hidden');

            if (section === 'liked') {
                loadLikedSongs();
            }
        });
    });

    // Playlist click handlers
    playlistElements.forEach(playlist => {
        playlist.addEventListener('click', () => {
            const playlistId = playlist.getAttribute('data-playlist');
            loadPlaylist(playlistId);

            // Hide all sections and show playlist section
            sections.forEach(section => section.classList.add('hidden'));
            document.getElementById('playlist-section').classList.remove('hidden');
// Remove active class from all links
            navLinks.forEach(l => l.classList.remove('active'));
        });
    });

    // Initialize liked songs
    loadLikedSongs();
```

```
    }

    // Toggle play/pause
    function togglePlayPause() {
        if (currentPlaylist.length === 0) return;

        if (isPlaying) {
            audioPlayer.pause();
            playIcon.classList.remove('fa-pause');
            playIcon.classList.add('fa-play');
        } else {
            audioPlayer.play();
            playIcon.classList.remove('fa-play');
            playIcon.classList.add('fa-pause');
        }

        isPlaying = !isPlaying;
    }

    // Play previous song
    function playPrevious() {
        if (currentPlaylist.length === 0) return;

        currentSongIndex = (currentSongIndex - 1 + currentPlaylist.length) %
currentPlaylist.length;
        playSong(currentPlaylist[currentSongIndex]);
    }
// Play next song
    function playNext() {
        if (currentPlaylist.length === 0) return;

        if (shuffle) {
```

```
        currentSongIndex = Math.floor(Math.random() * currentPlaylist.length);
    } else {
        currentSongIndex = (currentSongIndex + 1) % currentPlaylist.length;
    }

    playSong(currentPlaylist[currentSongIndex]);
}

// Handle song end
function handleSongEnd() {
    if (repeat) {
        audioPlayer.currentTime = 0;
        audioPlayer.play();
    } else {
        playNext();
    }
}

// Toggle shuffle
function toggleShuffle() {
    shuffle = !shuffle;
    shuffleBtn.style.color = shuffle ? '#1DB954' : '#b3b3b3';
}

// Toggle repeat
function toggleRepeat() {
    repeat = !repeat;
    repeatBtn.style.color = repeat ? '#1DB954' : '#b3b3b3';
}
// Update progress bar
function updateProgress() {
    const duration = audioPlayer.duration;
```

```
    const currentTime = audioPlayer.currentTime;

    if (duration) {
        const progressPercent = (currentTime / duration) * 100;
        progressCurrent.style.width = ${progressPercent}%;

        // Update time displays
        currentTimeEl.textContent = formatTime(currentTime);
        totalTimeEl.textContent = formatTime(duration);
    }
}

// Set progress when clicked
function setProgress(e) {
    const width = this.clientWidth;
    const clickX = e.offsetX;
    const duration = audioPlayer.duration;

    audioPlayer.currentTime = (clickX / width) * duration;
}

// Set volume when clicked
function setVolume(e) {
    const width = this.clientWidth;
    const clickX = e.offsetX;
    const volume = clickX / width;

    audioPlayer.volume = volume;
    volumeCurrent.style.width = ${volume * 100}%;
}
// Format time in MM:SS        // Create song row
function createSongRow(song, index) {
```

```
const row = document.createElement('tr');
row.innerHTML = `
    <td class="song-number">${index}</td>
    <td>
        <div class="song-info">
            <img src="${song.thumbnail}" alt="${song.title}" class="song-thumbnail">
            <div>
                <div class="song-title">${song.title}</div>
                <div class="song-artist">${song.artist}</div>
            </div>
        </div>
    </td>
    <td class="song-album">${song.album}</td>
    <td class="song-duration">${song.duration}</td>
    <td>
        <button class="control-button like-button ${song.liked ? 'liked' : ''}">
            <i class="fas fa-heart"></i>
        </button>
    </td>
`;

