ITCS212 Web Programming

Project Report All Phase

Presented to

Aj. Wudhichart Sawangphol

Aj. Jidapa Kraisangka

By

6388010 Teerapat Burasotikul Section 1

6388021 Thanawat Kanjanapoo Section 1

6388047 Phawat Rittarkananone Section 1

6388128 Thanakorn Charoenritthitham Section 1

Faculty of Information and Communication Technology

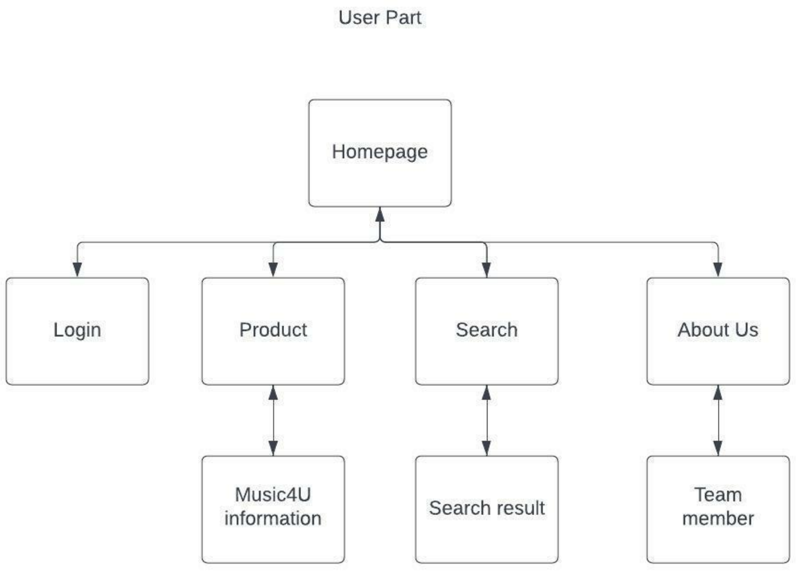
Mahidol University, Semester 2/2022

Overview

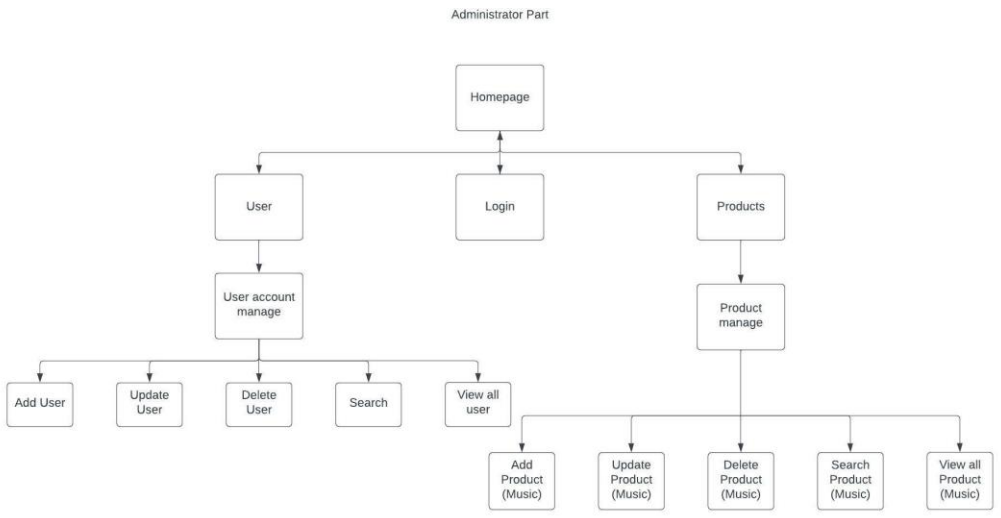
Music4U is the name of our website. Our website was created to be a database website for a music which user can search the song and artist to show the result. The result that will show is music name, artist, available streaming platform, and music video. The data can be adding, update, or delete only by admin. On the other hand, users can only search the database for music information.

Navigation Diagram

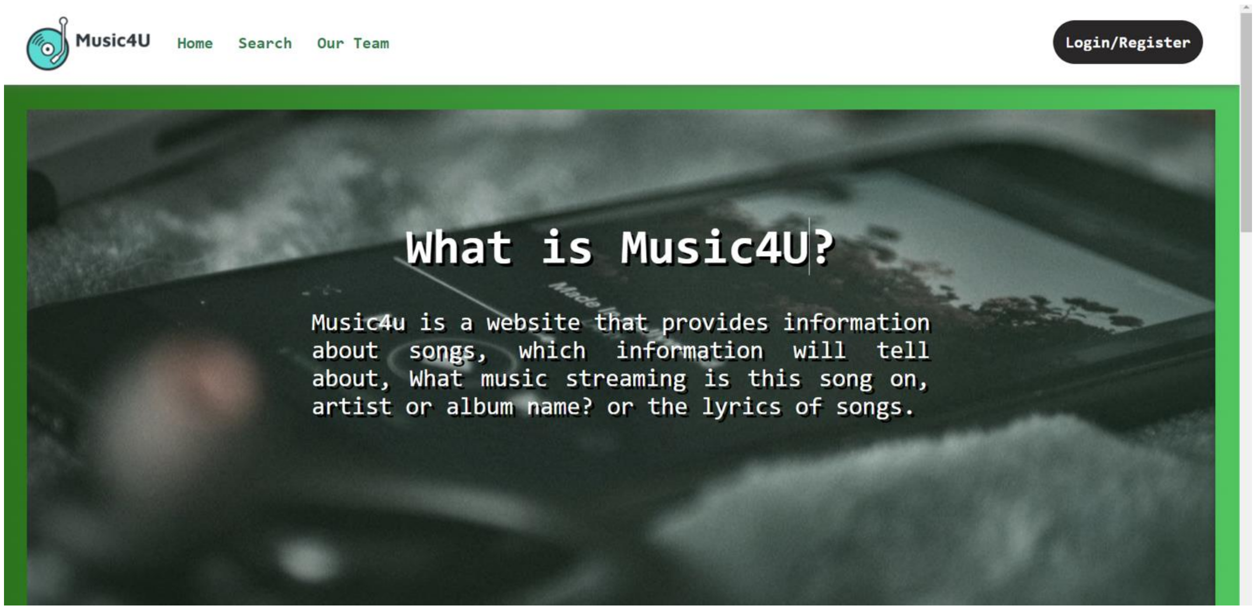
User diagram



Administrator diagram



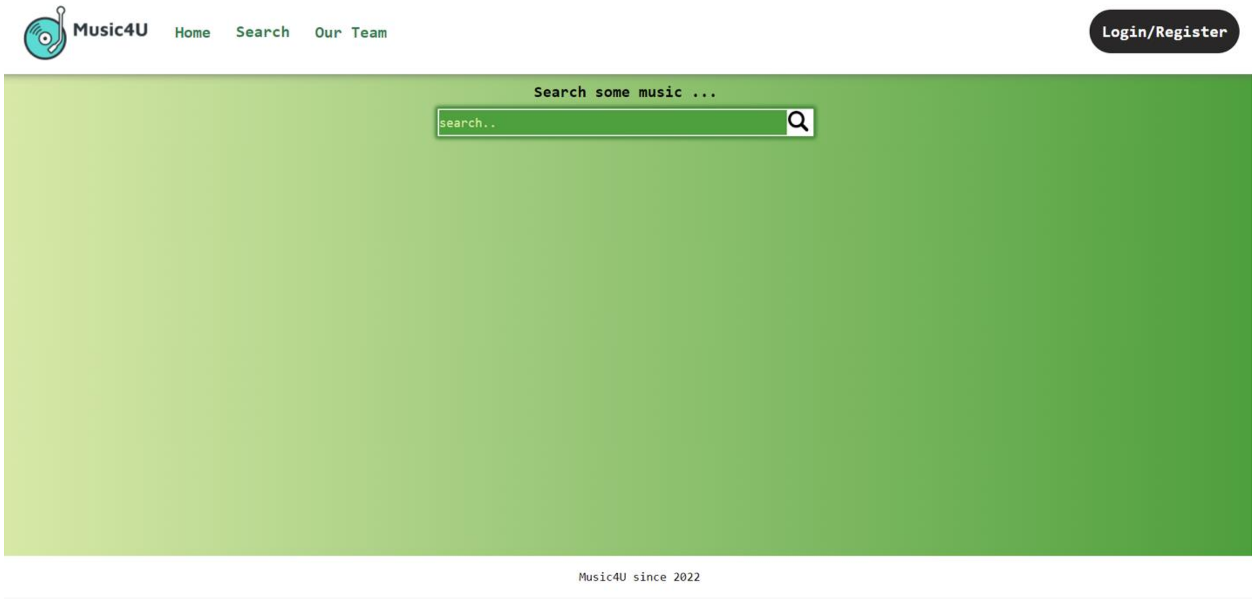
Normal user pages with explanation



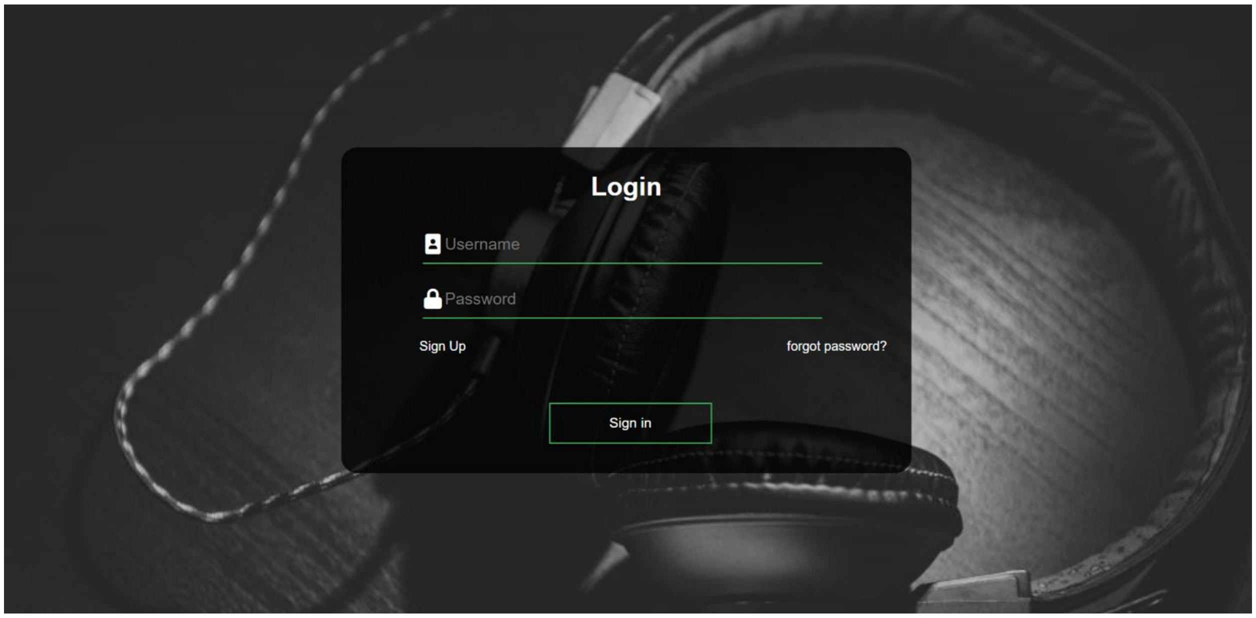
What is Music4U?

Music4u is a website that provides information about songs, which information will tell about, What music streaming is this song on, artist or album name? or the lyrics of songs.

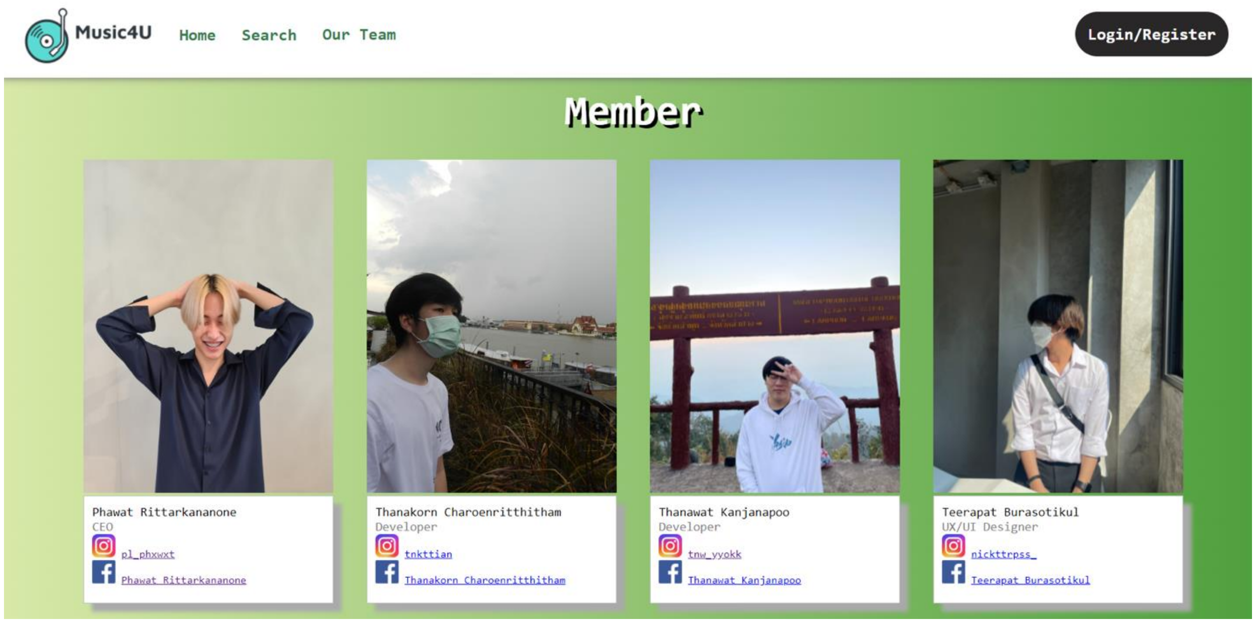
Homepage: This is the homepage of our website that provides information about our website and what our website can do.



