**DBMS PROJECT**

**T-SERIES MUSIC DATABASE**

**(DOCUMENTATION)**

**BY:**

**M.M.S. SWAROOP,120316426 (PROJECT HEAD)**

**P. HARISH,1210316440**

**P. AJAY,1210316402**

**S. NAGARJUNA,1210316432**

**B4, 3RD YEAR**

**CONTENTS**

* QUESTION
* TECHNOLOGIES USED
* DESCRIPTION ABOUT THE TECHNOLOGIES
* LINKS TO INSTALL REQUIREMENTS
* ER DIAGRAM
* UML DIAGRAM
* SQL CODE FOR CREATION OF TABLES (with Primary keys and Foreign keys)
* SERVER-SIDE CODE (BACKEND)
* CLIENT-SIDE CODE (FRONTEND)
  + - EJS CODE (TO DISPLAY THE CONTENT)
    - EXPRESSJS CODE (ACTS AS API)
    - CSS CODE (TO ADD STYLES TO THE CONTENT)
    - BASIC JAVASCRIPT WITH JQUERY (TO ADD ANIMATIONS)
* AKNOWLEDGEMENT
* SOURCES

**T-SERIES RECORDING COMPANY**

T-series recording company has decided to store information about musicians who performs an its album (as well as other company date) in a database. (Assume if any data is required)

1. Each musician that records at t-series has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.
2. Each instrument used in songs recorded at T-series has a unique identification number =, a name (e.g., guitar, synthesizer, flute) and a musical key (E.G., C, B-flat, e-flat).
3. Each album records on the T-series label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
4. Each song recorded at T-series has a title and an author.
5. Each musician may play several instruments, and a given instrument may be played by several musicians.
6. Each album has a number of songs on it, but no song appears on more than one album.
7. Each song is performed by one or more musicians, and a musician may perform a number of songs.
8. Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

**TECHNOLOGIES USED**

1. NODEJS
2. EXPRESSJS
3. EJS (embedded JAVASCRIPT)
4. JQUERY (FOR ANIMATIONS)
5. CSS (FOR STYLING)
6. NODEMON (A NODEJS PACKAGE)
7. MYSQL (FOR DATABASE)
8. MYSQL SERVER 8.0 (TO HOST THE DB)
9. ATOM (EDITOR)

**ABOUT TECHNOLOGIES**

**Nodejs**



Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.

JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser.

Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client-side scripts.

The version which we used for this project is 9.2.0.0

**Expressjs**



Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

Web Application Framework:-

A web framework (WF) or web application framework (WAF) is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs. Web frameworks provide a standard way to build and deploy web applications. Web frameworks aim to automate the overhead associated with common activities performed in web development

**EJS**

EMBEDDED JAVASCRIPT

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript.

**Features**

Fast compilation and rendering

Simple template tags: <% %>

Custom delimiters (e.g., use <? ?> instead of <% %>)

**Includes**

1) Both server JS and browser support

2) Static caching of intermediate JavaScript

3) Static caching of templates

4) Complies with the Express view system

**Tags**

* <% 'Scriptlet' tag, for control-flow, no output
* <%\_ 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
* <%= Outputs the value into the template (escaped)
* <%- Outputs the unescaped value into the template
* <%# Comment tag, no execution, no output
* <%% Outputs a literal '<%'
* %%> Outputs a literal '%>'
* %> Plain ending tag
* -%> Trim-mode ('newline slurp') tag, trims following newline
* \_%> 'Whitespace Slurping' ending tag, removes all whitespace after it

**MYSQL**

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications.

MySQL is based on a client-server model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). MySQL server is available as a separate program for use in a client-server networked environment and as a library that can be embedded (or linked) into separate applications.

