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)

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

ARCHITECTURE:

```
+---------------------------+
|      Client Side          |
| (HTML/CSS/JavaScript)     |
|                           |
| +-----------------------+ |
| | Music Player UI       | |
| |  - Play/Pause         | |
| |  - Next/Prev          | |
| |  - Progress Bar       | |
| |  - Volume Control     | |
| +-----------------------+ |
|                           |
+-------------|-------------+
              |
              v
+-------------|-------------+
|      Server Side          |
| (Node.js/Express or Flask)|
|                           |
| +-----------------------+ |
| | API Endpoints         | |
| |  - /songs             | |
| |  - /artists           | |
| |  - /covers            | |
| +-----------------------+ |
|                           |
+-------------|-------------+
              |
              v
+-------------|-------------+
|      Database             |
| (MySQL/PostgreSQL/MongoDB)|
|                           |
| +-----------------------+ |
| | Tables:               | |
| |  - artists            | |
| |  - songs              | |
| |  - covers             | |
| +-----------------------+ |
|                           |
```

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 {
  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:

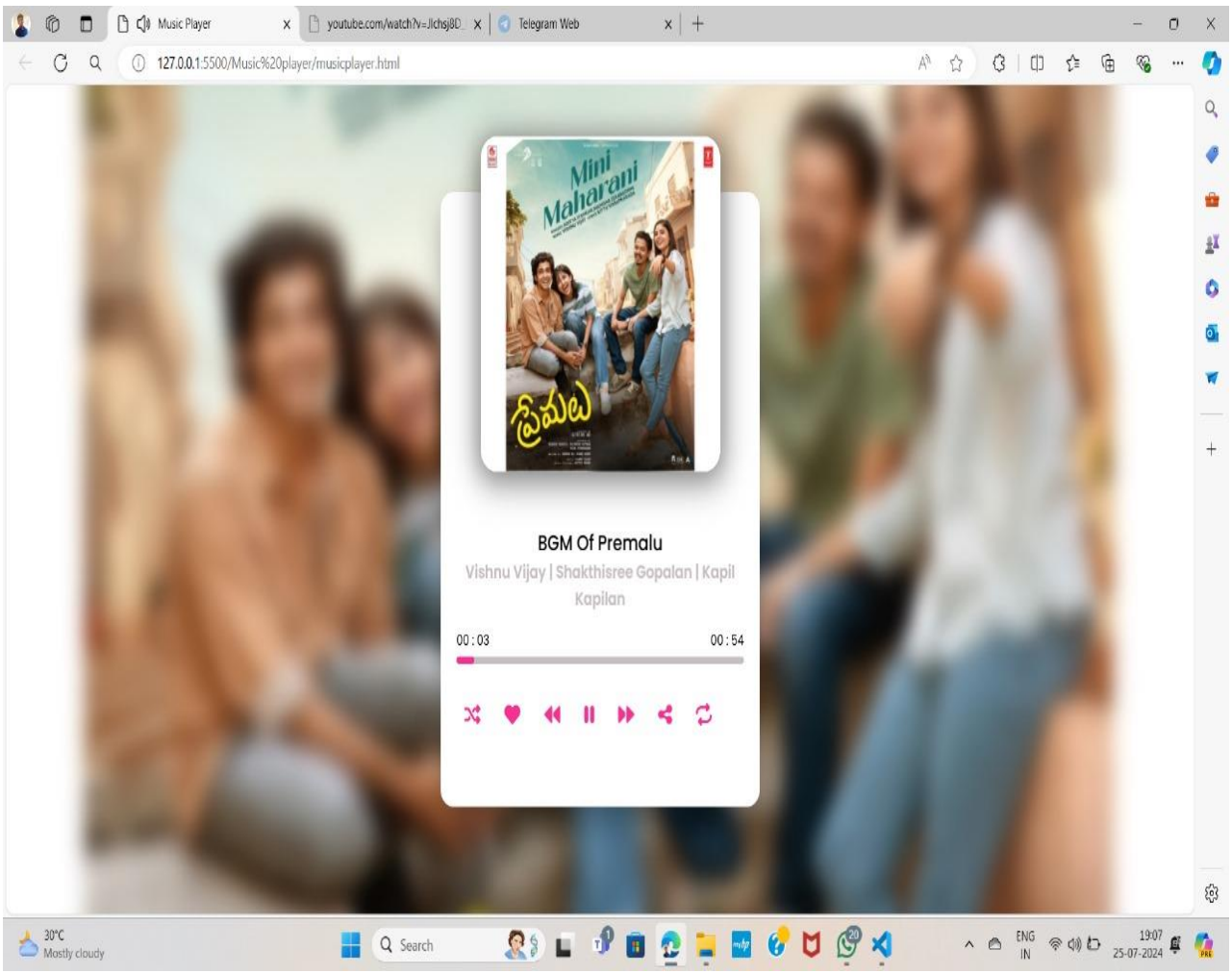
```
"use strict";
const imgEl = document.getElementById("bg_img");
const imgCoverEl = document.getElementById("cover");
const musicTitleEl = document.getElementById("music_title");
const musicArtistEl = document.getElementById("musric_artist");
const playerProgressEl = document.getElementById("player_progress");
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");
```

