A PROJECT REPORT ON

MUSIC APPLICATION

A Report Submitted to

IIDT - Blackbuck Engineers Pvt. Ltd

Submitted by

Team Members
Name & Roll numbers

I.VENKATESH

(Y21CSE279034)

Y. MOULI ARAVIND

(Y21CSE279128)

B. SIVA

(Y21CSE279004)

P. PREM CHANDRA

(Y21CSE279098)

G. KONDA REDDY

(Y21CSE279034)

Acknowledgement

I would like to express my sincere gratitude to everyone who supported me throughout the development of this music application. Their guidance and encouragement have been invaluable. First and foremost, I would like to thank my mentor, for their expert advice and insightful feedback. Their support and patience have significantly contributed to the success of this project. I am also grateful to my colleagues and friends for their continuous encouragement and for providing a collaborative environment that fostered creativity and innovation. Special thanks to the developers and contributors of open-source libraries and frameworks used in this project, including [list of major libraries and frameworks], for providing robust tools that made the development process smoother. Finally, I would like to thank my family for their unwavering support and understanding throughout this journey. Their belief in me has been a source of motivation and strength. This project would not have been possible without the combined efforts of all these individuals. I am truly grateful for their contributions

Introduction

Overview
This document provides a comprehensive guide to building a music application using HTML, CSS, and JavaScript for the frontend, and Node.js with Express for the backend. The application allows users to browse, search, and play songs, as well as manage playlists.

Purpose
The purpose of this documentation is to provide a theoretical framework and practical guidance for developers to create a fully functional music application. It covers setup, usage, configuration, architecture, development, deployment, maintenance, and contribution guidelines.

Scope
This documentation covers:

- Frontend development with HTML, CSS, and JavaScript.
- Backend development with Node.js and Express.
- Integration between frontend and backend.
- Deployment and maintenance of the application.

Audience
This documentation is intended for web developers, software engineers, and anyone interested in building a music application.

Getting Started

Prerequisites

- Node.js installed
- Basic knowledge of HTML, CSS, and JavaScript
- Basic understanding of RESTful APIs

Installation

1. **Clone the repository**:

bash

Copy code

```
git clone https://github.com/yourusername/music-app.git
cd music-app
```

2. **Install backend dependencies**:

bash

Copy code

```
cd backend
npm install
```

Setup

1. **Backend Setup**:
    - Create a .env file in the backend directory and add the following:

makefile

Copy code

```
PORT=5000
MONGO_URI=your_mongodb_uri
JWT_SECRET=your_jwt_secret
```

2. **Frontend Setup**:
    - No additional setup required for HTML, CSS, and JavaScript.

Quick Start Guide

1. **Run the backend server**:

```
bash
Copy code
cd backend
npm start
```

2. **Open index.html in your browser** to view the frontend.

Usage

Basic Usage

- **Homepage**: Displays featured songs and playlists.
- **Search**: Allows users to search for songs.
- **Playlists**: Users can create and manage playlists.
- **Login/Register**: Users can log in or register to access personalized features.

Advanced Usage

- **Admin Panel**: For managing songs and user accounts (if implemented).
- **User Profiles**: Users can view and edit their profiles.

Examples

- **Adding a song to a playlist**: Navigate to the song, click "Add to Playlist", and select the desired playlist.
- **Playing a song**: Click on a song in the list to start playing.

Configuration

Default Settings

- Default port for the backend server: 5000
- Default database: MongoDB

Customization

- **CSS**: Modify styles/style.css to customize the appearance.
- **HTML**: Update index.html to change the structure.

Environment Variables

- **PORT**: Port for the backend server.
- **MONGO_URI**: MongoDB connection string.
- **JWT_SECRET**: Secret key for JWT authentication.

API Documentation

Endpoints

- **GET /api/songs**: Retrieve all songs.
- **POST /api/songs**: Add a new song.
- **GET /api/playlists**: Retrieve all playlists.
- **POST /api/playlists**: Create a new playlist.
- **POST /api/auth/login**: User login.
- **POST /api/auth/register**: User registration.

Request and Response Formats

- **JSON** format for both requests and responses.

Authentication

- Use JWT for authentication.
- Include JWT token in the Authorization header for protected routes.

Architecture

System Architecture

- **Frontend**: HTML, CSS, JavaScript

- **Backend**: Node.js, Express, MongoDB

FRONTEND

CODE:
HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="musicplayer.css" />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.1/css/all.min.css"
    />
    <script defer src="musicplayer.js"></script><!-- Script for music player functionality -->
    <title>Music Player</title>
  </head>
  <body>
    <div class="background">
      <img src="background music 2.jpg" id="bg_img" /><!-- Background image for the player -->
    </div>
<div class="container">
      <!--===============Player Image================= -->
      <div class="player_img">
        <img src="media/image-1.jpg" id="cover" class="active" /><!-- Cover image of the currently
playing song -->
      </div>
      <!--============Player Content -->
      <h4 id="music_title">BGM &#128158;Of Premalu</h4><!-- Title of the currently playing
song -->
      <h5 id="musric_artist">Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan</h5><!-- Artist(s)
of the currently playing song -->