// Add click event to play song
row.addEventListener('click', () => {
```

CHAPTER 9

OUTPUT SCREENS

9.1 USER HOME PAGE

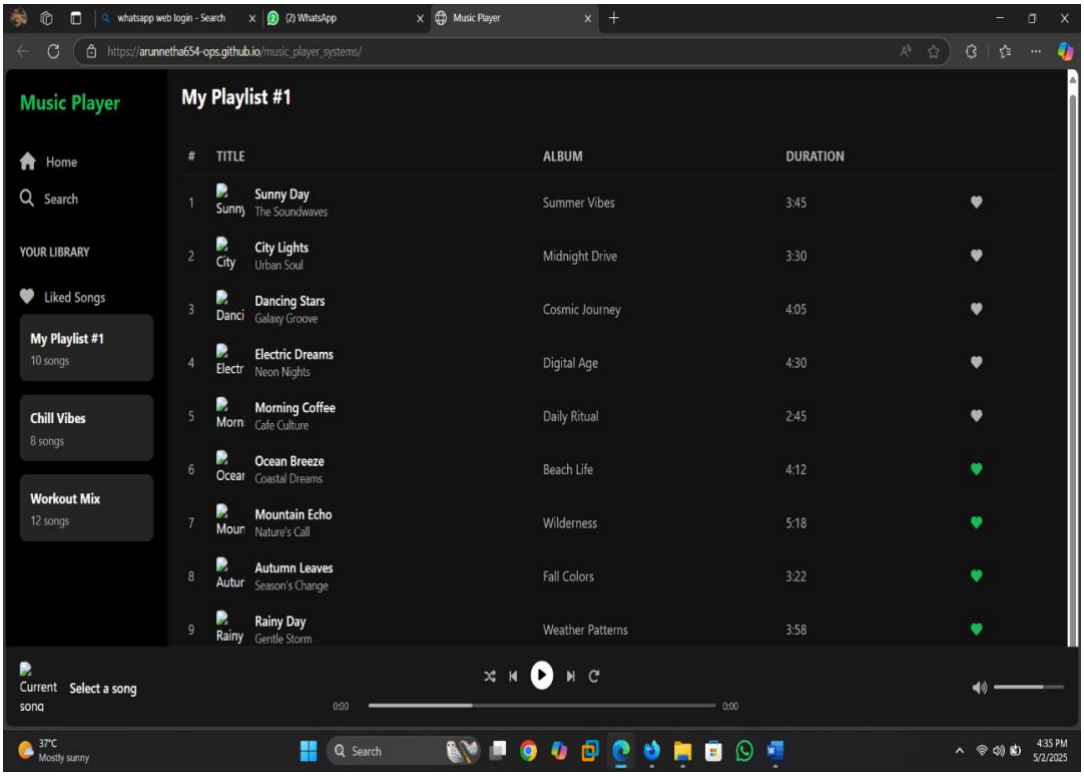

Figure no 9.1 *User Home Page*

9.2 MY PLAY LIST

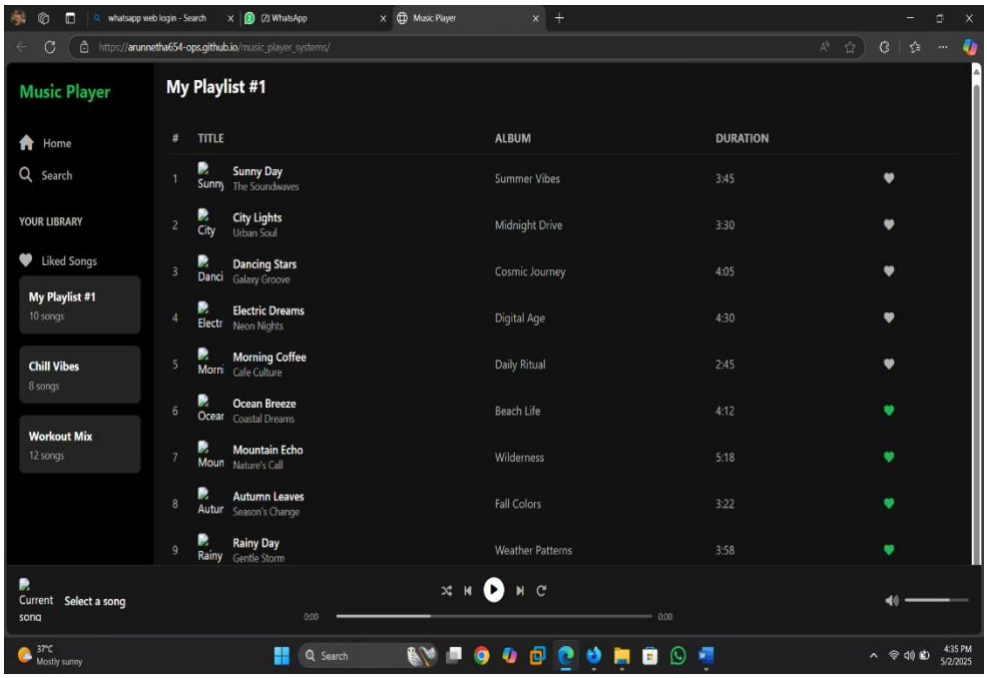

Figure no 9.2 *My Play List*

9.3 CHILL VIBES

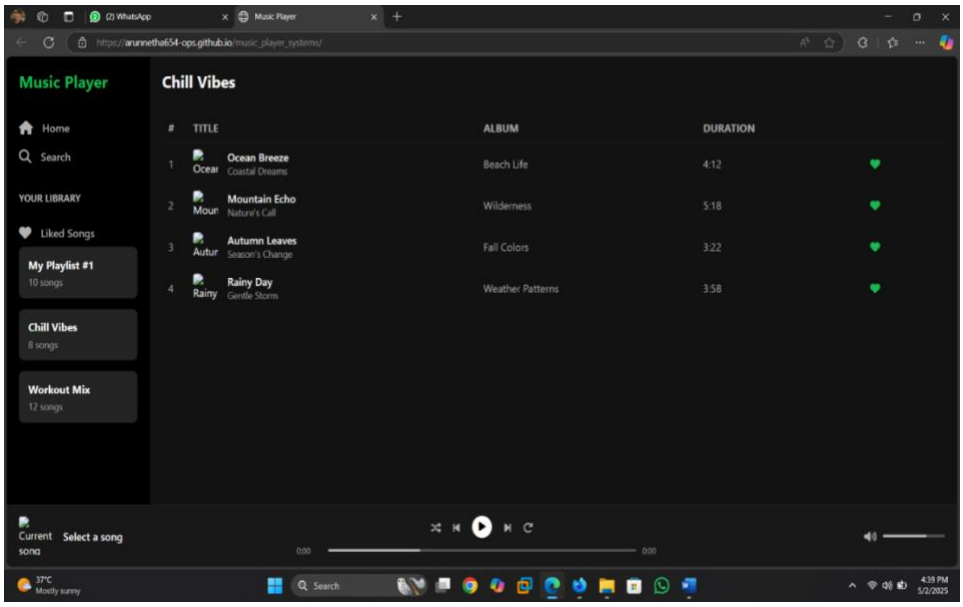

Figure no 9.3 *Chill Vibes*

9.4 USER WORK OUT MIX

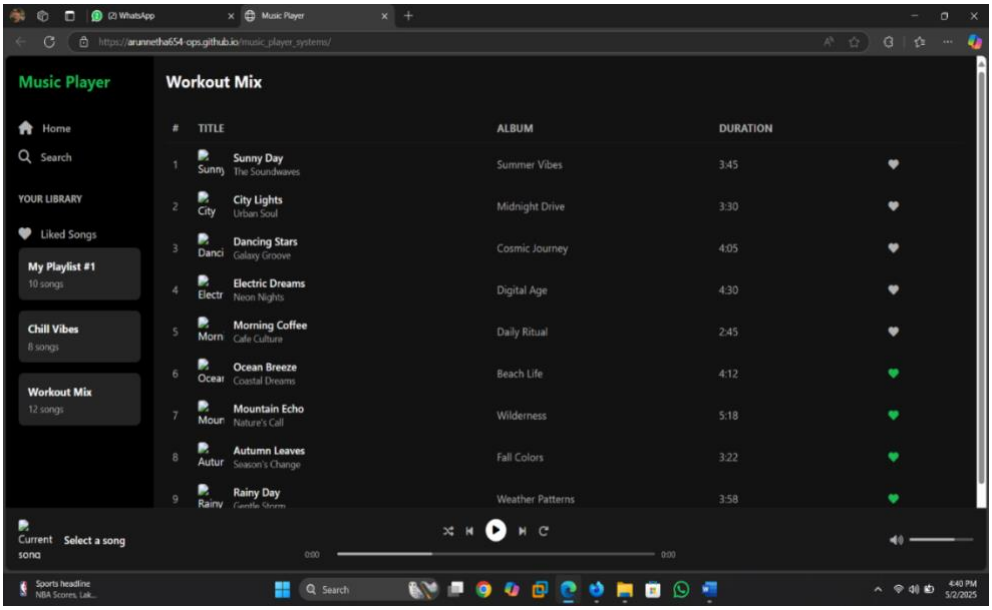

Figure no 9.*4 User Work Out Mix*

9.5 USER SEARCH PAGE

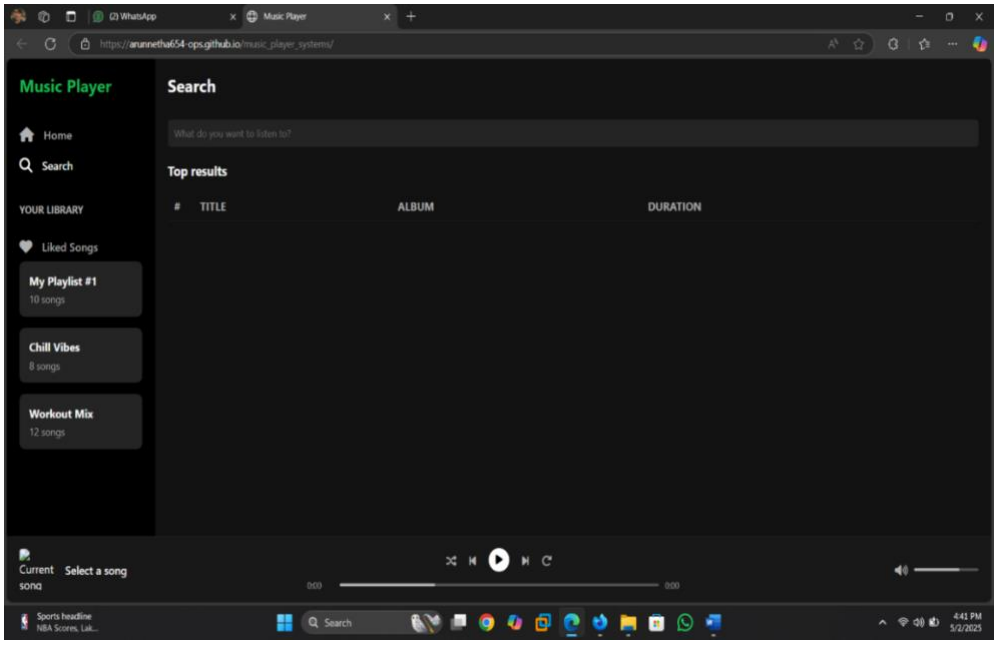

Figure no 9.5 *User Search page*

9.6 USER LIKED SONGS

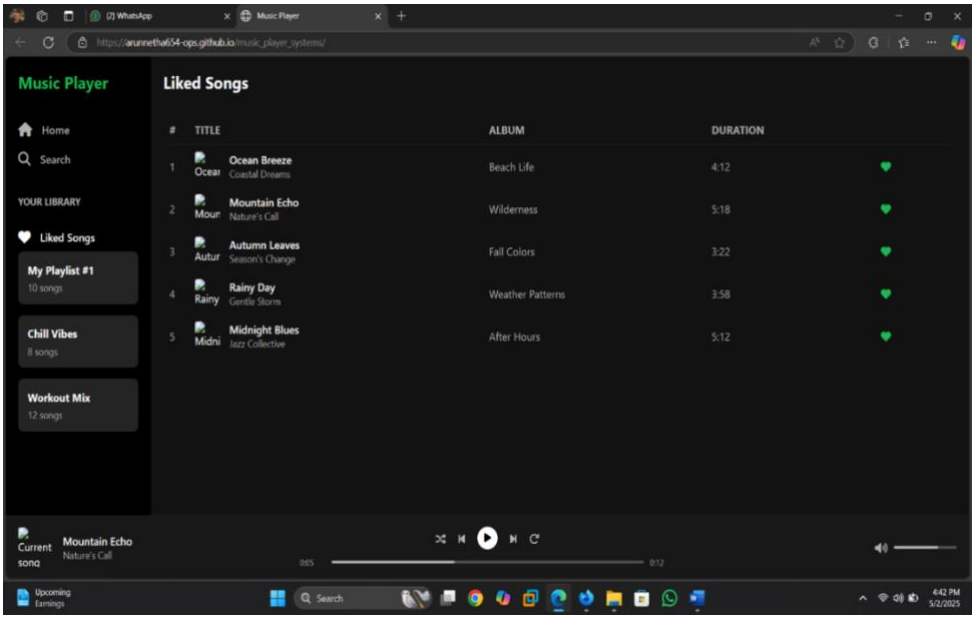

Figure no 9.6 *User Liked Songs*

CHAPTER 10

SYSTEM TESTING

10.1 UNIT TESTING

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..

10.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, 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.