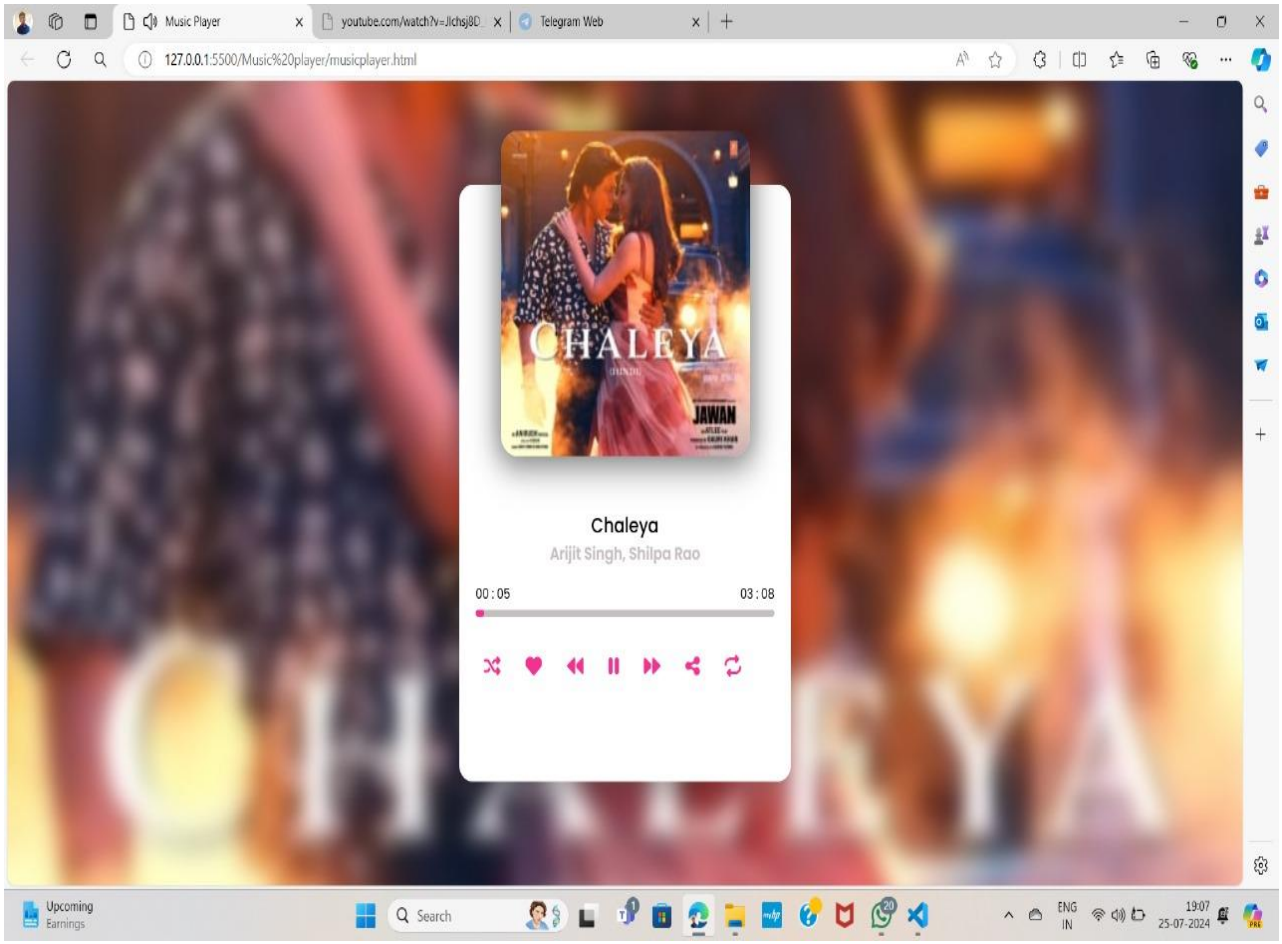
```
const repeatBtnEl = document.getElementById("rep");
const heartBtnEl = document.getElementById("heart");
const shareBtnEl = document.getElementById("shar");

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();
```