      <!--==============Player Progress & Timmer -->
      <div class="player_progress" id="player_progress">
        <div class="progress" id="progress">
          <div class="music_duration">
```

```
            <span id="current_time">0:00</span><!-- Current playback time of the song -->
            <span id="duration">0:00</span><!-- Total duration of the song -->
          </div>
        </div>
      </div>
      <!--===============Player Controllers -->
      <div class="player_controls">
        <i class="fa-solid fa-shuffle" title="shuffle" id="shuff"></i><!-- Shuffle button -->
        <i class="fa-solid fa-heart" title="like" id="heart"></i><!-- Like (heart) button -->
        <i class="fa-solid fa-backward" title="Previous" id="prev"></i><!-- Previous track button -->
        <i class="fa-solid fa-play" title="Play" id="play"></i><!-- Play/Pause button -->
        <i class="fa-solid fa-forward" title="Next" id="next"></i><!-- Next track button -->
        <i class="fa-solid fa-share-nodes"title="share" id="shar"></i><!-- Share button -->
        <i class="fa-solid fa-repeat" title="repeat" id="rep"></i> <!-- Repeat button -->
    </div>
      </div>
    </div>
  </body>
</html>
```

CSS

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&family=Ruda:wght@
400;600;700&display=swap");

* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
body {
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  font-family: "poppins", sans-serif;
  font-size: 0.8rem;
  overflow: hidden;
}
.background {
  position: fixed;
  width: 100%;
  height: 100%;
  z-index: -1;
}
```

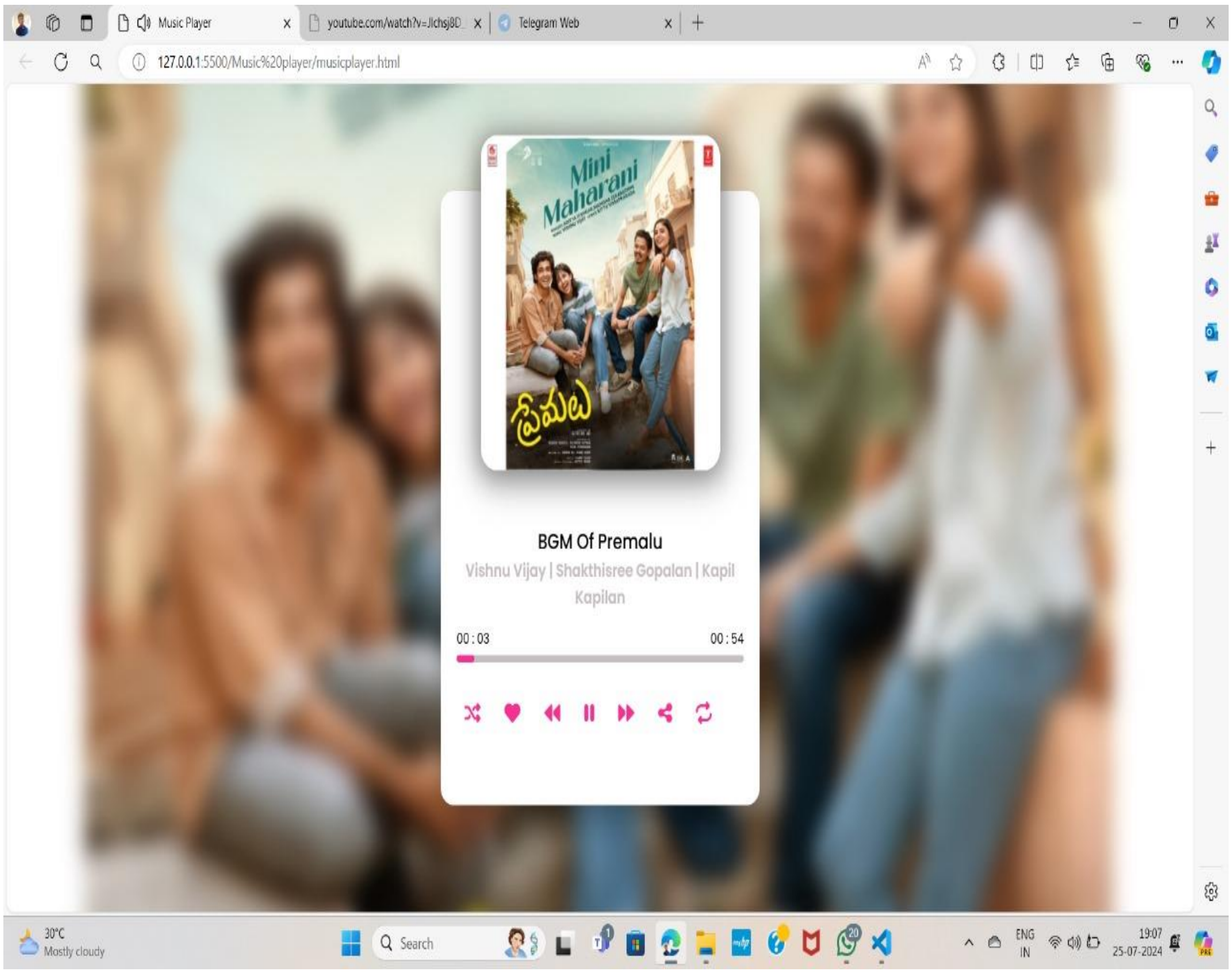
```
.background img {
  position: absolute;
  width: 100%;
  height: 100%;
  object-fit: cover;
  filter: blur(10px);
  transform: scale(1.1);
}
.container {
  background-color: #fff;
  width: 400px;
  height: 550px;
  border-radius: 1rem;
  box-shadow: 0 15px 30px rgba(0, 0, 0, 0.3);
}
.player_img {
  width: 300px;
  height: 300px;
  position: relative;
  top: -50px;
  left: 50px;
}
.player_img img {
  object-fit: cover;
  height: 0;
  width: 0;
  opacity: 0;
  box-shadow: 0 5px 30px 5px rgba(0, 0, 0, 0.5);
  border-radius: 20px;
}
.player_img img.active {
  width: 100%;
  height: 100%;
  opacity: 1;
}
h4 {
  font-size: 1.2rem;
  text-align: center;
  font-weight: 500;
}
h5 {
  font-size: 1rem;
  text-align: center;
  color: #c6bfbf;
}
.player_progress {
```