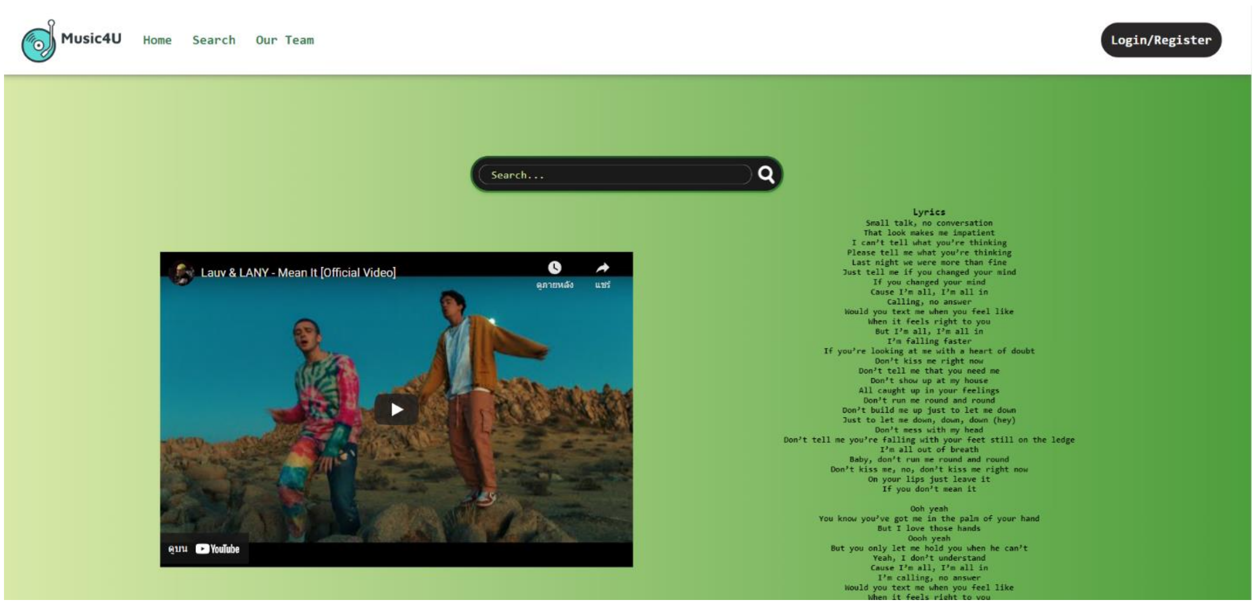
Search page: This is the search page of our website where users can search music information from the database.



Login page: This is the login page of our website that users can login if users already have an account.



About us page: This is an about us page of our website, this page will show the information of each member such as Instagram or Facebook.

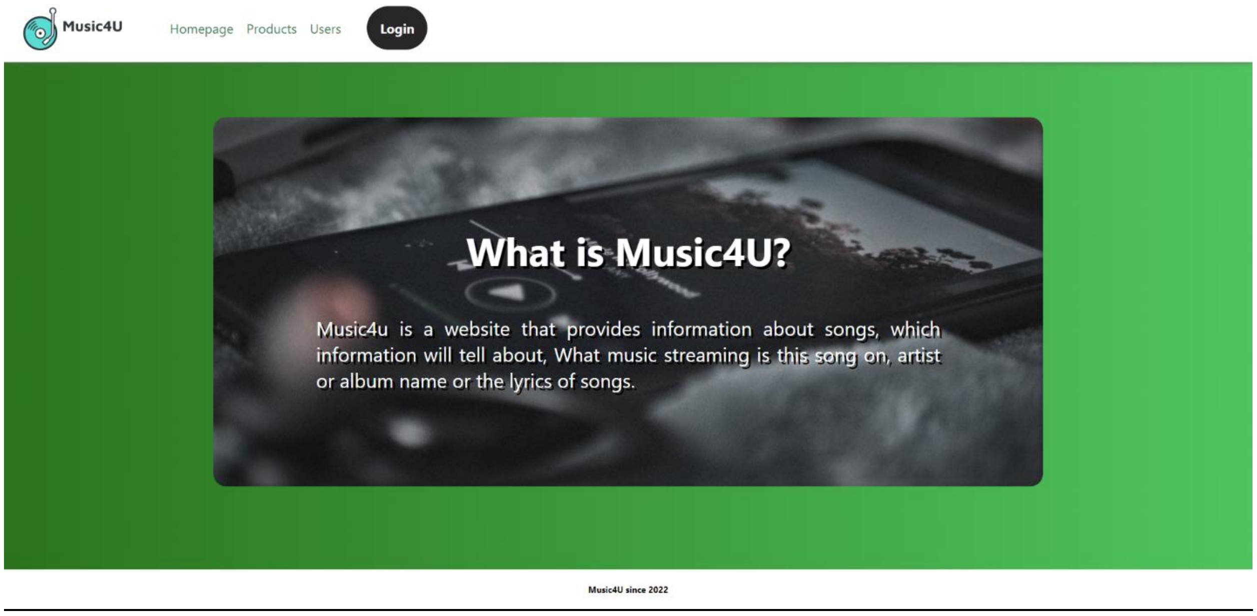


Lyrics
Small talk, no conversation
That look makes me impatient
I can’t tell what you’re thinking
Please tell me what you’re thinking
Last night we were more than fine
Just tell me if you changed your mind
If you changed your mind
Cause I’m all, I’m all in
Calling, no answer
Would you text me when you feel like
When it feels right to you
But I’m all, I’m all in
I’m falling faster
If you’re looking at me with a heart of doubt
Don’t kiss me right now
Don’t tell me that you need me
Don’t show up at my house
All caught up in your feelings
Don’t run me round and round
Don’t build me up just to let me down
Just to let me down, down, down (hey)
Don’t mess with my head
Don’t tell me you’re falling with your feet still on the ledge
I’m all out of breath
Baby, don’t run me round and round
Don’t kiss me, no, don’t kiss me right now
On your lips just leave it
If you don’t mean it

Ooh yeah
You know you’ve got me in the palm of your hand
But I love those hands
Oooh yeah
But you only let me hold you when he can’t
Yeah, I don’t understand
Cause I’m all, I’m all in
I’m calling, no answer
Would you text me when you feel like
When it feels right to you

Result page: This is the result page after users search for music on our website.

Administrators pages with explanation

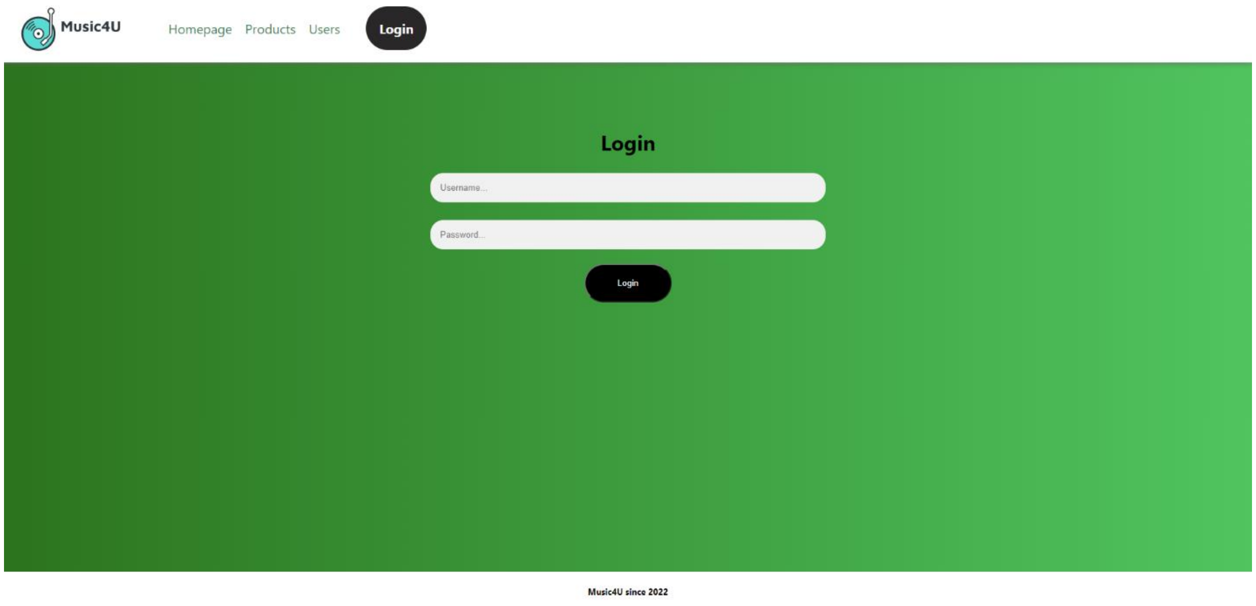


What is Music4U?

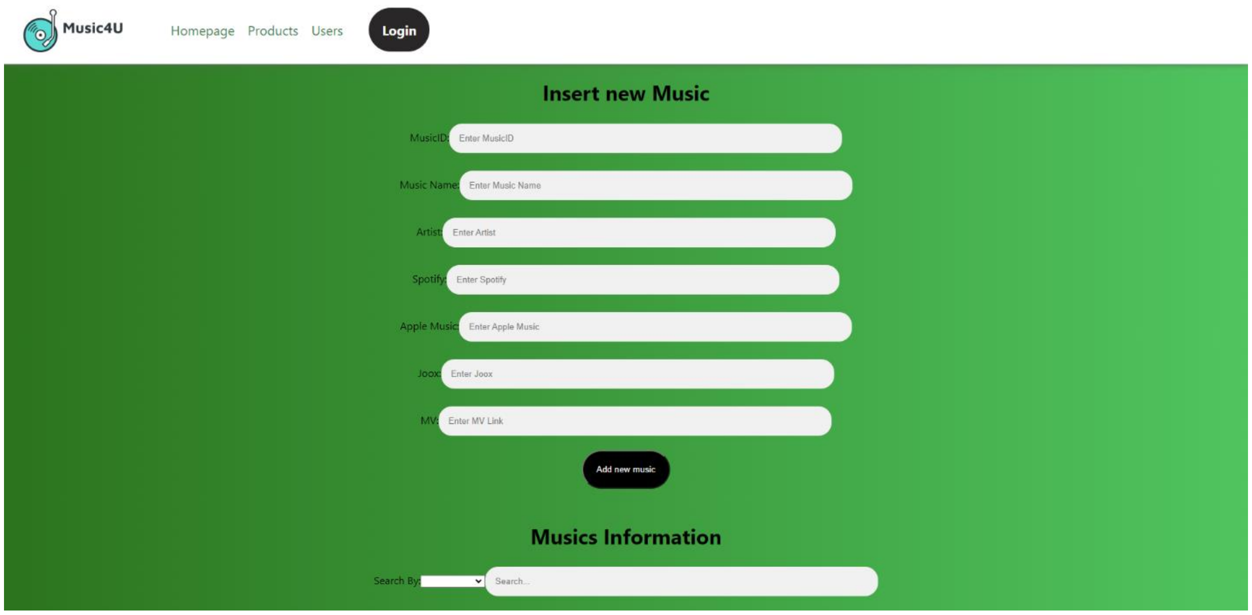
Music4u is a website that provides information about songs, which information will tell about, What music streaming is this song on, artist or album name or the lyrics of songs.

Music4U since 2022

Homepage: This is a homepage for administrators.



Login page: This is a login page for administrators of our website.