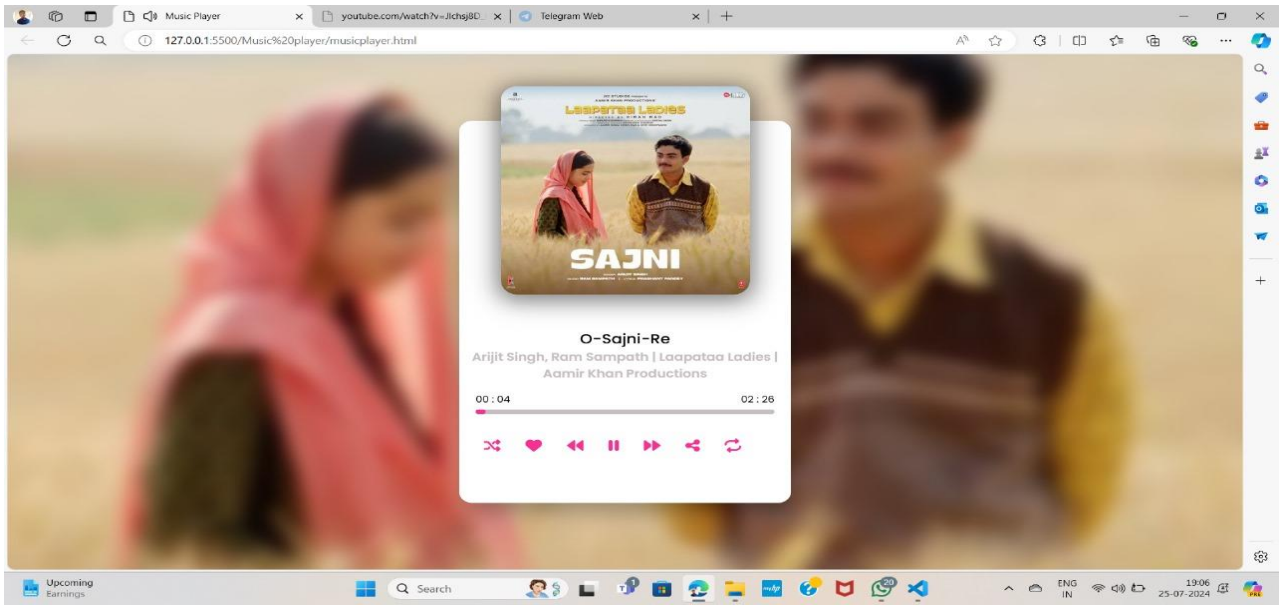
```
  }
}
//================== Play Music====================
function playMusic() {
  isPlaying = true;
  playvBtnEl.classList.replace("fa-play", "fa-pause");
  playvBtnEl.setAttribute("title", "pause");
  music.play();
}
//================== Pause Music====================
function pauseMusic() {
  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=JIchsj8D...
Telegram Web
127.0.0.1:5500/Music%20player/musicplayer.html
Mini Maharani
ప్రేమలు
BGM Of Premalu
Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan
00 : 03
00 : 54
30°C
Mostly cloudy
Search
ENG
IN
19:07
25-07-2024


Music Player
youtube.com/watch?v=JIchsj8D...
Telegram Web
127.0.0.1:5500/Music%20player/musicplayer.html
CHALEYA
JAWAN
Chaleya
Arijit Singh, Shilpa Rao
00 : 05
03 : 08
Upcoming
Earnings
Search
ENG
IN
19:07
25-07-2024


Music Player
youtube.com/watch?v=JIchsj8D...
Telegram Web
127.0.0.1:5500/Music%20player/musicplayer.html
LAPATAA LADIES
SAJNI
O-Sajni-Re
Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions
00 : 04
02 : 26
Upcoming
Earnings
Search
ENG
IN
19:06
25-07-2024

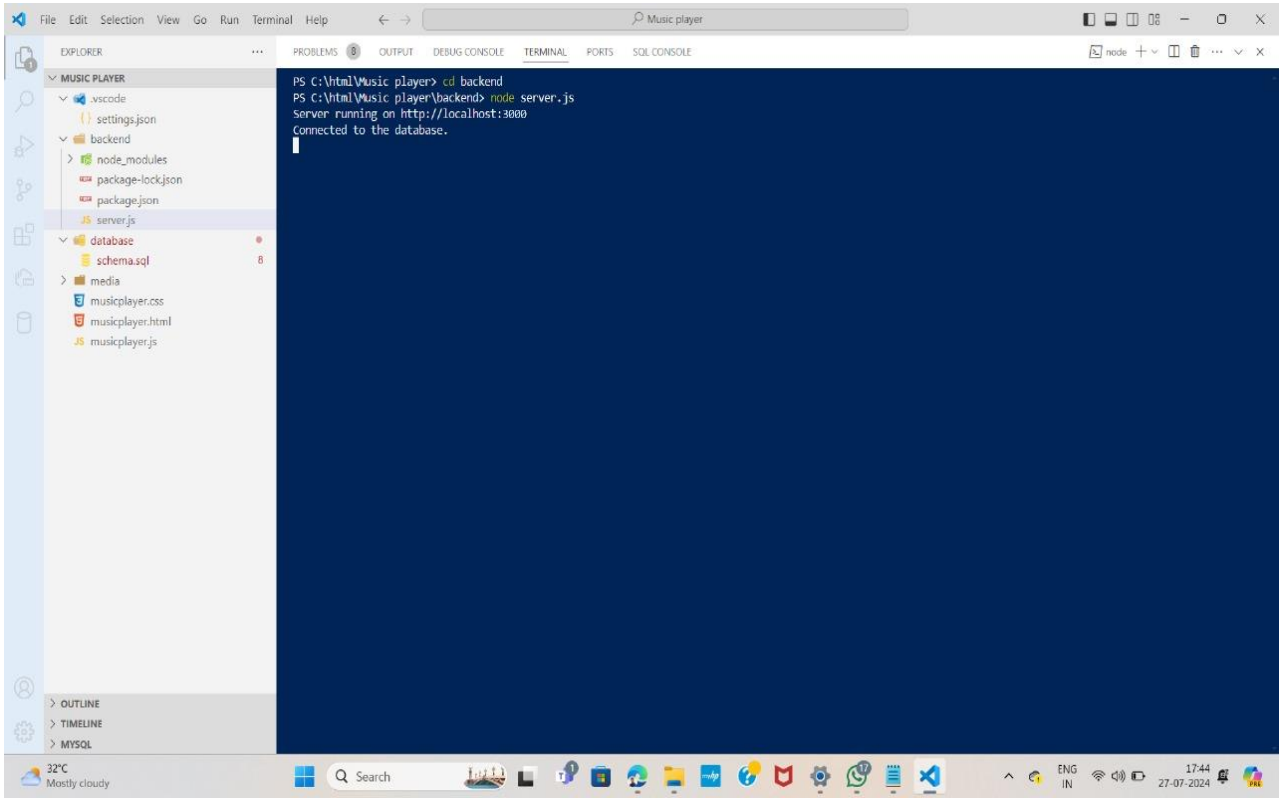
BACKEND
Code:

```
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:



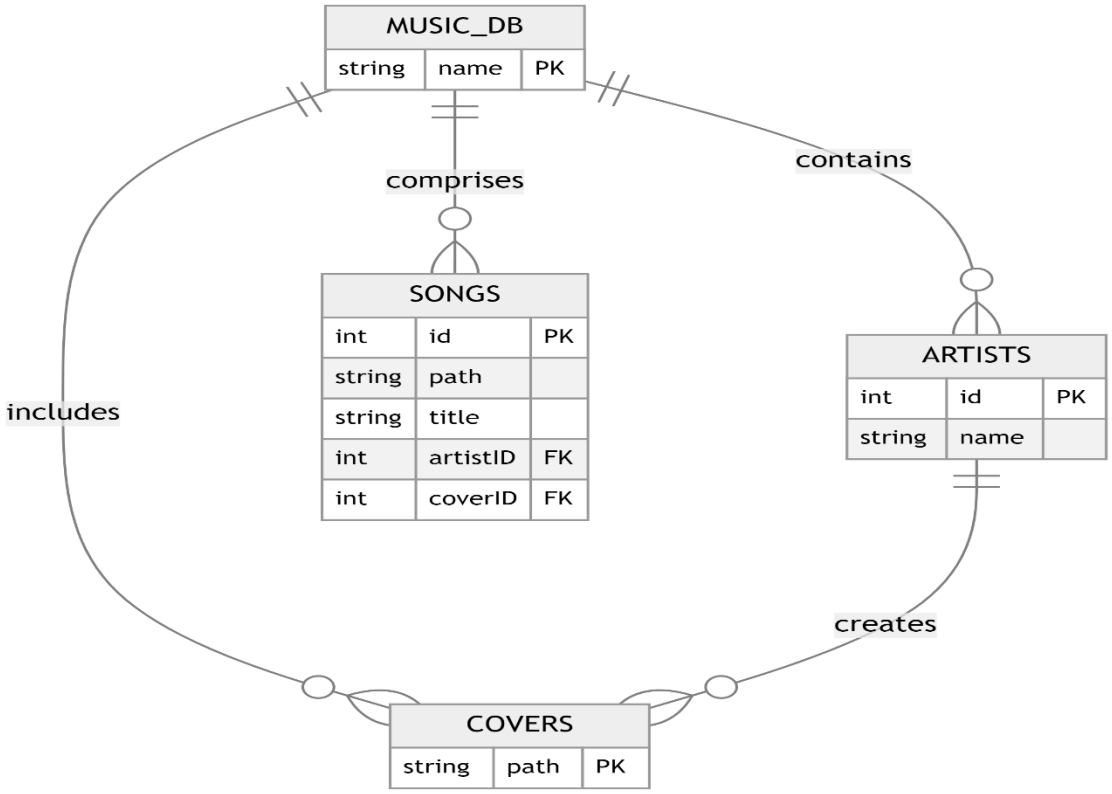
Music Player | localhost:3000/api/songs

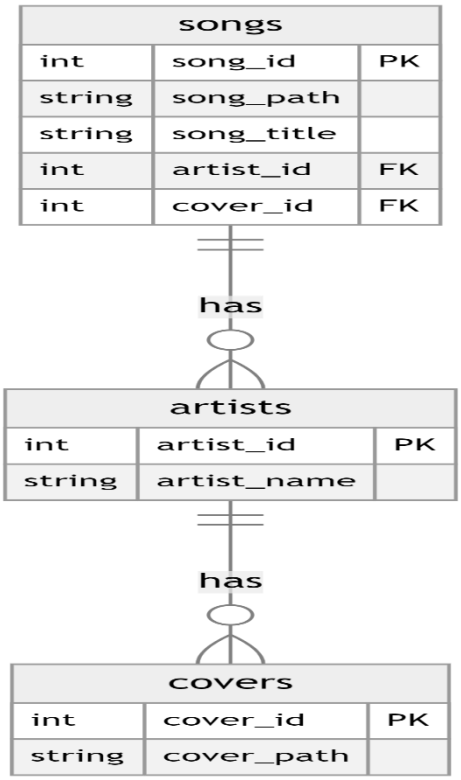
localhost:3000/api/songs

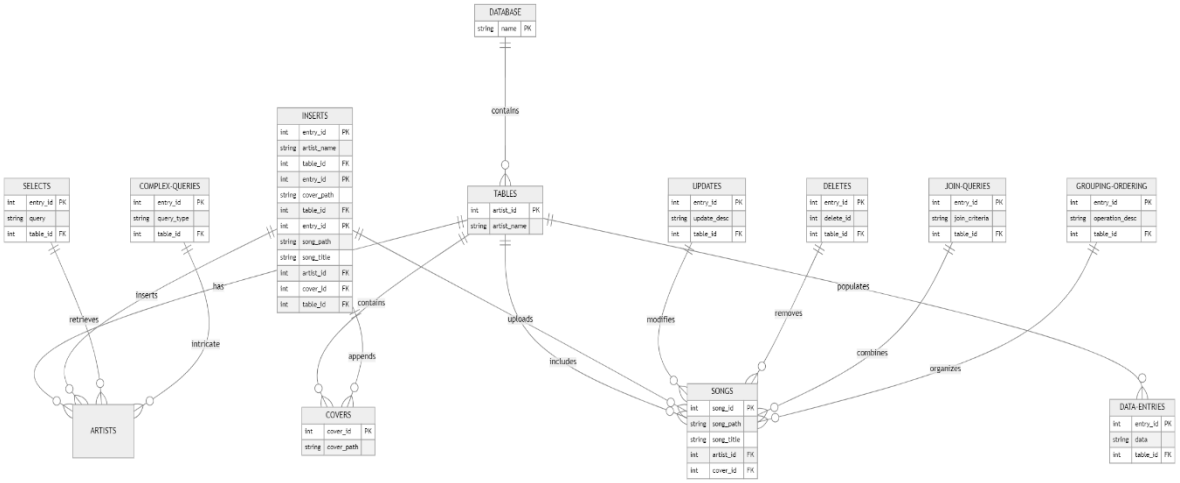

```
[
    {
        "id": 1,
        "path": "media/song1.mp3",
        "displayName": "BGM Of Premalu",
        "cover": "media/image-1.jpg",
        "artist": "Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan"
    },
    {
        "id": 2,
        "path": "media/song2.mp3",
        "displayName": "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"
    }
]
```