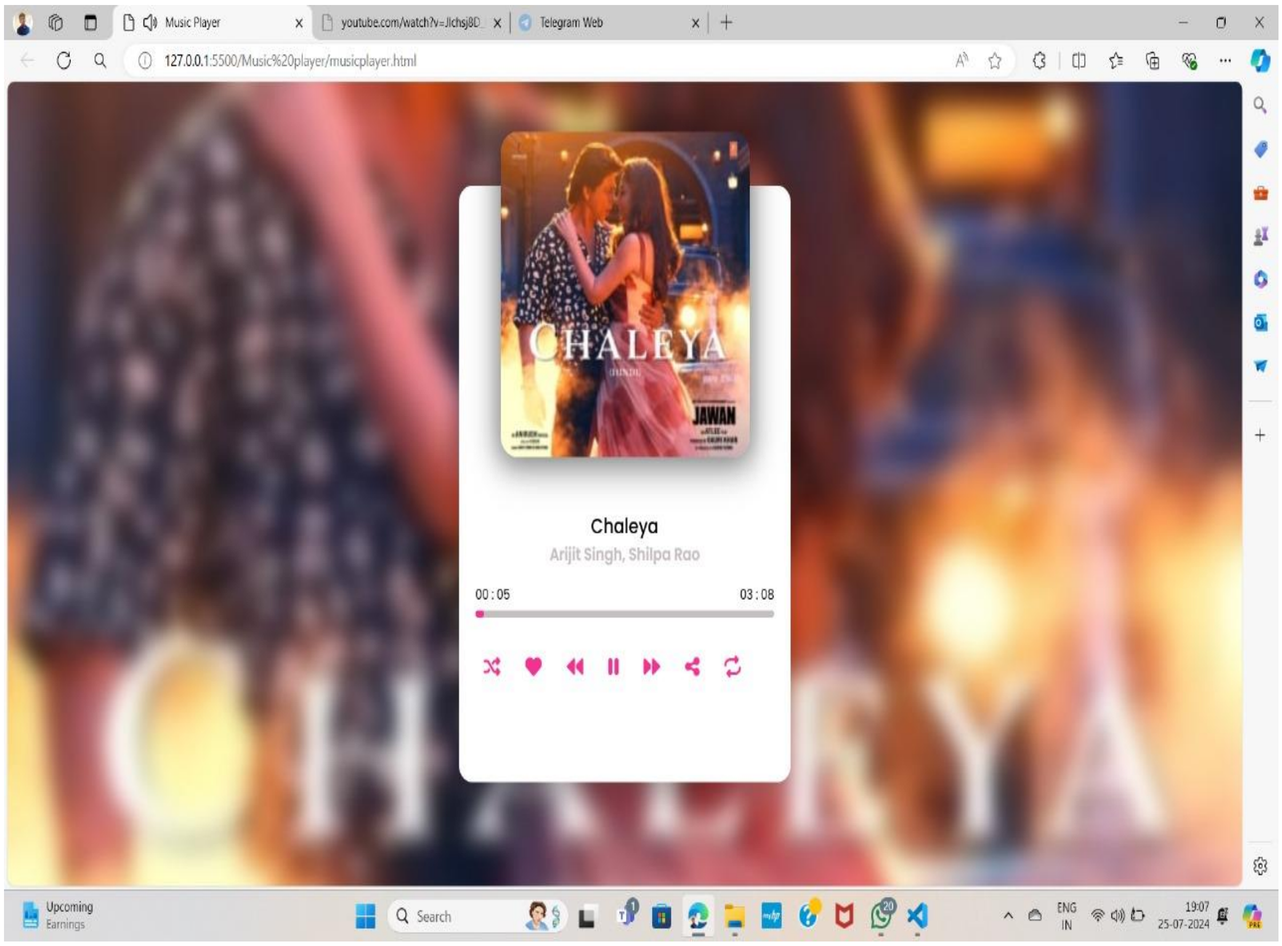
```css
  background-color: #c6bfbf;
  border-radius: 5px;
  height: 6px;
  width: 90%;
  margin: 40px 20px 35px;
  position: relative;
  cursor: pointer;
}
#progress {
  -webkit-appearance: none;
  appearance: none;
  width: 100%;
  height: 6px;
  background: #f53192;
  border-radius: 4px;
  cursor: pointer;
  margin: 40px 0;
}

#progress::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  background: #fff;
  width: 30px;
  height: 30px;
  border-radius: 50%;
  border: 8px solid #f53192;
  box-shadow: 0 5px 10px rgba(0, 0, 0, 0.2); /* Adjusted box shadow */
  cursor: pointer;
}

#progress:hover::-webkit-slider-thumb {
  background: #f53192; /* Change thumb color on hover */
}

#progress:focus::-webkit-slider-thumb {
  box-shadow: 0 0 0 2px #f53192, 0 0 0 4px rgba(245, 49, 146, 0.5); /* Focus style */
}
.music_duration {
  width: 100%;
  display: flex;
  justify-content: space-between;
  position: absolute;
  top: -25px;
}
.player_controls {
  display: flex;
  justify-content: center;
  align-items: center;
}
.fa-solid.fa-shuffle,
.fa-solid.fa-repeat {
  margin-right: 30px; /* Adjust as needed */
}

.fa-solid {
  font-size: 20px;
  color:  #f53192;
  cursor: pointer;
  margin-right: 30px;
  user-select: none;
  transition: all 0.3s ease-in;

}
.fa-solid:hover {
  filter: brightness(40%);
}
.play-button {
  font-size: 44px;
}
```