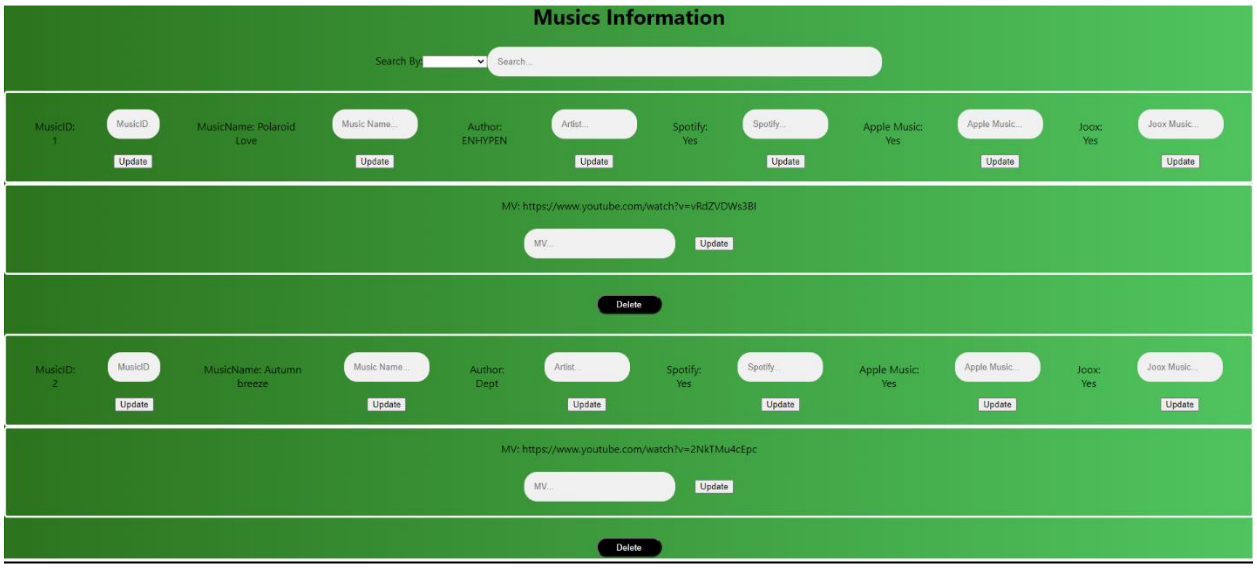


Insert new Music

MusicID:	Enter MusicID
Music Name:	Enter Music Name
Artist:	Enter Artist
Spotify:	Enter Spotify
Apple Music:	Enter Apple Music
Joox:	Enter Joox
MV:	Enter MV Link

Musics Information

Search By: Search...


Musics Information

Search By: Search...

MusicID: 1	MusicID / Update	MusicName: Polaroid Love	Music Name... / Update	Author: ENHYPEN	Artist... / Update	Spotify: Yes	Spotify... / Update	Apple Music: Yes	Apple Music... / Update	Joox: Yes	Joox Music... / Update

MV: https://www.youtube.com/watch?v=vRdZVDWs3BI

MV... Update

Delete

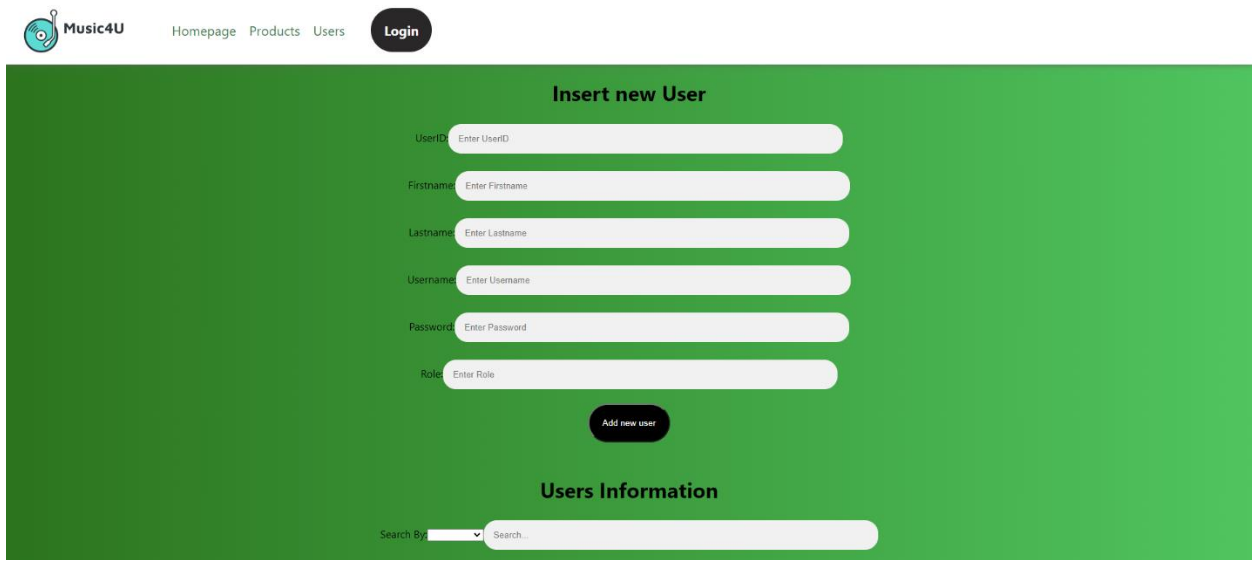
MusicID: 2	MusicID / Update	MusicName: Autumn breeze	Music Name... / Update	Author: Dept	Artist... / Update	Spotify: Yes	Spotify... / Update	Apple Music: Yes	Apple Music... / Update	Joox: Yes	Joox Music... / Update

MV: https://www.youtube.com/watch?v=2NkTMu4cEpc

MV... Update

Delete

Product page: This is a product page for administrator of our website that administrator can be able to add, insert, delete, search and view about product.



Insert new User

UserID: Enter UserID

Firstname: Enter Firstname

Lastname: Enter Lastname

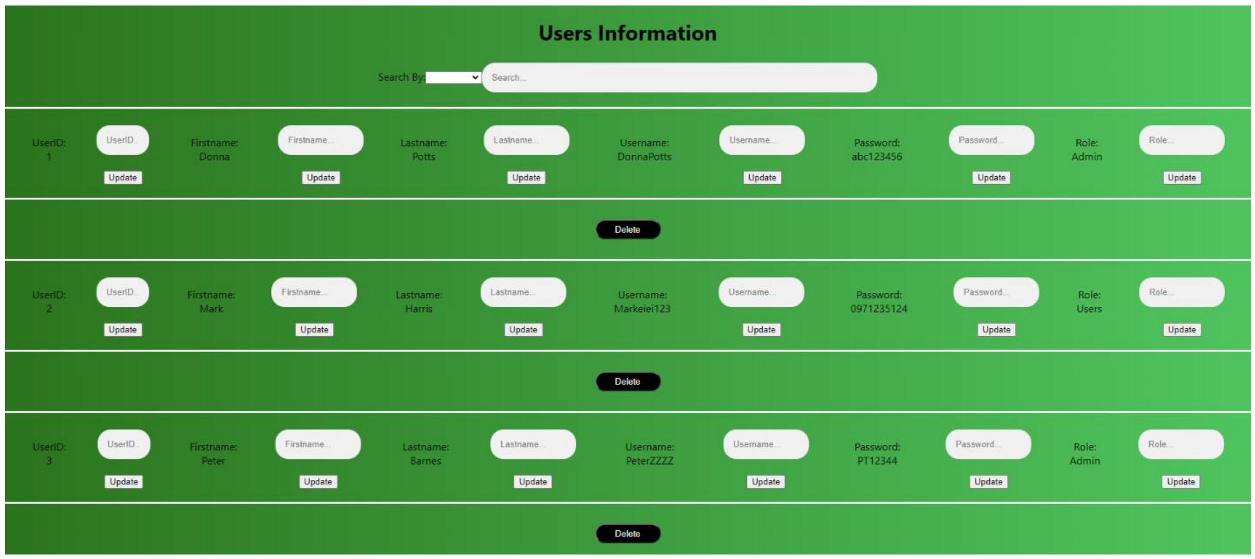
Username: Enter Username

Password: Enter Password

Role: Enter Role

Users Information

Search By: Search...



Users Information

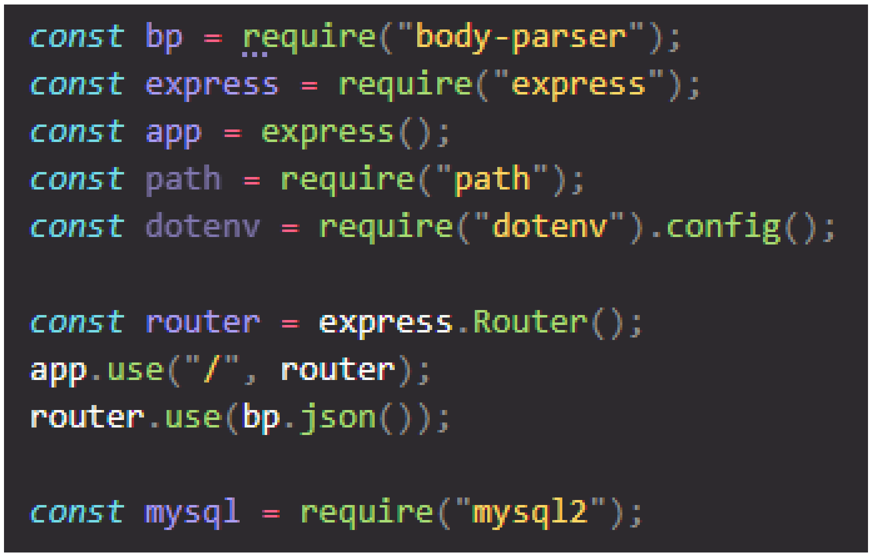
Search By: Search...

UserID: 1	Firstname: Donna	Lastname: Potts	Username: DonnaPotts	Password: abc123456	Role: Admin
UserID: 2	Firstname: Mark	Lastname: Harris	Username: Markeiei123	Password: 0971235124	Role: Users
UserID: 3	Firstname: Peter	Lastname: Barnes	Username: PeterZZZZ	Password: PT12344	Role: Admin

User page: This is a user page for the administrator of our website that the administrator can be able to add, insert, delete, search and view about user information.

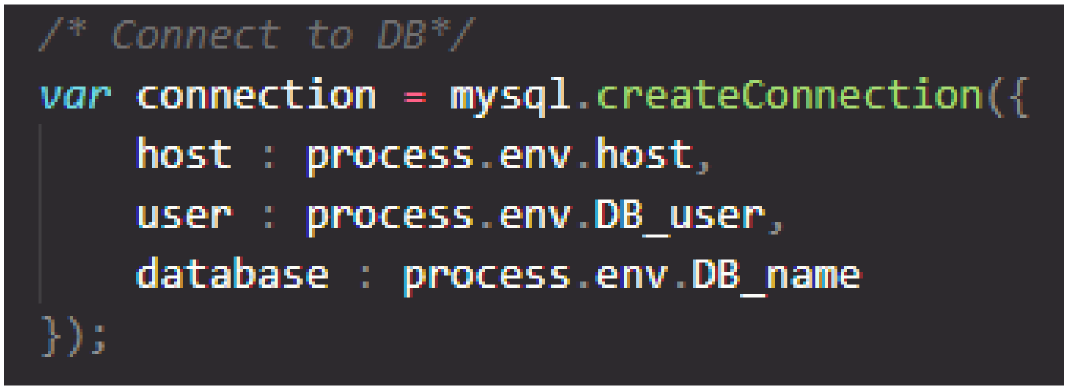
<u>Web service with code explanation (Phase II)</u>

Import modules.



```
const bp = require("body-parser");
const express = require("express");
const app = express();
const path = require("path");
const dotenv = require("dotenv").config();

const router = express.Router();
app.use("/", router);
router.use(bp.json());

const mysql = require("mysql2");
```

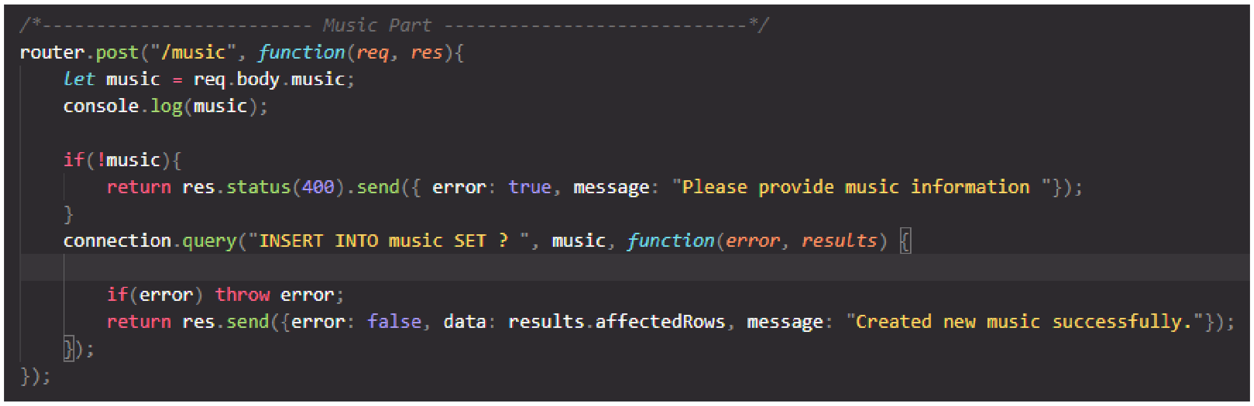
Connect to the database with the createConnection function. Next, we create variables and access from the env file and the data in the env file comes from our database.



```
/* Connect to DB*/
var connection = mysql.createConnection({
    host : process.env.host,
    user : process.env.DB_user,
    database : process.env.DB_name
});
```

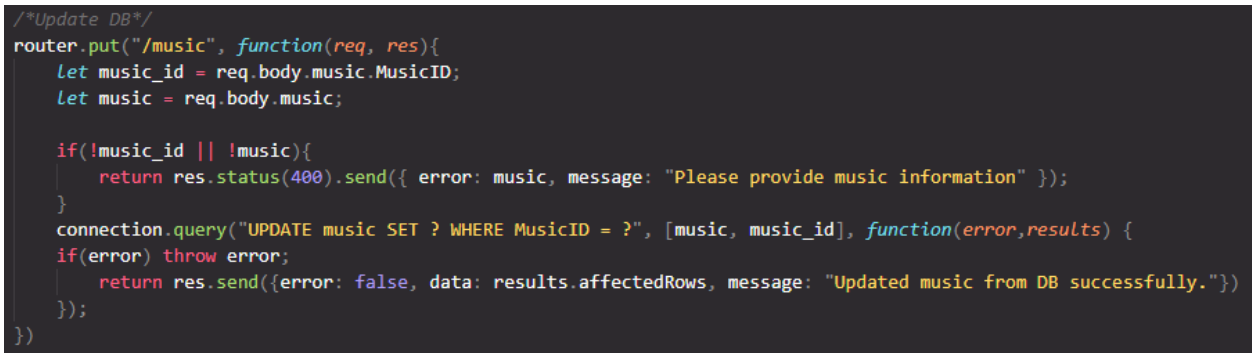
Music Part

Adding music to the database.