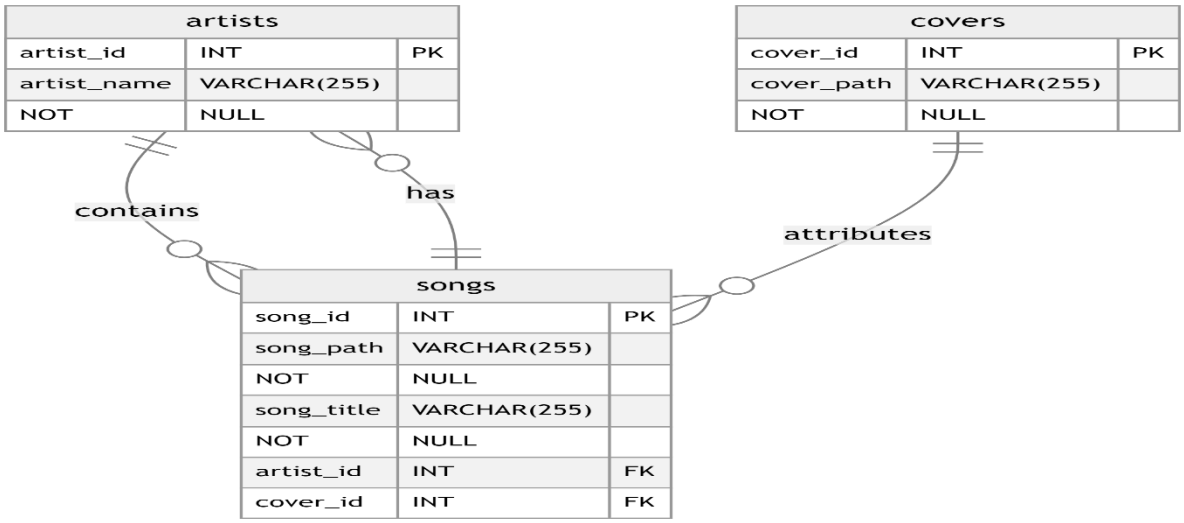

32°C Mostly cloudy | Search | ENG IN | 17:44 27-07-2024

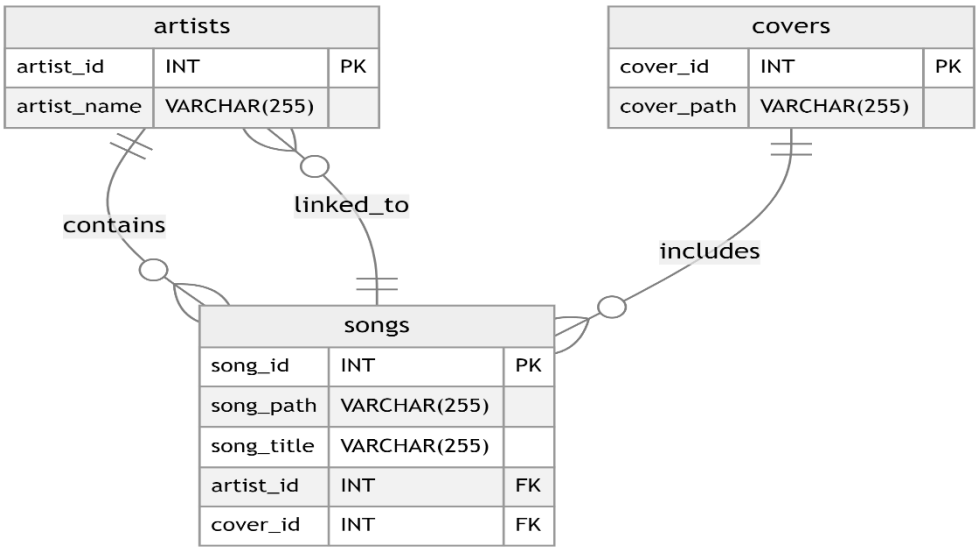
ER DIAGRAMS:


MUSIC_DB
string name PK
comprises
contains
includes
SONGS
int id PK
string path
string title
int artistID FK
int coverID FK
ARTISTS
int id PK
string name
creates
COVERS
string path PK


songs
int song_id PK
string song_path
string song_title
int artist_id FK
int cover_id FK
has
artists
int artist_id PK
string artist_name
has
covers
int cover_id PK
string cover_path


DATABASE
contains
TABLES
SELECTS
COMPLEX-QUERIES
INSERTS
UPDATES
DELETES
JOIN-QUERIES
GROUPING-ORDERING
ARTISTS
COVERS
SONGS
DATA-ENTRIES
inserts
has
retrieves
intricate
appends
contains
uploads
includes
modifies
removes
populates
combines
organizes


artists
artist_id INT PK
artist_name VARCHAR(255)
NOT NULL
covers
cover_id INT PK
cover_path VARCHAR(255)
NOT NULL
contains
has
attributes
songs
song_id INT PK
song_path VARCHAR(255)
NOT NULL
song_title VARCHAR(255)
NOT NULL
artist_id INT FK
cover_id INT FK


artists
artist_id INT PK
artist_name VARCHAR(255)
covers
cover_id INT PK
cover_path VARCHAR(255)
contains
linked_to
includes
songs
song_id INT PK
song_path VARCHAR(255)
song_title VARCHAR(255)
artist_id INT FK
cover_id INT FK

DATABASE

CODE:

```
-- Step 1: Create and use the database
CREATE DATABASE IF NOT EXISTS music_db;
USE music_db;

-- Step 2: Create Tables
CREATE TABLE artists (
    artist_id INT AUTO_INCREMENT PRIMARY KEY,
    artist_name VARCHAR(255) NOT NULL
);

CREATE TABLE covers (
    cover_id INT AUTO_INCREMENT PRIMARY KEY,
    cover_path VARCHAR(255) NOT NULL
);

CREATE TABLE songs (
    song_id INT AUTO_INCREMENT PRIMARY KEY,
    song_path VARCHAR(255) NOT NULL,
    song_title VARCHAR(255) NOT NULL,
    artist_id INT NOT NULL,
    cover_id INT NOT NULL,
    FOREIGN KEY (artist_id) REFERENCES artists (artist_id),
    FOREIGN KEY (cover_id) REFERENCES covers (cover_id)
);

-- Step 3: Inserting data into artists
INSERT INTO artists (artist_name) VALUES
('Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan'),
('VishwakSen, Neha Shetty | Yuvan Shankar Raja'),
('Arijit Singh, Shilpa Rao'),