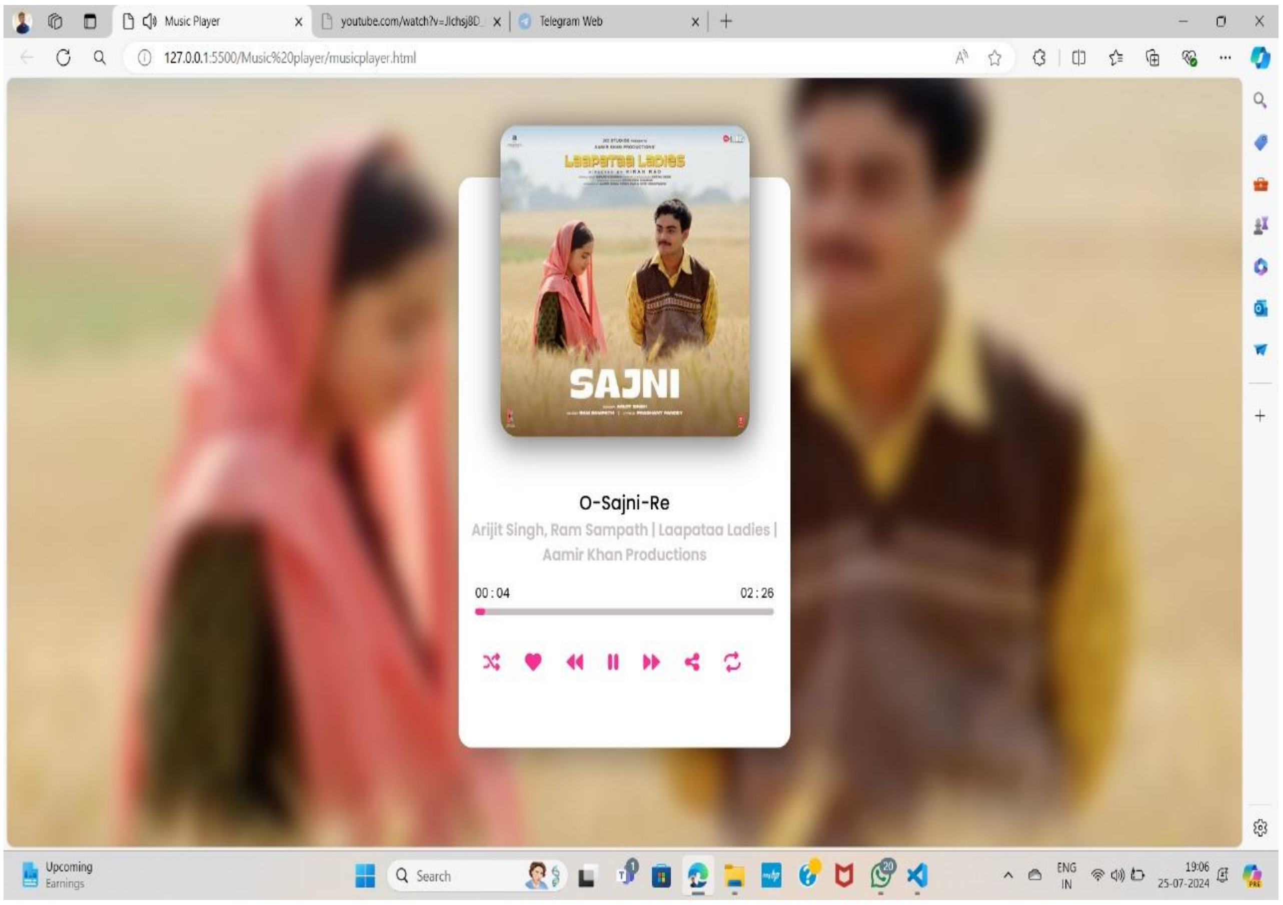
JAVASCRIPT:

```
"use strict";
const imgEl = document.getElementById("bg_img");
const imgCoverEl = document.getElementById("cover");
const musicTitleEl = document.getElementById("music_title");
const musicArtistEl = document.getElementById("musric_artist");
const playerProgressEl = document.getElementById("player_progress");
const progressEl = document.getElementById("progress");
const currentTimeEl = document.getElementById("current_time");
const durationEl = document.getElementById("duration");
const prevBtnEl = document.getElementById("prev");
const playvBtnEl = document.getElementById("play");
const nextvBtnEl = document.getElementById("next");
const shuffleBtnEl = document.getElementById("shuff");
const repeatBtnEl = document.getElementById("rep");
const heartBtnEl = document.getElementById("heart");
const shareBtnEl = document.getElementById("shar");

const songs = [
  {
    path: "media/song1.mp3",
    displayName: "BGM Of Premalu",
    cover: "media/image-1.jpg",
    artist: "Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan",
  },
  {
    path: "media/song2.mp3",
    displayName: "Suttamla Soosi",
    cover: "media/image-2.jpg",
    artist: "VishwakSen, Neha Shetty | Yuvan Shankar Raja",
  },
  {
    path: "media/song3.mp3",
    displayName: "Chaleya",
    cover: "media/image-3.jpg",
    artist: "Arijit Singh, Shilpa Rao",
  },
  {
    path: "media/song4.mp3",
    displayName: "Nadaniya",
    cover: "media/image-4.jpg",
    artist: "Akshath",
  },
  {
    path: "media/song5.mp3",
    displayName: "O-Sajni-Re",
    cover: "media/image-5.jpg",
    artist: "Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions",
  },
];

const music = new Audio();
let musicIndex = 0;
let isPlaying = false;
//================== Play Song  True or False====================
function togglePlay() {
  if (isPlaying) {
    pauseMusic();
  } else {
    playMusic();
  }
}
//================== Play Music====================
function playMusic() {
  isPlaying = true;
  playvBtnEl.classList.replace("fa-play", "fa-pause");
  playvBtnEl.setAttribute("title", "pause");
  music.play();
}
//================== Pause Music====================
function pauseMusic() {