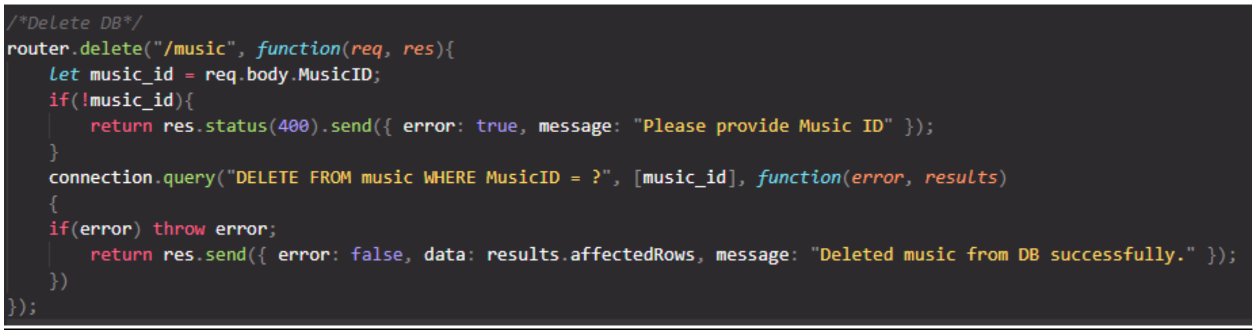
```
/*------------------------- Music Part ----------------------------*/
router.post("/music", function(req, res){
    let music = req.body.music;
    console.log(music);

    if(!music){
        return res.status(400).send({ error: true, message: "Please provide music information "});
    }
    connection.query("INSERT INTO music SET ? ", music, function(error, results) {

        if(error) throw error;
        return res.send({error: false, data: results.affectedRows, message: "Created new music successfully."});
    });
});
```

Update music from the database.



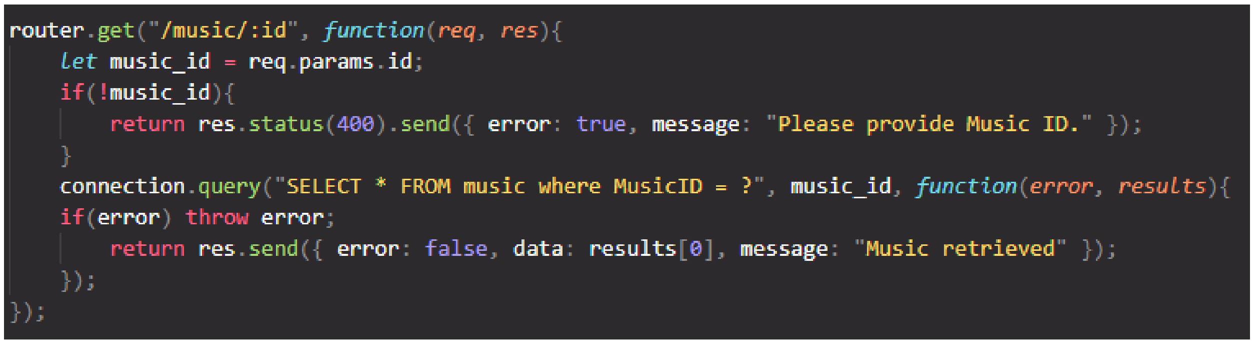
```
/*Update DB*/
router.put("/music", function(req, res){
    let music_id = req.body.music.MusicID;
    let music = req.body.music;

    if(!music_id || !music){
        return res.status(400).send({ error: music, message: "Please provide music information" });
    }
    connection.query("UPDATE music SET ? WHERE MusicID = ?", [music, music_id], function(error,results) {
    if(error) throw error;
        return res.send({error: false, data: results.affectedRows, message: "Updated music from DB successfully."})
    });
})
```

Delete music from the database.



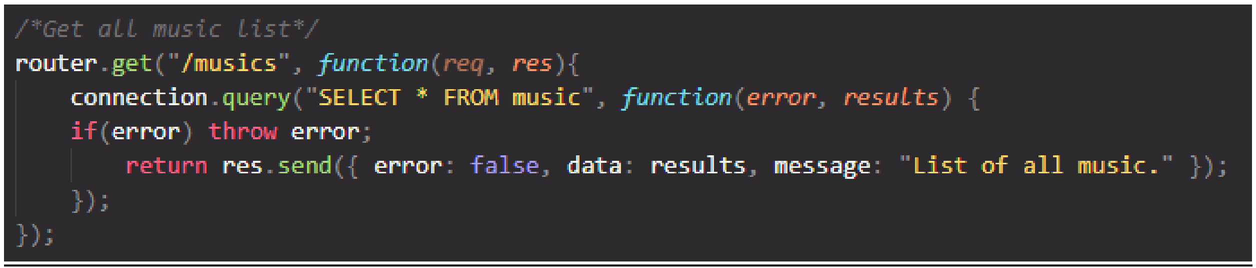
```
/*Delete DB*/
router.delete("/music", function(req, res){
    let music_id = req.body.MusicID;
    if(!music_id){
        return res.status(400).send({ error: true, message: "Please provide Music ID" });
    }
    connection.query("DELETE FROM music WHERE MusicID = ?", [music_id], function(error, results)
    {
    if(error) throw error;
        return res.send({ error: false, data: results.affectedRows, message: "Deleted music from DB successfully." });
    })
});
```

Show music information by music ID.



```
router.get("/music/:id", function(req, res){
    let music_id = req.params.id;
    if(!music_id){
        return res.status(400).send({ error: true, message: "Please provide Music ID." });
    }
    connection.query("SELECT * FROM music where MusicID = ?", music_id, function(error, results){
    if(error) throw error;
        return res.send({ error: false, data: results[0], message: "Music retrieved" });
    });
});
```

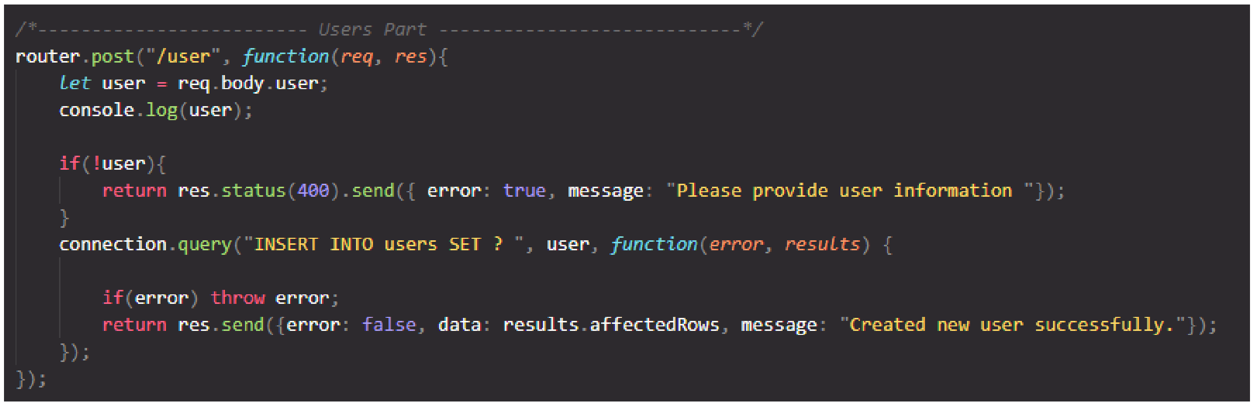
Show all music information.



```
/*Get all music list*/
router.get("/musics", function(req, res){
    connection.query("SELECT * FROM music", function(error, results) {
    if(error) throw error;
        return res.send({ error: false, data: results, message: "List of all music." });
    });
});
```

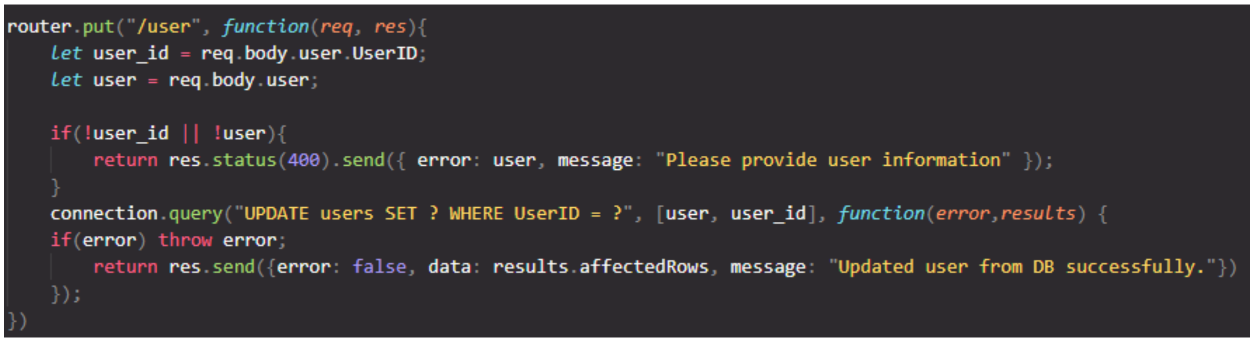
User Part

Adding user to database.



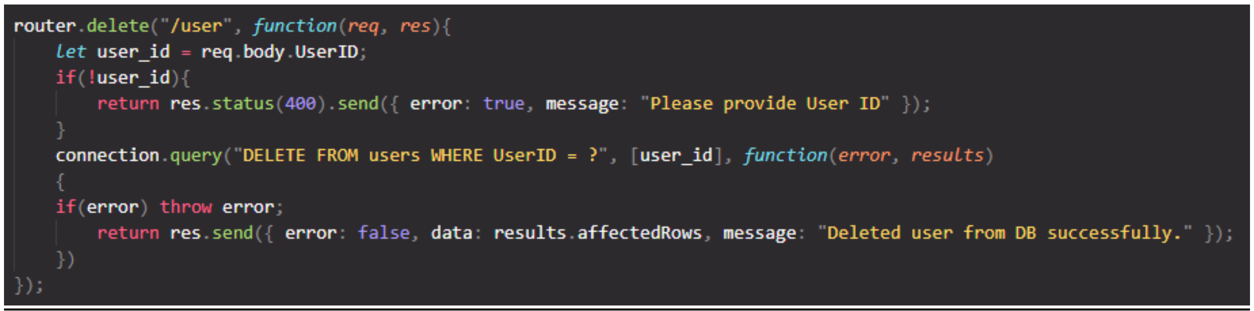
```
/*------------------------- Users Part ----------------------------*/
router.post("/user", function(req, res){
    let user = req.body.user;
    console.log(user);

    if(!user){
        return res.status(400).send({ error: true, message: "Please provide user information "});
    }
    connection.query("INSERT INTO users SET ? ", user, function(error, results) {

        if(error) throw error;
        return res.send({error: false, data: results.affectedRows, message: "Created new user successfully."});
    });
});
```

Update user from database.