('Akshath'),
('Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions');

-- Step 4: Inserting data into covers
INSERT INTO covers (cover_path) VALUES
('media/image-1.jpg'),
('media/image-2.jpg'),
('media/image-3.jpg'),
('media/image-4.jpg'),
('media/image-5.jpg');
```

```
-- Step 5: Inserting data into songs
INSERT INTO songs (song_path, song_title, artist_id, cover_id) VALUES
('media/song1.mp3', 'BGM Of Premalu', 1, 1),
('media/song2.mp3', 'Suttamla Soosi', 2, 2),
('media/song3.mp3', 'Chaleya', 3, 3),
('media/song4.mp3', 'Nadaniya', 4, 4),
('media/song5.mp3', 'O-Sajni-Re', 5, 5);

-- Step 6: Data Updating
-- Update a song's title
UPDATE songs
SET song_title = 'BGM Of Love'
WHERE song_id = 1;

-- Step 7: Data Deletion
-- Delete a song
DELETE FROM songs
WHERE song_id = 5;

-- Step 8: Simple Select Queries

-- Retrieve all artists
SELECT * FROM artists;

-- Retrieve all covers
SELECT * FROM covers;

-- Retrieve all songs
SELECT * FROM songs;

-- Step 9: Join Queries

-- Retrieve song details along with artist names and cover paths
SELECT s.song_id, s.song_title, a.artist_name, s.song_path, c.cover_path
FROM songs s
JOIN artists a ON s.artist_id = a.artist_id
JOIN covers c ON s.cover_id = c.cover_id;

-- Step 10: Grouping and Ordering

-- Count the number of songs per artist
SELECT a.artist_name, COUNT(s.song_id) AS song_count
FROM artists a
```

```
JOIN songs s ON a.artist_id = s.artist_id
GROUP BY a.artist_name
ORDER BY song_count DESC;

-- List all songs ordered by title
SELECT song_title, song_path
FROM songs
ORDER BY song_title;

-- Step 11: Complex Queries

-- Subquery: Retrieve the artist with the most songs
SELECT artist_name
FROM artists
WHERE artist_id = (
    SELECT artist_id
    FROM songs
    GROUP BY artist_id
    ORDER BY COUNT(*) DESC
    LIMIT 1
);

-- Using a CTE: Retrieve the top 3 artists by song count
WITH artist_song_count AS (
    SELECT artist_id, COUNT(*) AS song_count
    FROM songs
    GROUP BY artist_id
)
SELECT a.artist_name, ascnt.song_count
FROM artist_song_count ascnt
JOIN artists a ON ascnt.artist_id = a.artist_id
ORDER BY ascnt.song_count DESC
LIMIT 3;
```