In our project, we have used MySQL as our database

**LINKS TO INSTALL REQUIREMENTS**

MYSQL –

<https://dev.mysql.com/downloads/windows/installer/8.0.html>

NODEJS –

<https://nodejs.org/en/download>

EXPRESSJS –

Use command – npm install express –save

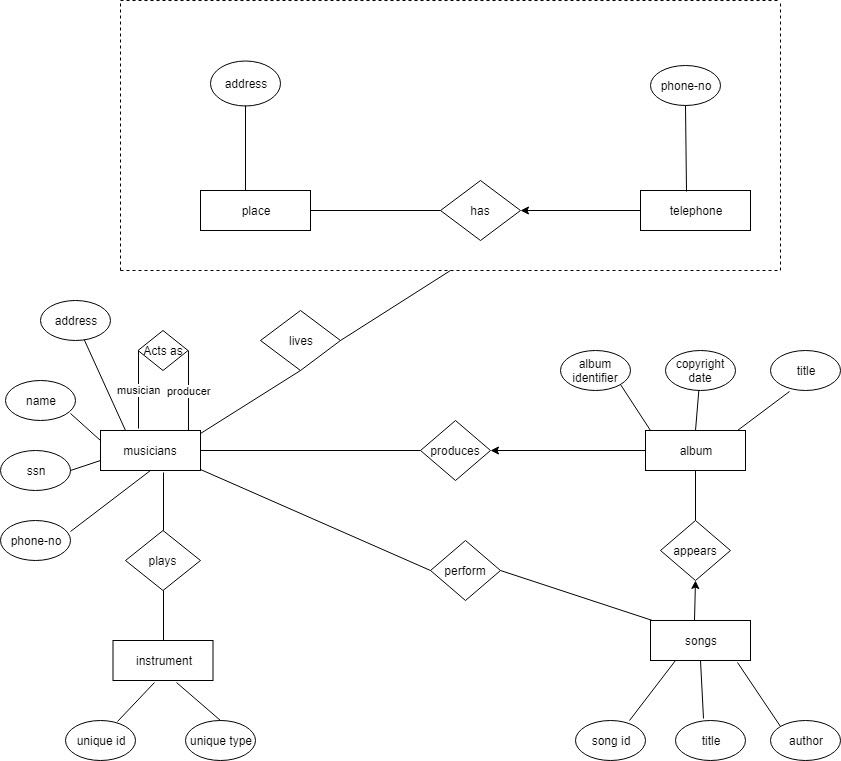
EJS –

Use command – npm install ejs –save

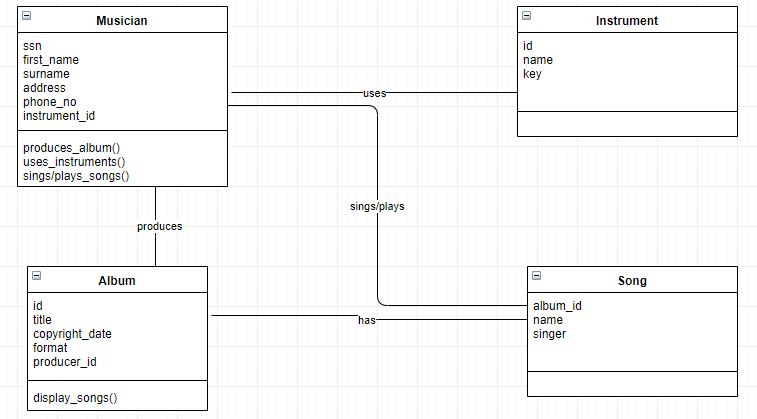
NODEMON –

Use command – npm install -g nodemon

**ER DIAGRAM**

****

**UML DIAGRAM**

****

**CODE TO CREATE TABLES IN DATABASE**

create table musician

(ssn integer primary key,

first\_name char(20),

surname char(20),

address varchar(100) not null,

phone\_no char(15) not null,

id\_i integer,

foreign key(id\_i) references instrument(id\_i)

);

create table instrument

(id\_i integer primary key,

name\_mi char(20),

m\_key varchar(10)

);

create table album

(id\_a integer primary key,

title char(20),

cp\_date date,

format1 char(20),

producer\_id integer,

foreign key(producer\_id) references musician(ssn)

);

create table song

(id\_a integer,

foreign key(id\_a) references album(id\_a),

title char(20),

author char(20)

);

**CODE IN JAVASCRIPT TO CONNECT THE DATABASE TO THE FRONT-END AND ALSO TO IMPORT SOME PACKAGES**

var mysql = require('mysql');

var express = require('express');

var app = express();

var bodyParser = require('body-parser');

const path = require('path');

var connection = mysql.createConnection({

host : '127.0.0.1',

user : 'root',

password : '\*\*\*\*',

database: "Tseries"