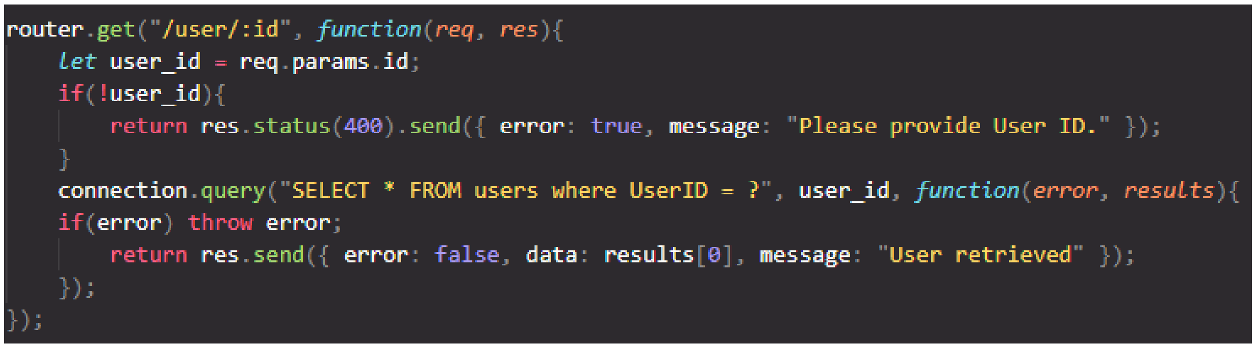
```
router.put("/user", function(req, res){
    let user_id = req.body.user.UserID;
    let user = req.body.user;

    if(!user_id || !user){
        return res.status(400).send({ error: user, message: "Please provide user information" });
    }
    connection.query("UPDATE users SET ? WHERE UserID = ?", [user, user_id], function(error,results) {
    if(error) throw error;
        return res.send({error: false, data: results.affectedRows, message: "Updated user from DB successfully."})
    });
})
```

Delete user from database.



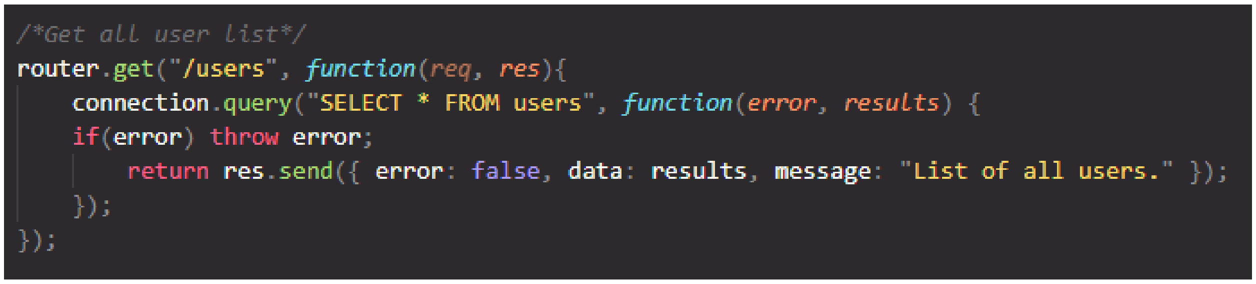
```
router.delete("/user", function(req, res){
    let user_id = req.body.UserID;
    if(!user_id){
        return res.status(400).send({ error: true, message: "Please provide User ID" });
    }
    connection.query("DELETE FROM users WHERE UserID = ?", [user_id], function(error, results)
    {
    if(error) throw error;
        return res.send({ error: false, data: results.affectedRows, message: "Deleted user from DB successfully." });
    })
});
```

Show user information by user ID.



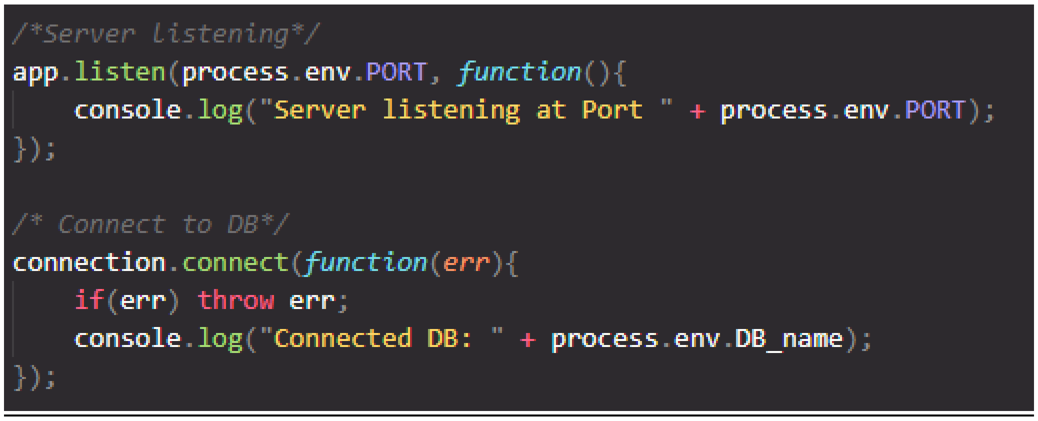
```
router.get("/user/:id", function(req, res){
    let user_id = req.params.id;
    if(!user_id){
        return res.status(400).send({ error: true, message: "Please provide User ID." });
    }
    connection.query("SELECT * FROM users where UserID = ?", user_id, function(error, results){
    if(error) throw error;
        return res.send({ error: false, data: results[0], message: "User retrieved" });
    });
});
```

Show all users information.



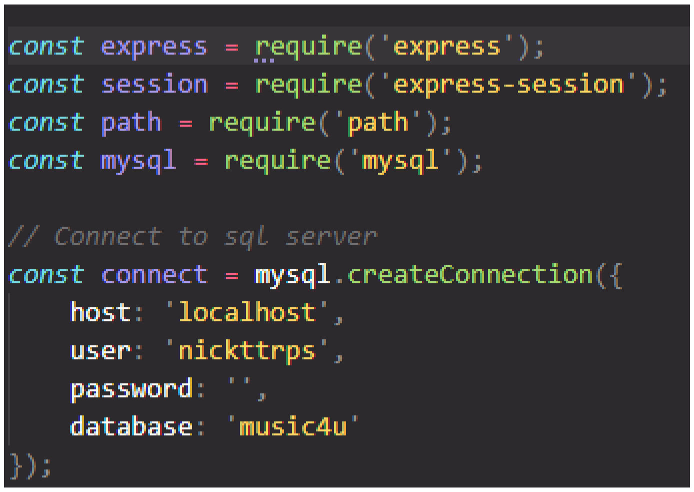
```
/*Get all user list*/
router.get("/users", function(req, res){
    connection.query("SELECT * FROM users", function(error, results) {
    if(error) throw error;
        return res.send({ error: false, data: results, message: "List of all users." });
    });
});
```

Server listening at PORT 3030 and if we can connect to the database, it will show that we are connected to our database.



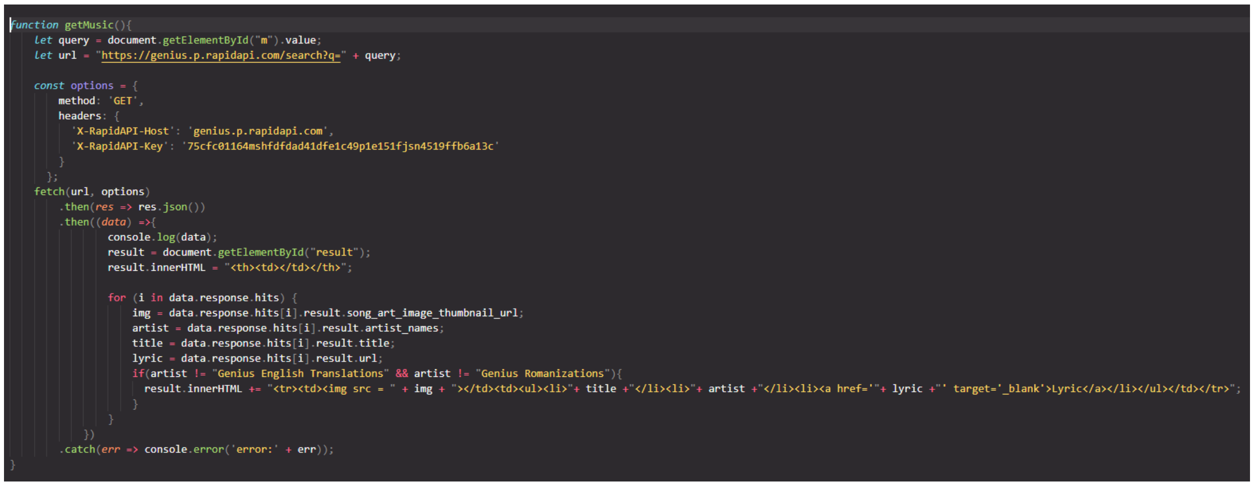
```
/*Server listening*/
app.listen(process.env.PORT, function(){
    console.log("Server listening at Port " + process.env.PORT);
});

/* Connect to DB*/
connection.connect(function(err){
    if(err) throw err;
    console.log("Connected DB: " + process.env.DB_name);
});
```

Import modules and connect to SQL Server.



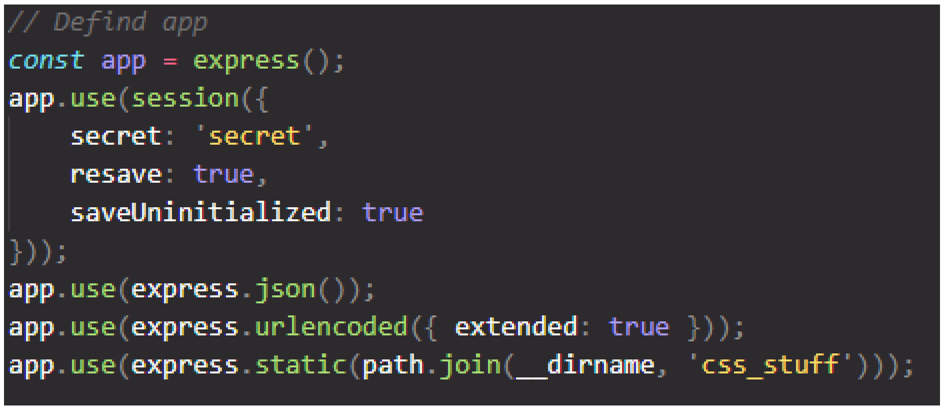
```
const express = require('express');
const session = require('express-session');
const path = require('path');
const mysql = require('mysql');

// Connect to sql server
const connect = mysql.createConnection({
    host: 'localhost',
    user: 'nickttrps',
    password: '',
    database: 'music4u'
});
```

Fetching data from public APIs for search algorithms.



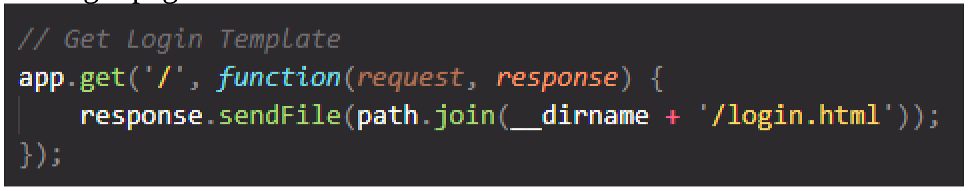
```
function getMusic(){
    let query = document.getElementById("m").value;
    let url = "https://genius.p.rapidapi.com/search?q=" + query;

    const options = {
        method: 'GET',
        headers: {
          'X-RapidAPI-Host': 'genius.p.rapidapi.com',
          'X-RapidAPI-Key': '75cfc01164mshfdfdad41dfe1c49p1e151fjsn4519ffb6a13c'
        }
      };
    fetch(url, options)
        .then(res => res.json())
        .then((data) =>{
                console.log(data);
                result = document.getElementById("result");
                result.innerHTML = "<th><td></td></th>";

                for (i in data.response.hits) {
                    img = data.response.hits[i].result.song_art_image_thumbnail_url;
                    artist = data.response.hits[i].result.artist_names;
                    title = data.response.hits[i].result.title;
                    lyric = data.response.hits[i].result.url;
                    if(artist != "Genius English Translations" && artist != "Genius Romanizations"){
                      result.innerHTML += "<tr><td><img src = " + img + "></td><td><ul><li>"+ title +"</li><li>"+ artist +"</li><li><a href='"+ lyric +"' target='_blank'>Lyric</a></li></ul></td></tr>";
                    }
                }
            })
        .catch(err => console.error('error:' + err));
}
```

Define app



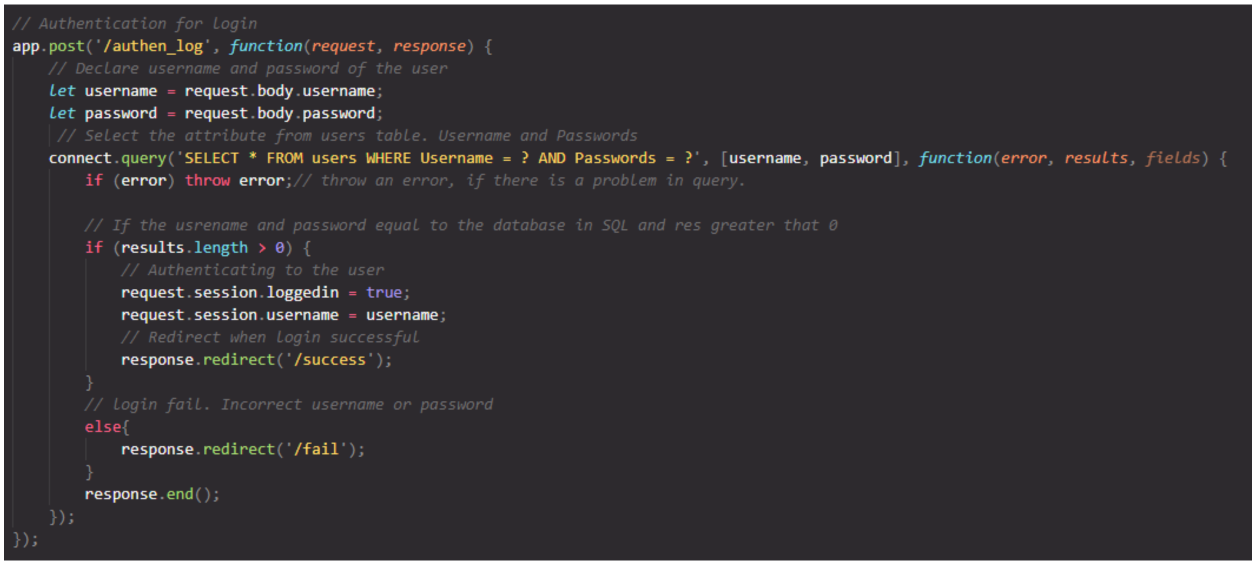
```
// Defind app
const app = express();
app.use(session({
    secret: 'secret',
    resave: true,
    saveUninitialized: true
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'css_stuff')));
```

Get login page.