OUTPUT:

```
mysql> -- Step 1: Create Database
mysql> CREATE DATABASE IF NOT EXISTS music_db;
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> USE music_db;
Database changed
mysql>
mysql> -- Step 2: Create Tables
mysql> DROP TABLE IF EXISTS songs;
Query OK, 0 rows affected (0.05 sec)

mysql> DROP TABLE IF EXISTS covers;
Query OK, 0 rows affected (0.01 sec)

mysql> DROP TABLE IF EXISTS artists;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE artists (
    ->     artist_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     artist_name VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE covers (
    ->     cover_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     cover_path VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE songs (
    ->     song_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     song_path VARCHAR(255) NOT NULL,
    ->     song_title VARCHAR(255) NOT NULL,
    ->     artist_id INT NOT NULL,
    ->     cover_id INT NOT NULL,
    ->     FOREIGN KEY (artist_id) REFERENCES artists (artist_id),
    ->     FOREIGN KEY (cover_id) REFERENCES covers (cover_id)
```

```
    ->     FOREIGN KEY (cover_id) REFERENCES covers (cover_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> -- Step 3: Inserting data into artists
mysql> INSERT INTO artists (artist_name) VALUES
    -> ('Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan'),
    -> ('VishwakSen, Neha Shetty | Yuvan Shankar Raja'),
    -> ('Arijit Singh, Shilpa Rao'),
    -> ('Akshath'),
    -> ('Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Step 4: Inserting data into covers
mysql> INSERT INTO covers (cover_path) VALUES
    -> ('media/image-1.jpg'),
    -> ('media/image-2.jpg'),
    -> ('media/image-3.jpg'),
    -> ('media/image-4.jpg'),
    -> ('media/image-5.jpg');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Step 5: Inserting data into songs
mysql> INSERT INTO songs (song_path, song_title, artist_id, cover_id) VALUES
    -> ('media/song1.mp3', 'BGM Of Premalu', 1, 1),
    -> ('media/song2.mp3', 'Suttamla Soosi', 2, 2),
    -> ('media/song3.mp3', 'Chaleya', 3, 3),
    -> ('media/song4.mp3', 'Nadaniya', 4, 4),
    -> ('media/song5.mp3', 'O-Sajni-Re', 5, 5);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Step 6: Data Updating
mysql> UPDATE songs
    -> SET song_title = 'BGM Of Love'
```

```
    -> WHERE song_id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> -- Step 7: Data Deletion
mysql> DELETE FROM songs
    -> WHERE song_id = 5;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> -- Step 8: Simple Select Queries
mysql> -- Retrieve all artists
mysql> SELECT * FROM artists;
+-----------+-------------------------------------------------------------------+
| artist_id | artist_name                                                       |
+-----------+-------------------------------------------------------------------+
|         1 | Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan                |
|         2 | VishwakSen, Neha Shetty | Yuvan Shankar Raja                      |
|         3 | Arijit Singh, Shilpa Rao                                          |
|         4 | Akshath                                                           |
|         5 | Arijit Singh, Ram Sampath | Laapataa Ladies | Aamir Khan Productions |
+-----------+-------------------------------------------------------------------+
5 rows in set (0.00 sec)

mysql>
mysql> -- Retrieve all covers
mysql> SELECT * FROM covers;
+----------+-------------------+
| cover_id | cover_path        |
+----------+-------------------+
|        1 | media/image-1.jpg |
|        2 | media/image-2.jpg |
|        3 | media/image-3.jpg |
|        4 | media/image-4.jpg |
|        5 | media/image-5.jpg |
+----------+-------------------+
5 rows in set (0.00 sec)

mysql>
mysql> -- Retrieve all songs
```

```
mysql>
mysql> -- Retrieve all songs
mysql> SELECT * FROM songs;
+---------+----------------+---------------+-----------+----------+
| song_id | song_path      | song_title    | artist_id | cover_id |
+---------+----------------+---------------+-----------+----------+
|       1 | media/song1.mp3 | BGM Of Love   |         1 |        1 |
|       2 | media/song2.mp3 | Suttamla Soosi |         2 |        2 |
|       3 | media/song3.mp3 | Chaleya       |         3 |        3 |