});

app.set('view engine', 'ejs');

app.use(bodyParser.json() ); // to support JSON-encoded bodies

app.use(bodyParser.urlencoded({extended: true })); // to support URL-encoded bodies

app.use(express.static(\_\_dirname + '/login'));

connection.connect(function(err) {

if (err) {

console.error('error connecting: ' + err.stack);

return;

}

console.log('connected as id ' + connection.threadId);

});

app.listen(8099); //EXPREESS APPLICATION HOSTED AT PORT 8099

**EJS CODE (TO DISPLAY THE CONTENT AND TO TAKE INPUTS)**

<form class="login1001-form validate-form" method="POST" action="/find\_m">

<span class="login1001-form-title p-b-20 p-t-20">

MUSICIANS

</span>

<div class="wrap-input1001 validate-input" data-validate = "Enter Firstname">

<input class="input1001" type="text" name="Firstname" placeholder="Firstname">

<span class="focus-input100" ></span>

</div>

<div class="wrap-input1001 validate-input move" data-validate="Enter Surname">

<input class="input1001" type="text" name="Surname" placeholder="Surname">

<span class="focus-input100"></span>

</div>

<button class="login1001-form-btn">

<i class="fa fa-search f "></i>

</button>

</form>

<div class="result r p-b-10 ">

<span class="result p-b-10 m-b-10"><%=errorm%></span>

<p class="result p-b-10 ">Name : <%=fname%> <%=surname%></p>

<p class="result p-b-10 p-t-10">Address : <%=address%></p>

<p class="result p-b-10 p-t-10">Phone no. : <%=pno%></p>

<p class="result p-b-10 p-t-10">Instrument used: <%=inu%></p>

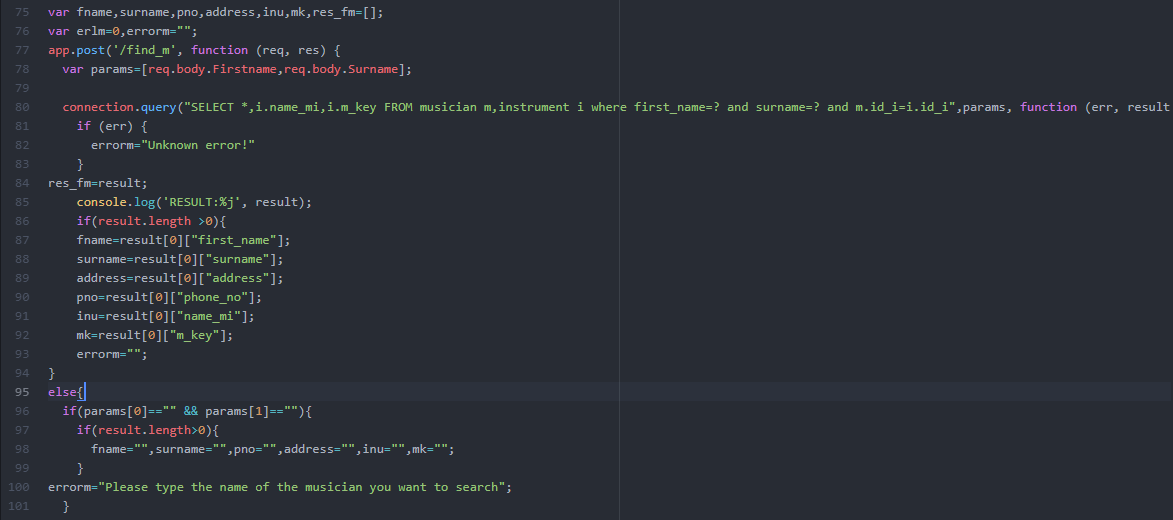
<p class="result p-b-10 p-t-10">Key: <%=mk%></p>

</div>

* THERE ARE MANY OTHER EJS FILES SIMILAR TO THE ABOVE FILE TO DISPLAY THE CONTENT



**EXPRESSJS CODE**



var fname,surname,pno,address,inu,mk,res\_fm=[];

var erlm=0,errorm="";

app.post('/find\_m', function (req, res) {

var params=[req.body.Firstname,req.body.Surname];

connection.query("SELECT \*,i.name\_mi,i.m\_key FROM musician m,instrument i where first\_name=? and surname=? and m.id\_i=i.id\_i",params, function (err, result) {

if (err) {

errorm="Unknown error!"