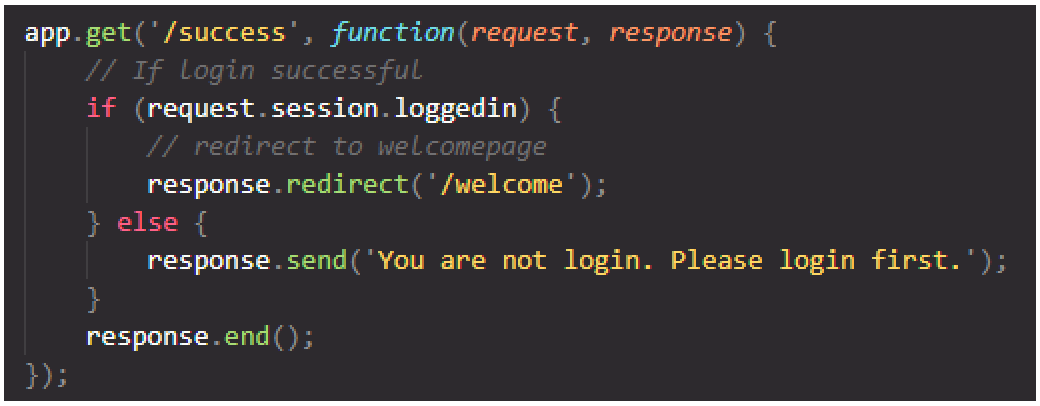
```
// Get Login Template
app.get('/', function(request, response) {
    response.sendFile(path.join(__dirname + '/login.html'));
});
```

Authentication login for users.



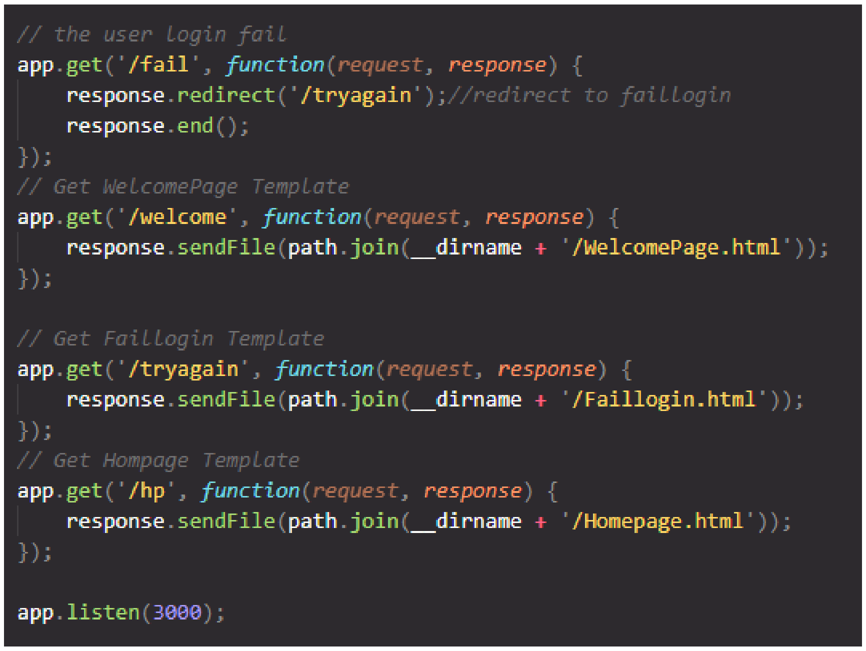
```
// Authentication for Login
app.post('/authen_log', function(request, response) {
    // Declare username and password of the user
    let username = request.body.username;
    let password = request.body.password;
     // Select the attribute from users table. Username and Passwords
    connect.query('SELECT * FROM users WHERE Username = ? AND Passwords = ?', [username, password], function(error, results, fields) {
        if (error) throw error;// throw an error, if there is a problem in query.

        // If the usrename and password equal to the database in SQL and res greater that 0
        if (results.length > 0) {
            // Authenticating to the user
            request.session.loggedin = true;
            request.session.username = username;
            // Redirect when login successful
            response.redirect('/success');
        }
        // login fail. Incorrect username or password
        else{
            response.redirect('/fail');
        }
        response.end();
    });
});
```

Get method if user successfully login and fail.



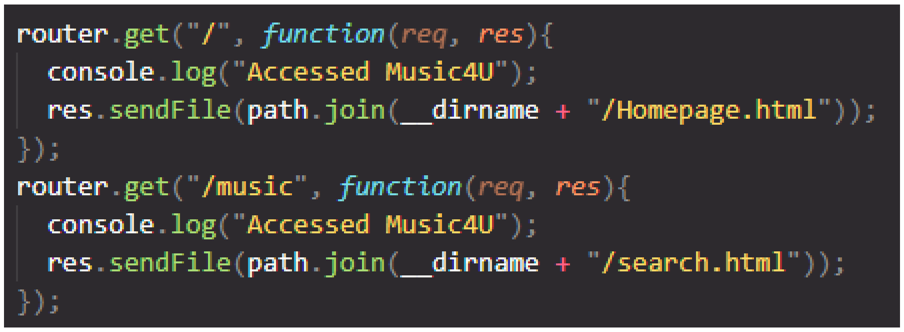
```
app.get('/success', function(request, response) {
    // If login successful
    if (request.session.loggedin) {
        // redirect to welcomepage
        response.redirect('/welcome');
    } else {
        response.send('You are not login. Please login first.');
    }
    response.end();
});
```

This is a continuous redirect, for example, if user login fails, it will come to the failure path. And then and will respond and redirect to try again path. After that, it will go to the try again path and send the response to the failure login page that we created.



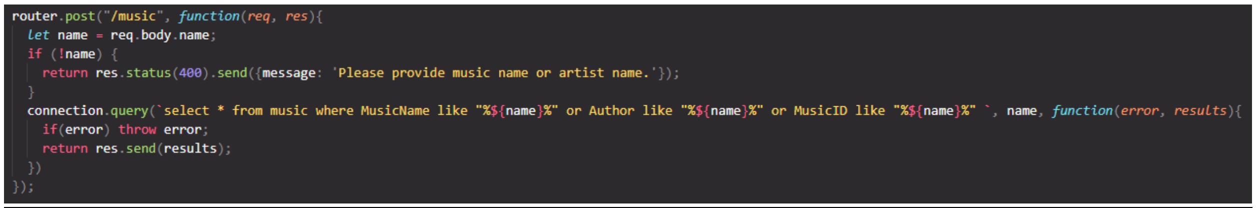
```
// the user login fail
app.get('/fail', function(request, response) {
    response.redirect('/tryagain');//redirect to faillogin
    response.end();
});
// Get WelcomePage Template
app.get('/welcome', function(request, response) {
    response.sendFile(path.join(__dirname + '/WelcomePage.html'));
});

// Get Faillogin Template
app.get('/tryagain', function(request, response) {
    response.sendFile(path.join(__dirname + '/Faillogin.html'));
});
// Get Hompage Template
app.get('/hp', function(request, response) {
    response.sendFile(path.join(__dirname + '/Homepage.html'));
});

app.listen(3000);
```

Get method that user entry and go to the home page first. And when we go to the search page and search music that user want to use, it will show the data of music and go to music path.



```
router.get("/", function(req, res){
  console.log("Accessed Music4U");
  res.sendFile(path.join(__dirname + "/Homepage.html"));
});
router.get("/music", function(req, res){
  console.log("Accessed Music4U");
  res.sendFile(path.join(__dirname + "/search.html"));
});
```

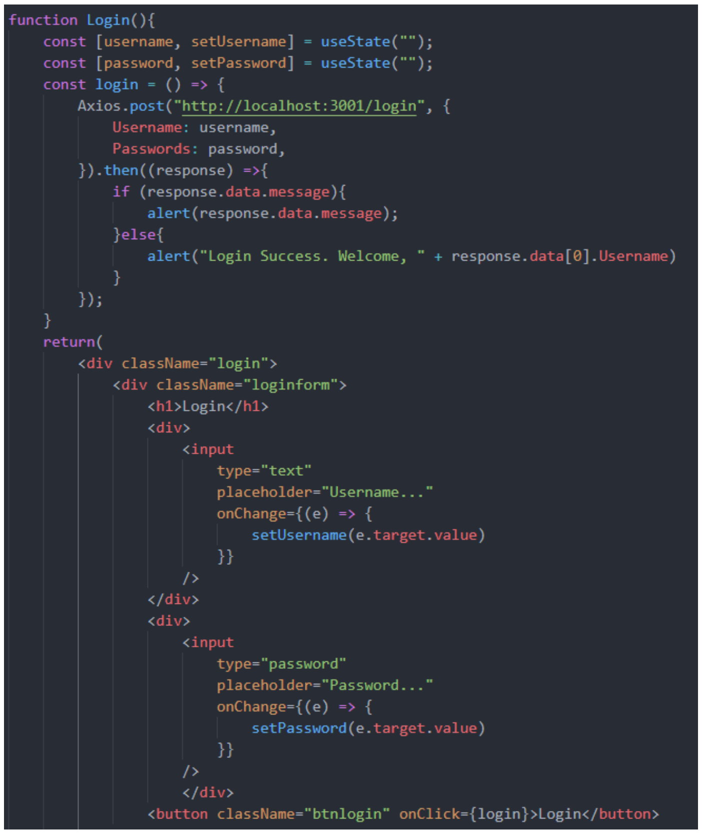
Search algorithms.



```
router.post("/music", function(req, res){
  let name = req.body.name;
  if (!name) {
    return res.status(400).send({message: 'Please provide music name or artist name.'});
  }
  connection.query(`select * from music where MusicName like "%${name}%" or Author like "%${name}%" or MusicID like "%${name}%" `, name, function(error, results){
    if(error) throw error;
    return res.send(results);
  })
});
```

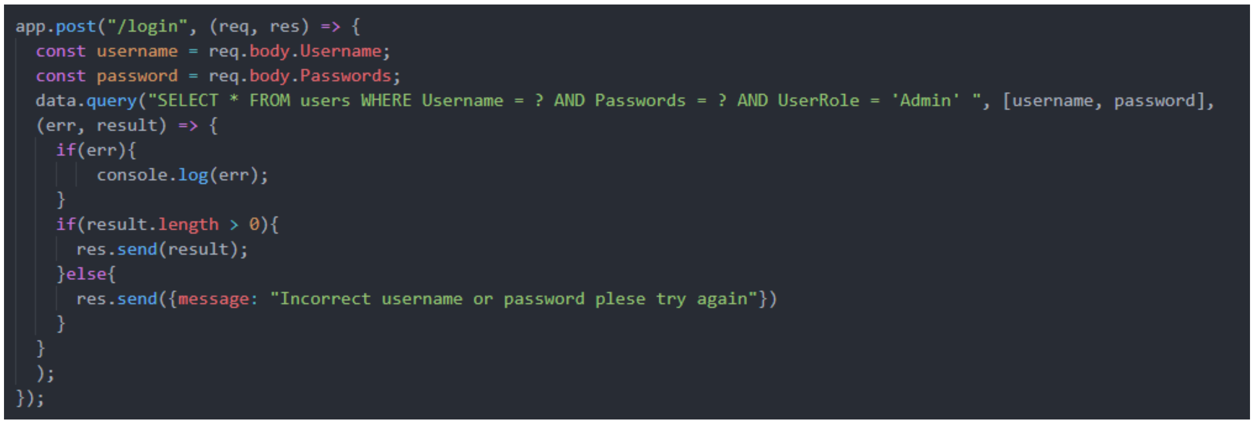
<u>Web service with code explanations (Phase III)</u>