|       4 | media/song4.mp3 | Nadaniya      |         4 |        4 |
+---------+----------------+---------------+-----------+----------+
4 rows in set (0.00 sec)

mysql>
mysql> -- Step 9: Join Queries
mysql> -- Retrieve song details along with artist names and cover paths
mysql> SELECT s.song_id, s.song_title, a.artist_name, s.song_path, c.cover_path
    -> FROM songs s
    -> JOIN artists a ON s.artist_id = a.artist_id
    -> JOIN covers c ON s.cover_id = c.cover_id;
+---------+---------------+-------------------------------------------------+----------------+-------------------+
| song_id | song_title    | artist_name                                     | song_path      | cover_path        |
+---------+---------------+-------------------------------------------------+----------------+-------------------+
|       1 | BGM Of Love   | Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan | media/song1.mp3 | media/image-1.jpg |
|       2 | Suttamla Soosi | VishwakSen, Neha Shetty | Yuvan Shankar Raja | media/song2.mp3 | media/image-2.jpg |
|       3 | Chaleya       | Arijit Singh, Shilpa Rao                        | media/song3.mp3 | media/image-3.jpg |
|       4 | Nadaniya      | Akshath                                         | media/song4.mp3 | media/image-4.jpg |
+---------+---------------+-------------------------------------------------+----------------+-------------------+
4 rows in set (0.00 sec)

mysql>
mysql> -- Step 10: Grouping and Ordering
mysql> -- Count the number of songs per artist
mysql> SELECT a.artist_name, COUNT(s.song_id) AS song_count
    -> FROM artists a
    -> JOIN songs s ON a.artist_id = s.artist_id
    -> GROUP BY a.artist_name
    -> ORDER BY song_count DESC;
+-------------------------------------------------+------------+
| artist_name                                     | song_count |
```

```
    -> ORDER BY song_count DESC;
+-------------------------------------------------+------------+
| artist_name                                     | song_count |
+-------------------------------------------------+------------+
| Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan |          1 |
| VishwakSen, Neha Shetty | Yuvan Shankar Raja    |          1 |
| Arijit Singh, Shilpa Rao                        |          1 |
| Akshath                                         |          1 |
+-------------------------------------------------+------------+
4 rows in set (0.00 sec)

mysql>
mysql> -- List all songs ordered by title
mysql> SELECT song_title, song_path
    -> FROM songs
    -> ORDER BY song_title;
+----------------+-----------------+
| song_title     | song_path       |
+----------------+-----------------+
| BGM Of Love    | media/song1.mp3 |
| Chaleya        | media/song3.mp3 |
| Nadaniya       | media/song4.mp3 |
| Suttamla Soosi | media/song2.mp3 |
+----------------+-----------------+
4 rows in set (0.00 sec)

mysql>
mysql> -- Step 11: Complex Queries
mysql> -- Subquery: Retrieve the artist with the most songs
mysql> SELECT artist_name
    -> FROM artists
    -> WHERE artist_id = (
    ->     SELECT artist_id
    ->     FROM songs
    ->     GROUP BY artist_id
    ->     ORDER BY COUNT(*) DESC
    ->     LIMIT 1
    -> );
+-------------------------------------------------+
| artist_name                                     |
+-------------------------------------------------+
```

```
mysql> -- Step 11: Complex Queries
mysql> -- Subquery: Retrieve the artist with the most songs
mysql> SELECT artist_name
    -> FROM artists
    -> WHERE artist_id = (
    ->     SELECT artist_id
    ->     FROM songs
    ->     GROUP BY artist_id
    ->     ORDER BY COUNT(*) DESC
    ->     LIMIT 1
    -> );
+-------------------------------------------------+
| artist_name                                     |
+-------------------------------------------------+
| Vishnu Vijay | Shakthisree Gopalan | Kapil Kapilan |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql>
mysql> -- Using a CTE: Retrieve the top 3 artists by song count
mysql> WITH artist_song_count AS (
    ->     SELECT artist_id, COUNT(*) AS song_count
    ->     FROM songs
    ->     GROUP BY artist_id
    -> )
    -> SELECT a.artist_name, ascnt.song_count
    -> FROM artist_song_count ascnt
    -> JOIN artists a ON ascnt.artist_id = a.artist_id
    -> ORDER BY ascnt.song_count DESC
    -> LIMIT 3;
```