}

res\_fm=result;

console.log('RESULT:%j', result);

if(result.length >0){

fname=result[0]["first\_name"];

surname=result[0]["surname"];

address=result[0]["address"];

pno=result[0]["phone\_no"];

inu=result[0]["name\_mi"];

mk=result[0]["m\_key"];

errorm="";

}

else{

if(params[0]=="" && params[1]==""){

if(result.length>0){

fname="",surname="",pno="",address="",inu="",mk="";

}

errorm="Please type the name of the musician you want to search";

}

else{

if(result.length>0){

fname="",surname="",pno="",address="",inu="",mk="";

}

errorm="Sorry, no musician named "+params[0]+" "+params[1]+" was found!";

}

}

erlm=errorm.length;

res.render('C:/Users/mmssw/Desktop/dbms\_js/Login/musicians',{ fname:fname,surname:surname,address:address,pno:pno,inu:inu,mk:mk,errorm:errorm});

});

});

app.get('/musicians', function (req, res) {

if(erlm >0){

errorm="";

}

if(res\_fm.length>0){

fname="",surname="",pno="",address="",inu="",mk="";

}

res.render('C:/Users/mmssw/Desktop/dbms\_js/Login/musicians',{ fname:fname,surname:surname,address:address,pno:pno,inu:inu,mk:mk,errorm:errorm});

});

**CSS CODE**

.box1, .box2, .box3 , .box4{

float: right;

width: 20px;

height:20px;

font-size:25px;

color:white;

margin-right: 50px;

border-radius:20%;

transition: margin-right 0.5s;

}

table {

font-family: Poppins-Regular, sans-serif;

border-collapse: collapse;

width: 75%;

margin: auto;

margin-top: 20px;

}

td, th {

border: 1px solid #ffffff;

text-align: left;

padding: 8px;

color: #ffffff;

}

.act{

display: flex;

flex-wrap: wrap;

justify-content: center;

align-items: center;

}

.footer {

font-family: Poppins-Regular, sans-serif;

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: rgba(179, 22, 22, 0.5);

color: white;

}

.ud{

position: absolute;

right: 0;

top:0;

z-index: 11;

}

* THIS IS JUST A SMALL SNIPPET OF THE LARGE CSS FILE

**BASIC JAVASCRIPT CODE (WITH JQUERY)**

$(".slide1").hover(

function(){

$(".box1").css("margin-right",20);

},

function(){

$(".box1").css("margin-right",50);

});

$(".slide2").hover(

function(){

$(".box2").css("margin-right",20);

},

function(){

$(".box2").css("margin-right",50);

});

$(".slide3").hover(

function(){

$(".box3").css("margin-right",20);

},

function(){

$(".box3").css("margin-right",50);

});

$(".slide4").hover(

function(){

$(".box4").css("margin-right",20);