Login service.



```
function Login(){
    const [username, setUsername] = useState("");
    const [password, setPassword] = useState("");
    const login = () => {
        Axios.post("http://localhost:3001/login", {
            Username: username,
            Passwords: password,
        }).then((response) =>{
            if (response.data.message){
                alert(response.data.message);
            }else{
                alert("Login Success. Welcome, " + response.data[0].Username)
            }
        });
    }
    return(
        <div className="login">
            <div className="loginform">
                <h1>Login</h1>
                <div>
                    <input
                        type="text"
                        placeholder="Username..."
                        onChange={(e) => {
                            setUsername(e.target.value)
                        }}
                    />
                </div>
                <div>
                    <input
                        type="password"
                        placeholder="Password..."
                        onChange={(e) => {
                            setPassword(e.target.value)
                        }}
                    />
                    </div>
                <button className="btnlogin" onClick={login}>Login</button>
```

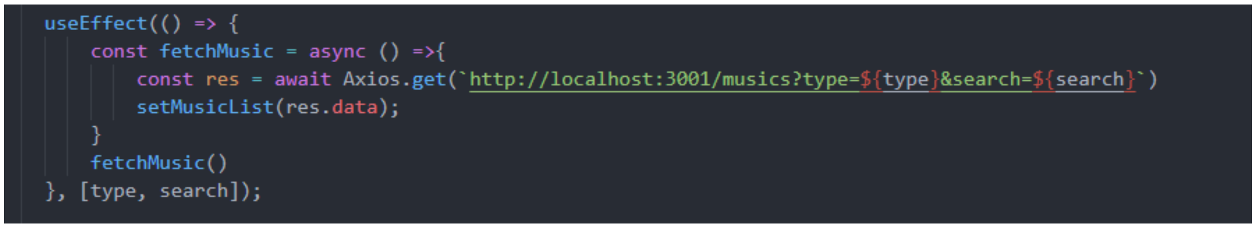
In the Login function, it will receive the username and password of the admin and it will post to localhost:3001 to check with the database.



```
app.post("/login", (req, res) => {
  const username = req.body.Username;
  const password = req.body.Passwords;
  data.query("SELECT * FROM users WHERE Username = ? AND Passwords = ? AND UserRole = 'Admin' ", [username, password],
  (err, result) => {
    if(err){
        console.log(err);
    }
    if(result.length > 0){
      res.send(result);
    }else{
      res.send({message: "Incorrect username or password plese try again"})
    }
  }
  );
});
```

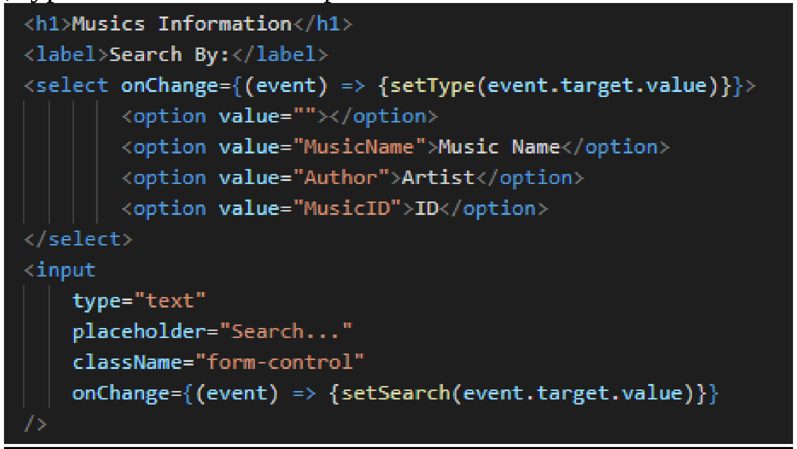
After the admin input the username and password it will request to “app.post”. If the username and password of those admin input are correct it will send the result, otherwise it will send this message “Incorrect username or password please try again”.

Search product for admin



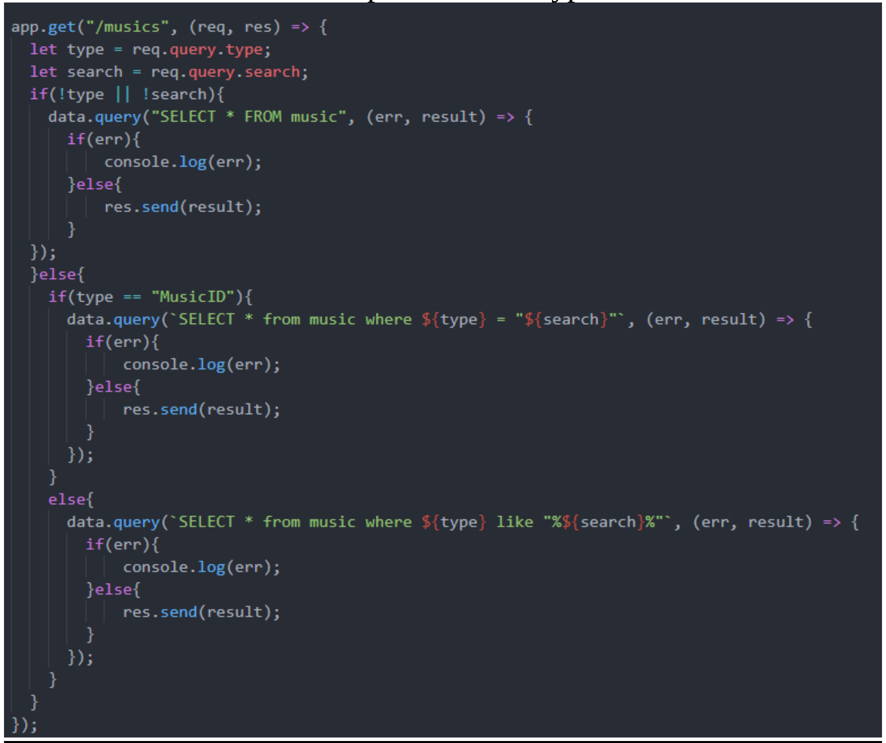
```
useEffect(() => {
    const fetchMusic = async () =>{
        const res = await Axios.get(`http://localhost:3001/musics?type=${type}&search=${search}`)
        setMusicList(res.data);
    }
    fetchMusic()
}, [type, search]);
```

Using useEffect() function to fetch the music data form the api that connects to the database. Having two query, type and search from input used for search function.



```
<h1>Musics Information</h1>
<label>Search By:</label>
<select onChange={(event) => {setType(event.target.value)}}>
        <option value=""></option>
        <option value="MusicName">Music Name</option>
        <option value="Author">Artist</option>
        <option value="MusicID">ID</option>
</select>
<input
    type="text"
    placeholder="Search..."
    className="form-control"
    onChange={(event) => {setSearch(event.target.value)}}
/>
```

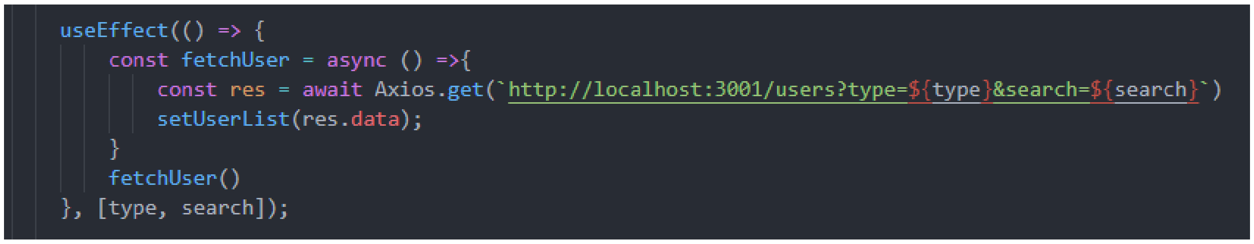
Select and Input for handle search bar and pass value of type and search variable that using in api



```
app.get("/musics", (req, res) => {
  let type = req.query.type;
  let search = req.query.search;
  if(!type || !search){
    data.query("SELECT * FROM music", (err, result) => {
      if(err){
        console.log(err);
      }else{
        res.send(result);
      }
  });
  }else{
    if(type == "MusicID"){
      data.query(`SELECT * from music where ${type} = "${search}"`, (err, result) => {
        if(err){
          console.log(err);
        }else{
          res.send(result);
        }
      });
    }
    else{
      data.query(`SELECT * from music where ${type} like "%${search}%"`, (err, result) => {
        if(err){
          console.log(err);
        }else{
          res.send(result);
        }
      });
    }
  }
});
```

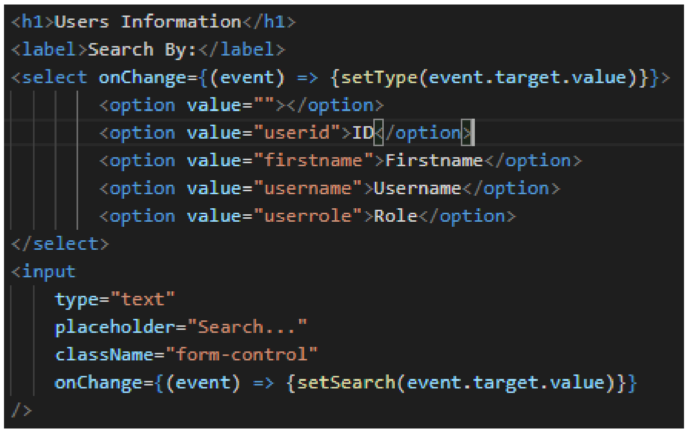
The api to get the music data from a database that requests two queries which are type and search if one of those missing it will bring all data and return it.

Search user for admin



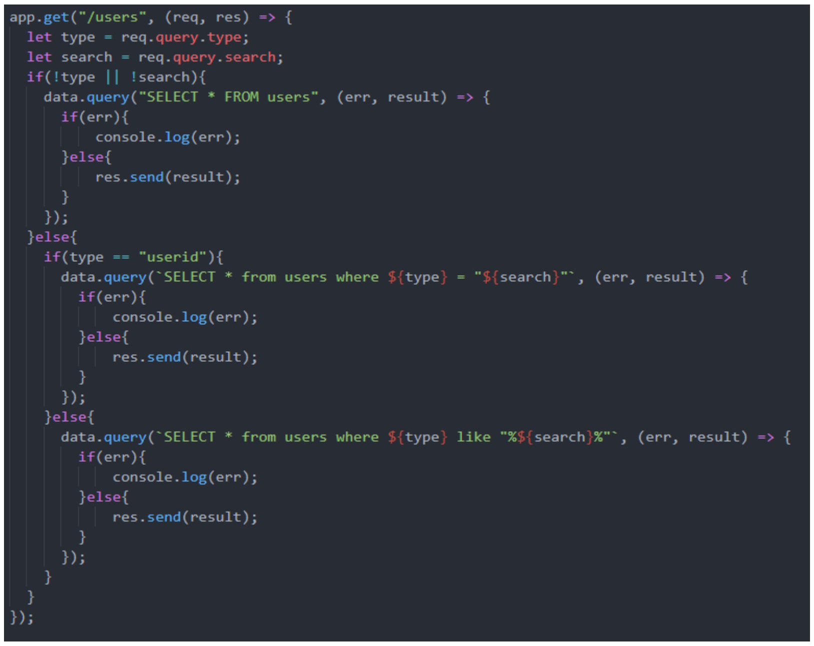
```
useEffect(() => {
    const fetchUser = async () =>{
        const res = await Axios.get(`http://localhost:3001/users?type=${type}&search=${search}`)
        setUserList(res.data);
    }
    fetchUser()
}, [type, search]);
```

Using useEffect() function to fetch the music data form the api that connects to the database. Having two query, type and search from input used for search function.



```
<h1>Users Information</h1>
<label>Search By:</label>
<select onChange={(event) => {setType(event.target.value)}}>
        <option value=""></option>
        <option value="userid">ID</option>
        <option value="firstname">Firstname</option>
        <option value="username">Username</option>
        <option value="userrole">Role</option>
</select>
<input
    type="text"
    placeholder="Search..."
    className="form-control"
    onChange={(event) => {setSearch(event.target.value)}}
/>
```