  isPlaying = false;
  playvBtnEl.classList.replace("fa-pause", "fa-play");
  playvBtnEl.setAttribute("pause", "title");
  music.pause();
}
//================== Load Songs ====================
function loadMusic(songs) {
  music.src = songs.path;
  musicTitleEl.textContent = songs.displayName;
  musicArtistEl.textContent = songs.artist;
  imgCoverEl.src = songs.cover;
  imgEl.src = songs.cover;
}
//================== Change Music ====================
function changeMusic(direction) {
  musicIndex = musicIndex + direction + (songs.length % songs.length);
  loadMusic(songs[musicIndex]);
  playMusic();
}
//================== Set Progress ====================
function setProgressBar(e) {
  const width = playerProgressEl.clientWidth;
  const xValue = e.offsetX;
  music.currentTime = (xValue / width) * music.duration;
}
//================== Set Progress ====================
function updateProgressBar() {
  const { duration, currentTime } = music;
  const ProgressPercent = (currentTime / duration) * 100;
  progressEl.style.width = ${ProgressPercent}%;
  const formattime = (timeRanges) =>
    String(Math.floor(timeRanges)).padStart(2, "0");
  durationEl.textContent = `${formattime(duration / 60)} : ${formattime(
    duration % 60,
  )}`;
  currentTimeEl.textContent = `${formattime(currentTime / 60)} : ${formattime(
    currentTime % 60,
  )}`;
}
//================= Btn Events======================
const btnEvents = () => {
  playvBtnEl.addEventListener("click", togglePlay);
  nextvBtnEl.addEventListener("click", () => changeMusic(1));
  prevBtnEl.addEventListener("click", () => changeMusic(-1));
  //========= Progressbar==========================
  music.addEventListener("ended", () => changeMusic(1));
  music.addEventListener("timeupdate", updateProgressBar);
  playerProgressEl.addEventListener("click", setProgressBar);
};
//================= Btn Events========================
document.addEventListener("DOMContentLoaded", btnEvents);

// Event listener for auto-playing the next song
music.addEventListener("ended", () => {
    if (!isRepeat) {
      changeMusic(1); // Play next song when current song ends, unless in repeat mode
    }
  });
//============ Calling Load Music
loadMusic(songs[musicIndex]);
```