},

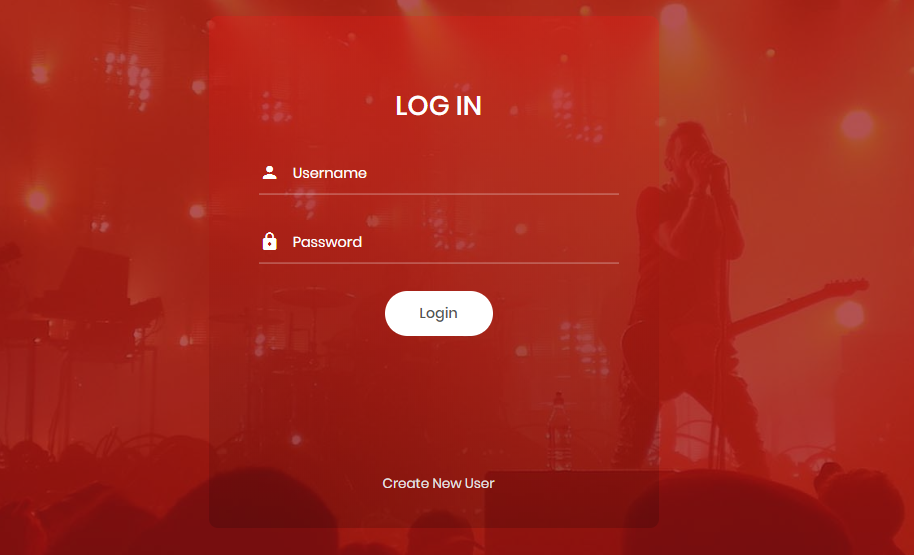
function(){

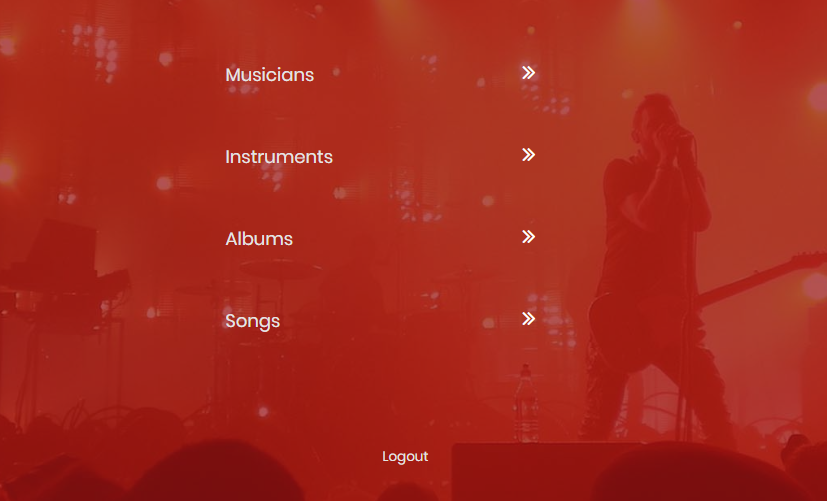
$(".box4").css("margin-right",50);

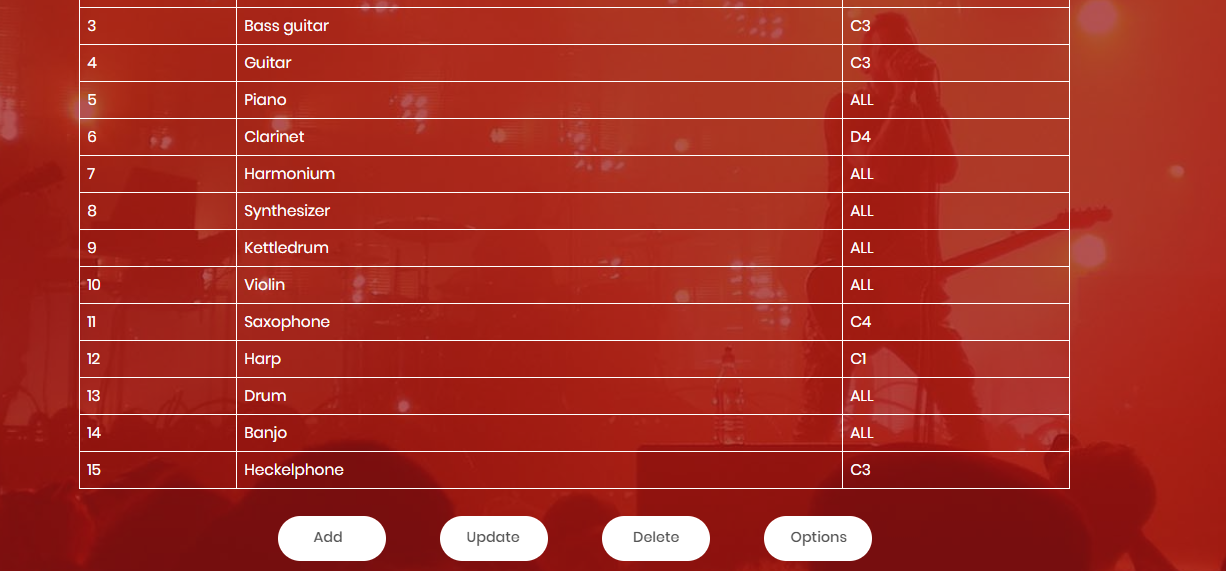
});