10.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available specified by the busines and technical requirements, system documents, available manuals.

10.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

CHAPTER 11

CONCLUSION

The Music Player System developed successfully meets the intended goals of providing users with an intuitive, user-friendly platform to play, manage, and organize their music. The system allows basic functionalities such as play, pause, stop, next, previous, and volume control, along with playlist management and song metadata display. It offers a seamless audio experience and demonstrates how multimedia applications can be effectively designed using programming tools and frameworks. This project not only strengthened our understanding of audio processing and user interface design but also highlighted the importance of efficient file handling, responsive UI, and real-time control in multimedia systems. With future enhancements like online streaming, lyrics integration, and voice command support, the Music Player System can evolve into a more advanced and interactive media platform.

FUTURE ENHANCEMENTS

To improve and expand the Music Player System, the following enhancements can be considered:

Online Streaming Support: Integrate streaming from online platforms like Spotify, YouTube, or SoundCloud.

Equalizer Settings: Add a customizable audio equalizer for better sound control and user preference.

Lyrics Display: Fetch and display real-time song lyrics using online APIs.

Voice Command Integration: Implement voice control using speech recognition for hands-free operation.

Smart Playlist Generation: Use AI to suggest or auto-generate playlists based on listening history and mood detection.

BIBLIOGRAPHY

1.HTML & CSS: Design and Build Websites – Jon Duckett
(Reference for designing web pages and understanding responsive layout principles)

2.JavaScript: The Definitive Guide – David Flanagan
(Used to understand and implement interactivity on the website)

3.W3Schools – (For HTML, CSS, and JavaScript tutorials and examples)

4.MDN Web Docs (Mozilla Developer Network)
(Technical references and best practices for web development)

5.GitHub Documentation– (For version control and GitHub Pages deployment)

6.Stack Overflow – (For troubleshooting errors and improving code snippets)

7.Free CodeCamp – (Helpful for frontend development tutorials and real-world project examples)