OUTPUT:


Music Player
youtube.com/watch?v=Jlchsj8D
Telegram Web
127.0.0.1:5500/Music%20player/musicplayer.html
BGM Of Premalu
Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan
00 : 03
00 : 54
Search
30°C
Mostly cloudy
19:07
25-07-2024


Music Player
youtube.com/watch?v=Jlchsj8D
Telegram Web
127.0.0.1:5500/Music%20player/musicplayer.html
Chaleya
Arijit Singh, Shilpa Rao
00 : 05
03 : 08
Search
Upcoming
Earnings
19:07
25-07-2024

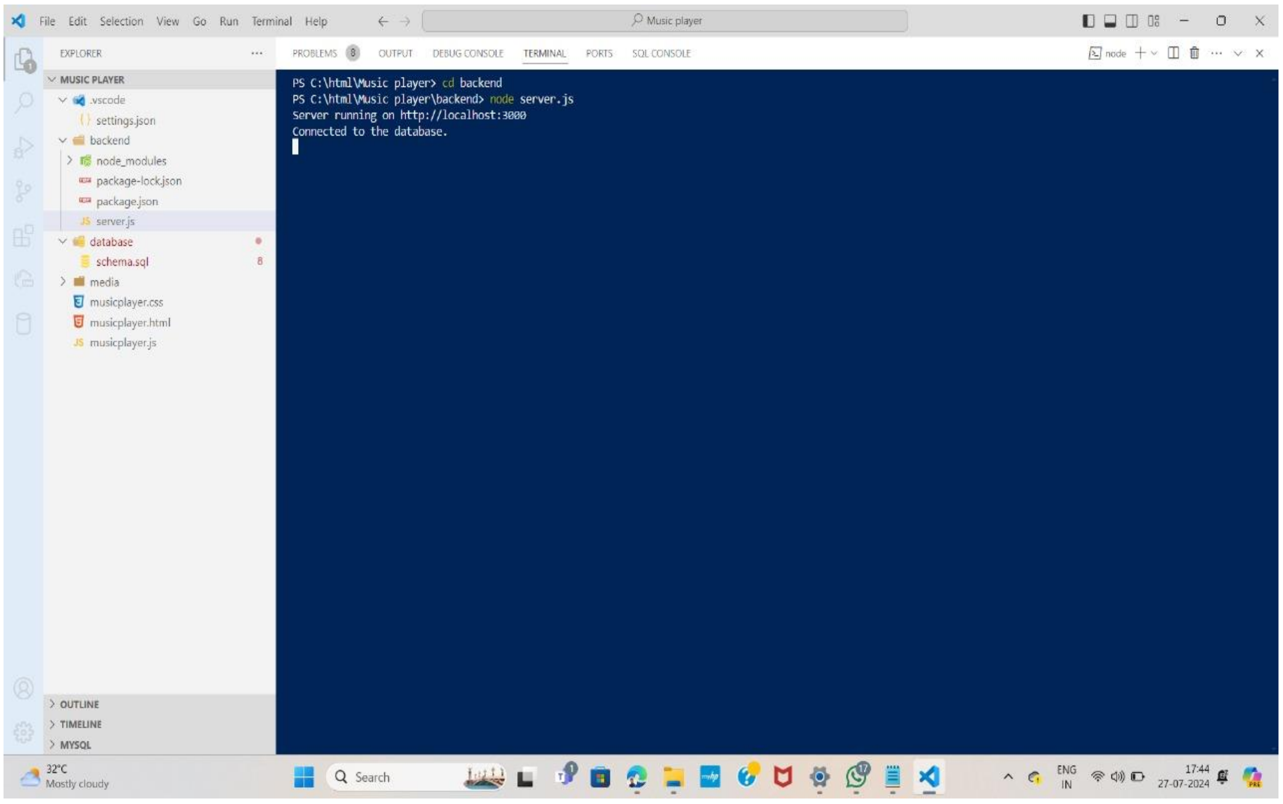

Music Player
youtube.com/watch?v=Jlchsj8D
Telegram Web
127.0.0.1:5500/Music%20player/musicplayer.html
O-Sajni-Re
Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions
00 : 04
02 : 26
Search
Upcoming
Earnings
19:06
25-07-2024

BACKEND
Code:

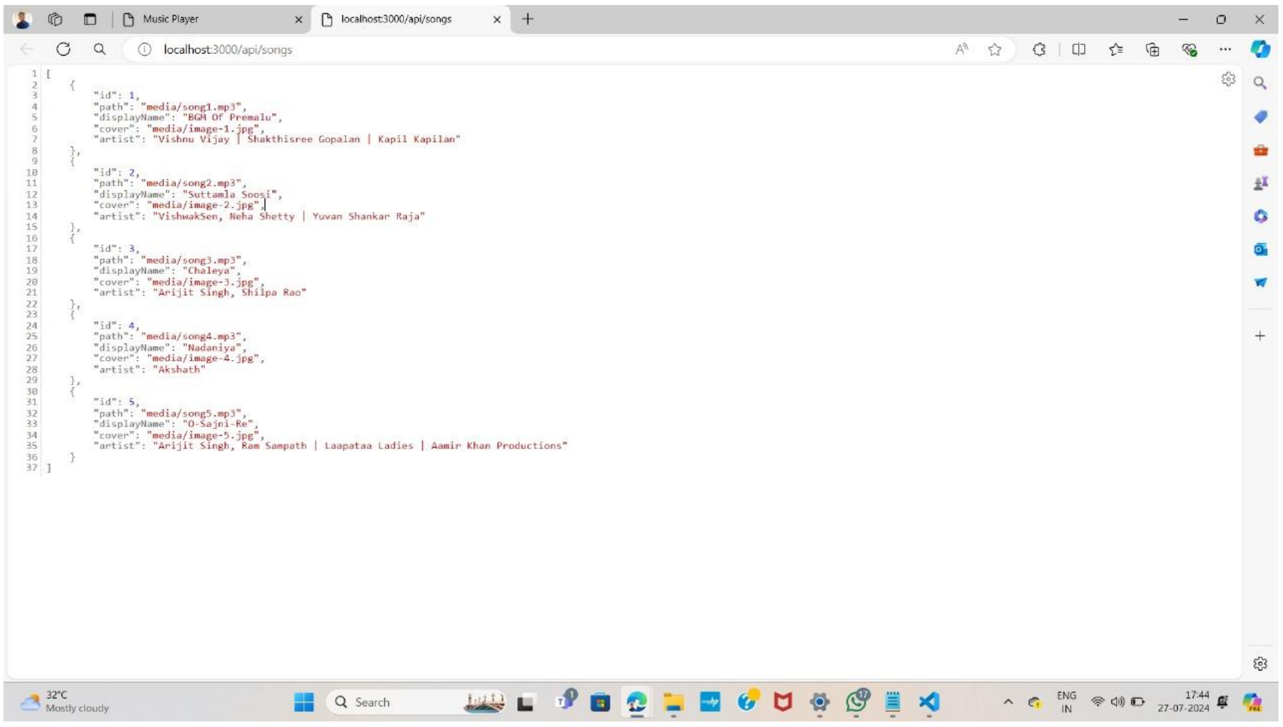
```
const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');

const app = express();
const port = 3000;

// Middleware
app.use(cors());
app.use(express.json());

// Database connection
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'Siva@2002', // Replace with your database password
  database: 'music_db' // Replace with your database name
});

// Connect to the database
db.connect(err => {
  if (err) {
    console.error('Database connection error:', err);
    return;
  }
  console.log('Connected to the database.');
});

// API endpoint to get songs
app.get('/api/songs', (req, res) => {
  db.query('SELECT * FROM songs', (err, results) => {
    if (err) {
      console.error('Error fetching songs:', err);
      res.status(500).json({ error: 'Failed to retrieve songs' });
      return;
    }
    res.json(results);
  });
});

// API endpoint to create a new song
app.post('/api/songs', (req, res) => {
  const { title, artist, album, genre, release_date } = req.body;

  if (!title || !artist || !album || !genre || !release_date) {
    return res.status(400).json({ error: 'All fields are required' });
  }

  const query = 'INSERT INTO songs (title, artist, album, genre, release_date) VALUES (?, ?, ?, ?,
?)';
  const values = [title, artist, album, genre, release_date];
  db.query(query, values, (err, result) => {
    if (err) {
      console.error('Error inserting song:', err);
      res.status(500).json({ error: 'Failed to add song' });
      return;
    }
    res.status(201).json({ id: result.insertId, title, artist, album, genre, release_date });
  });
});

// API endpoint to update a song
app.put('/api/songs/:id', (req, res) => {
  const songId = req.params.id;
  const { title, artist, album, genre, release_date } = req.body;

  const query = 'UPDATE songs SET title = ?, artist = ?, album = ?, genre = ?, release_date = ?
WHERE id = ?';
  const values = [title, artist, album, genre, release_date, songId];

  db.query(query, values, (err, result) => {
    if (err) {
      console.error('Error updating song:', err);
      res.status(500).json({ error: 'Failed to update song' });
      return;
    }
    if (result.affectedRows === 0) {
      res.status(404).json({ error: 'Song not found' });
      return;
    }
    res.json({ id: songId, title, artist, album, genre, release_date });
  });
});

// API endpoint to delete a song
app.delete('/api/songs/:id', (req, res) => {
  const songId = req.params.id;

  const query = 'DELETE FROM songs WHERE id = ?';
  const values = [songId];

  db.query(query, values, (err, result) => {
    if (err) {
      console.error('Error deleting song:', err);
      res.status(500).json({ error: 'Failed to delete song' });
      return;
    }
    if (result.affectedRows === 0) {
      res.status(404).json({ error: 'Song not found' });
      return;
    }
    res.status(204).end(); // No content response
  });
});
```

```
// Start the server
app.listen(port, () => {
  console.log(Server running on http://localhost:${port});
});
```