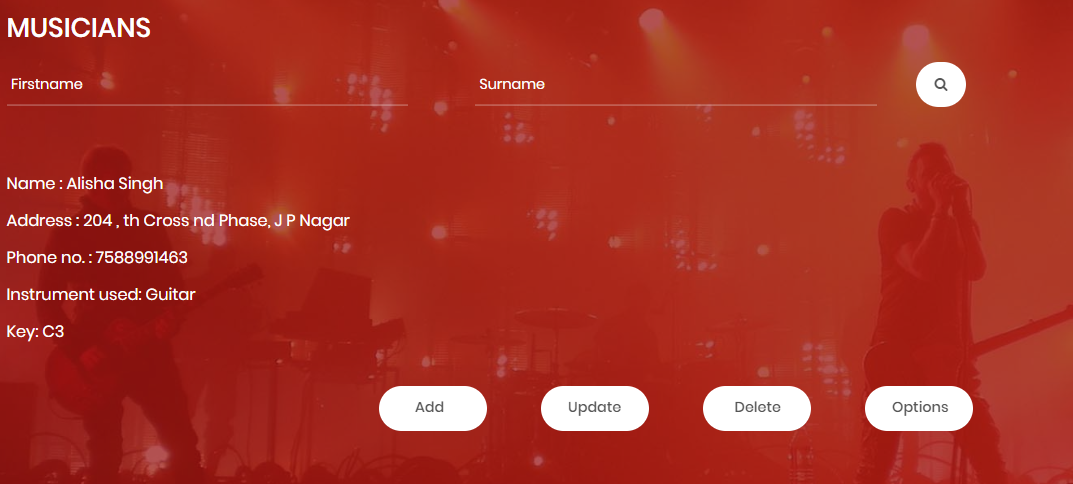
* THESE ARE FEW DOM ELEMENTS WHICH ARE ACCESED FOR ANIMATIONS.

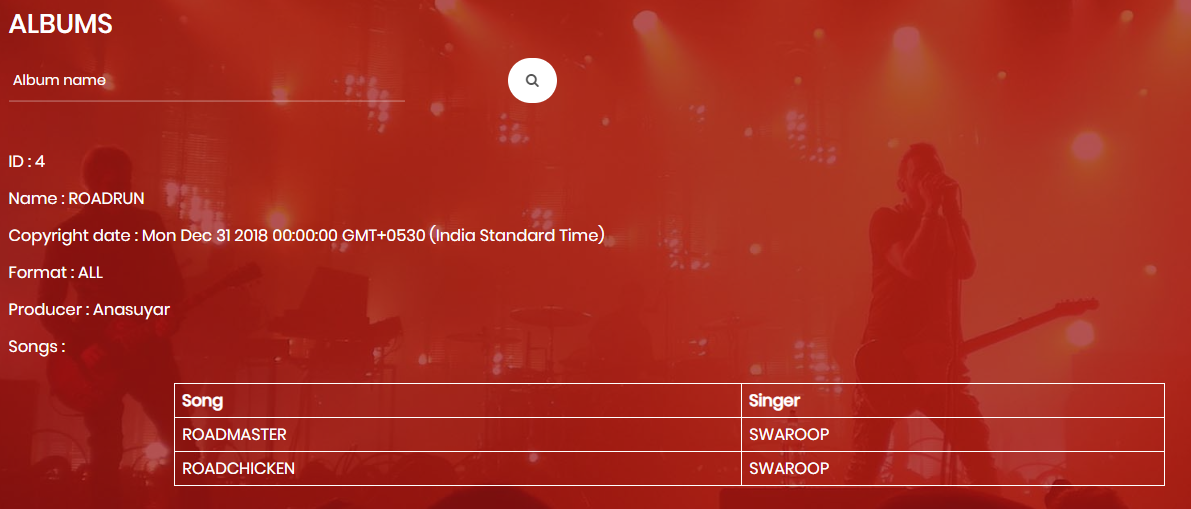
**OUTPUT**











**AKNOWLEDGEMENT**

We would like to thank our respected teachers, Professor K. Thammi Reddy and Ms. Himaja for giving us an opportunity to gain knowledge and get to know about the present industrial needs through this project. We, as a team have learnt a lot which are not generally covered in regular university curriculum through this project. This project will definitely effect our industrial journey in the field of Computer Science in a good way.

**Sources** : Google, Stackoverflow, Expressjs documentation, ejs documentation, Jquery documentation, tutorialspoint.com, draw.io, Medium articles, webapplog.com