OUTPUT:



```
PS C:\html\Music player> cd backend
PS C:\html\Music player\backend> node server.js
Server running on http://localhost:3000
Connected to the database.
```



```
[
    {
        "id": 1,
        "path": "media/song1.mp3",
        "displayName": "BGM Of Premalu",
        "cover": "media/image-1.jpg",
        "artist": "Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan"
    },
    {
        "id": 2,
        "path": "media/song2.mp3",
        "displayName": "Suttamla Soosi",
        "cover": "media/image-2.jpg",
        "artist": "VishwakSen, Neha Shetty | Yuvan Shankar Raja"
    },
    {
        "id": 3,
        "path": "media/song3.mp3",
        "displayName": "Chaleya",
        "cover": "media/image-3.jpg",
        "artist": "Arijit Singh, Shilpa Rao"
    },
    {
        "id": 4,
        "path": "media/song4.mp3",
        "displayName": "Nadaniya",
        "cover": "media/image-4.jpg",
        "artist": "Akshath"
    },
    {
        "id": 5,
        "path": "media/song5.mp3",
        "displayName": "O-Sajni-Re",
        "cover": "media/image-5.jpg",
        "artist": "Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions"
    }
]
```

Components

- **Frontend Components**: Navigation bar, song list, player, search bar, playlists.
- **Backend Components**: Controllers, models, routes, middlewares.

Data Flow

1. User interacts with the frontend.
2. Frontend makes API requests to the backend.
3. Backend processes the request, interacts with the database, and sends a response.
4. Frontend updates the UI based on the response.

Design Decisions

- **Separation of Concerns**: Separate frontend and backend for modularity.
- **RESTful APIs**: Use RESTful principles for API design.
- **JWT Authentication**: Secure user authentication.

Naming Conventions

- **Files**: Use lowercase with hyphens for file names (e.g., song-controller.js).
- **Variables**: Use camelCase for variables and functions.
- **Constants**: Use UPPER_CASE for constants.

Coding Standards

- **Frontend**: Follow best practices for HTML, CSS, and JavaScript.
- **Backend**: Follow best practices for Node.js and Express.

Testing

- **Unit Tests**: Write unit tests for backend components using a testing framework like Mocha or Jest.
- **Integration Tests**: Test the integration between frontend and backend.

Debugging

- **Frontend**: Use browser developer tools.
- **Backend**: Use Node.js debugging tools and loggers.

Deployment

Deployment Process

1. **Frontend**: Deploy to a static hosting service (e.g., Netlify, GitHub Pages).
2. **Backend**: Deploy to a cloud provider (e.g., Heroku, AWS).

Environments

- **Development**: Local environment for development.
- **Staging**: Pre-production environment for testing.
- **Production**: Live environment for end-users.

Rollback Procedures

- **Version Control**: Use version control (e.g., Git) to revert to previous versions.
- **Backup**: Maintain database backups.

Maintenance

Updating

- Regularly update dependencies to the latest versions.
- Apply security patches promptly.

Backup and Restore

- Schedule regular database backups.
- Test restore procedures to ensure data integrity.

Monitoring

- Use monitoring tools (e.g., New Relic, Loggly) to monitor application performance and errors.

Contributing

How to Contribute

- Fork the repository.
- Create a new branch for your feature or bugfix.
- Submit a pull request with a detailed description of your changes.

Code of Conduct

- Follow the project's code of conduct.

Issue Reporting

- Use GitHub Issues to report bugs and request features.

Pull Requests

- Ensure your pull request passes all tests.
- Provide a clear and detailed description of your changes.

FAQ

Common Issues

- **Server not starting**: Check .env file and database connection.
- **Frontend not loading**: Check console for errors and ensure correct file paths.

Troubleshooting Tips

- Use browser developer tools for frontend issues.
- Check server logs for backend issues.

Appendices

Glossary

- **API**: Application Programming Interface
- **JWT**: JSON Web Token
- **REST**: Representational State Transfer

References

- Node.js Documentation
- Express Documentation
- MongoDB Documentation

Related Documents

- Frontend Development Best Practices
- Backend Development Best Practices

CONCLUSION:

A well-rounded music application has the potential to significantly impact how users discover and enjoy music, offering a personalized, convenient, and engaging experience. By continually innovating and addressing challenges, the app can maintain a competitive edge and drive long-term success in the dynamic music industry