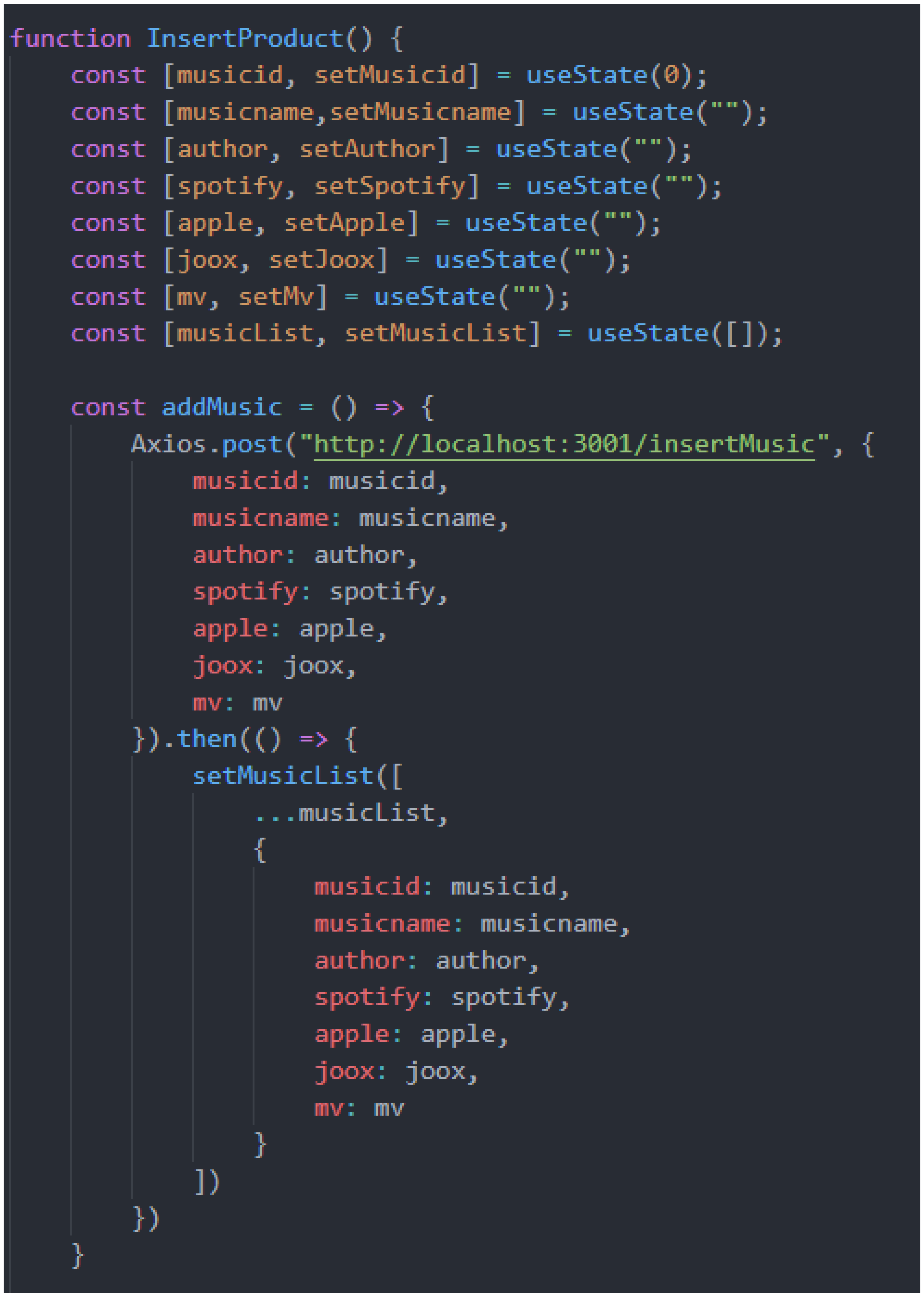
Select and Input for handle search bar and pass value of type and search variable that using in api



```
app.get("/users", (req, res) => {
  let type = req.query.type;
  let search = req.query.search;
  if(!type || !search){
    data.query("SELECT * FROM users", (err, result) => {
      if(err){
          console.log(err);
      }else{
          res.send(result);
      }
    });
  }else{
    if(type == "userid"){
      data.query(`SELECT * from users where ${type} = "${search}"`, (err, result) => {
        if(err){
            console.log(err);
        }else{
            res.send(result);
        }
      });
    }else{
      data.query(`SELECT * from users where ${type} like "%${search}%"`, (err, result) => {
        if(err){
            console.log(err);
        }else{
            res.send(result);
        }
      });
    }
  }
});
```

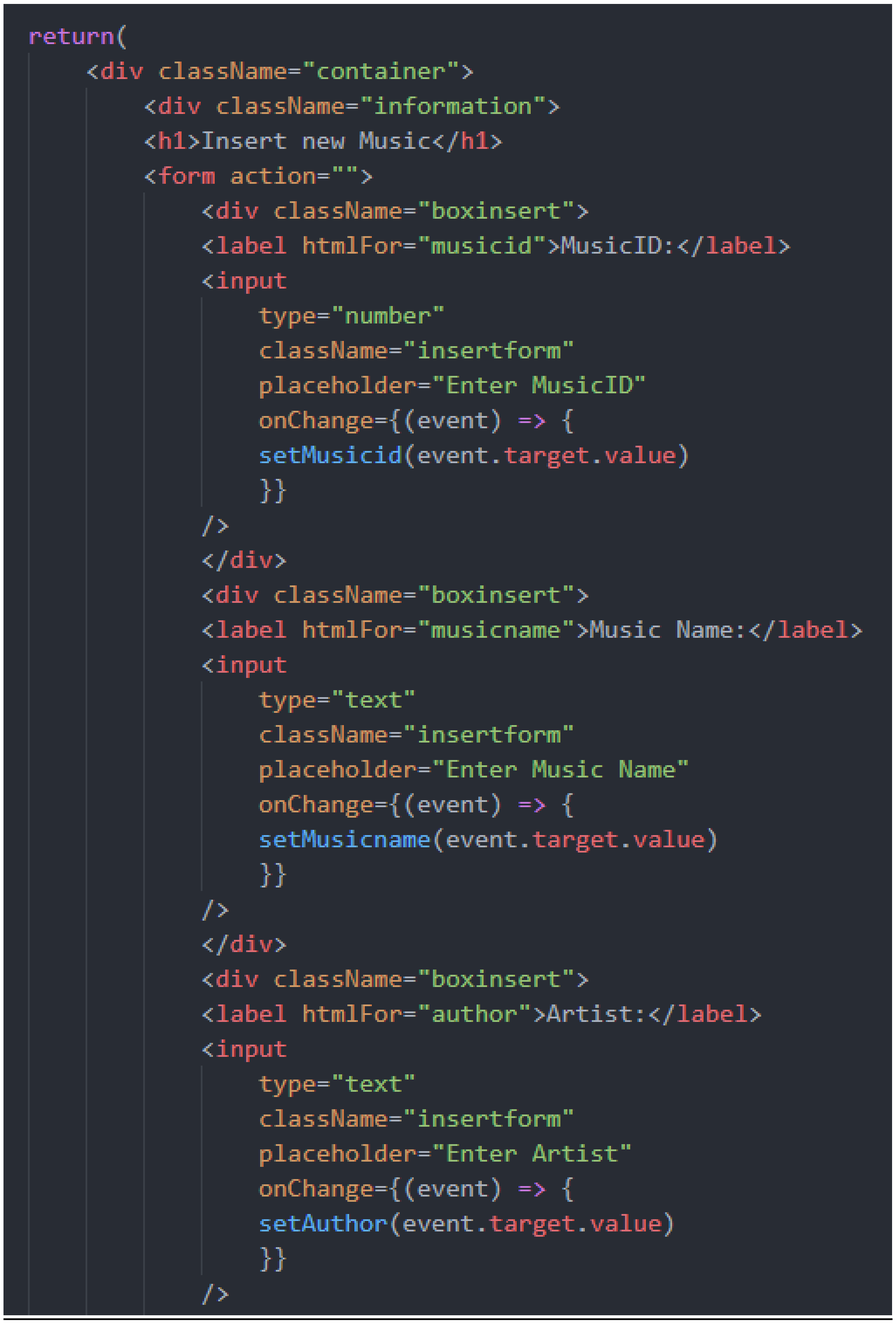
The api to get the user data from a database that requests two queries which are type and search if one of those missing it will bring all data and return it.

Insert product

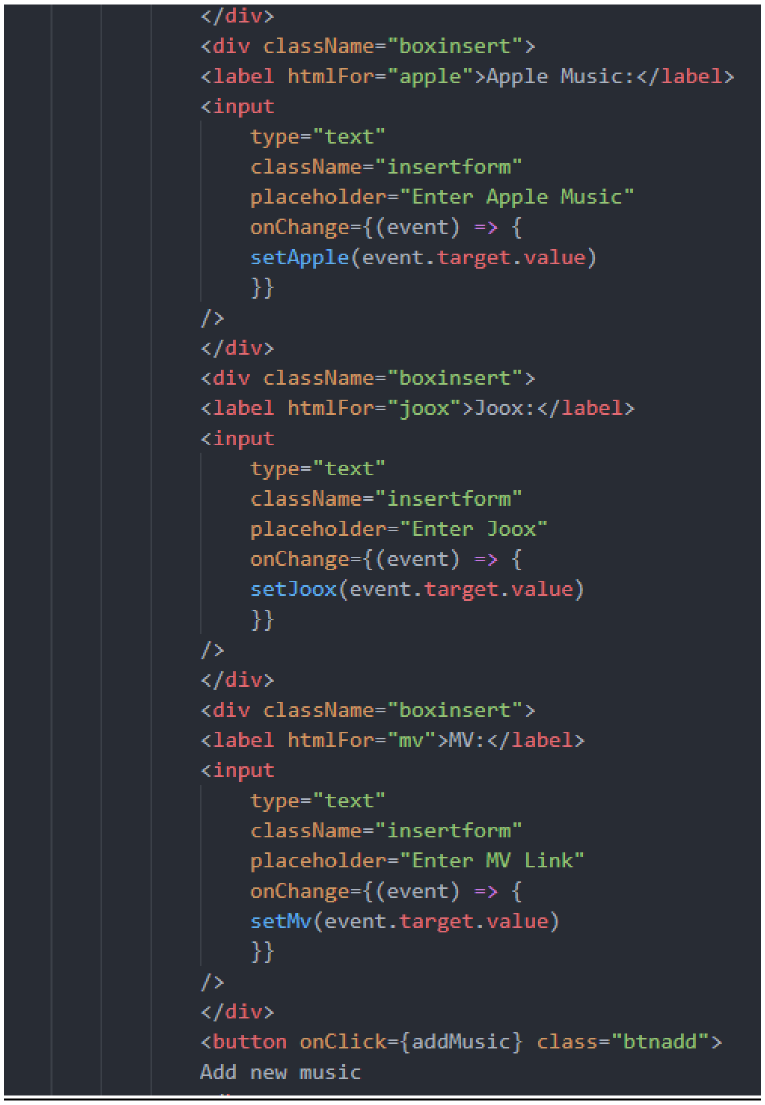


```
function InsertProduct() {
    const [musicid, setMusicid] = useState(0);
    const [musicname,setMusicname] = useState("");
    const [author, setAuthor] = useState("");
    const [spotify, setSpotify] = useState("");
    const [apple, setApple] = useState("");
    const [joox, setJoox] = useState("");
    const [mv, setMv] = useState("");
    const [musicList, setMusicList] = useState([]);

    const addMusic = () => {
        Axios.post("http://localhost:3001/insertMusic", {
            musicid: musicid,
            musicname: musicname,
            author: author,
            spotify: spotify,
            apple: apple,
            joox: joox,
            mv: mv
        }).then(() => {
            setMusicList([
                ...musicList,
                {
                    musicid: musicid,
                    musicname: musicname,
                    author: author,
                    spotify: spotify,
                    apple: apple,
                    joox: joox,
                    mv: mv
                }
            ])
        })
    }
```

For the addMusic function, it will use axios.post to post the new music that admin inserts to "localhost:3001/insertMusic" to create new music in the database.

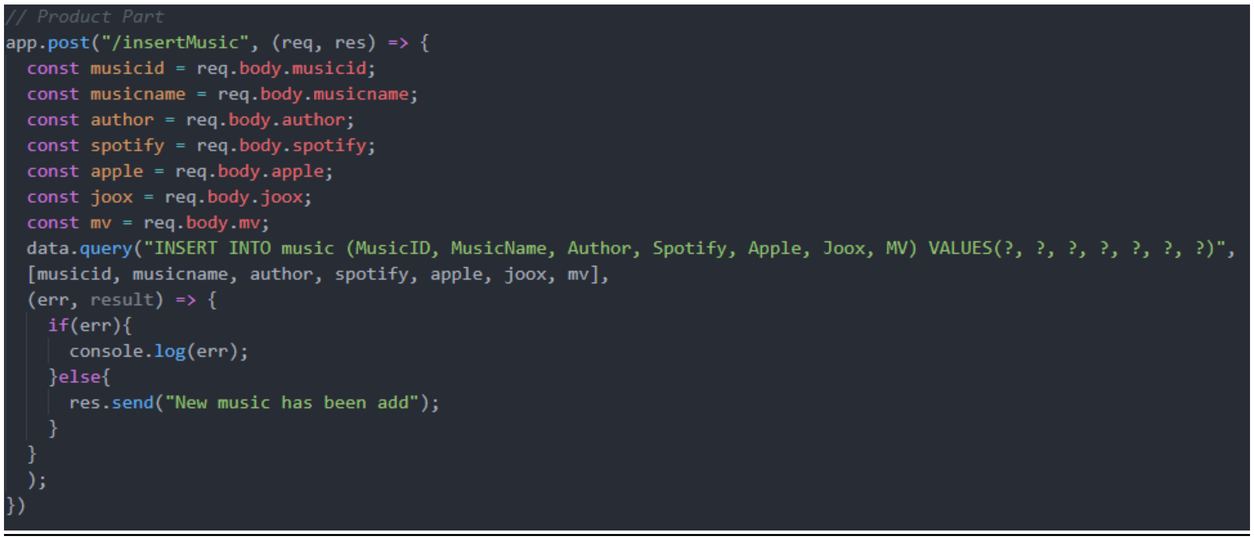


```
return(
    <div className="container">
        <div className="information">
        <h1>Insert new Music</h1>
        <form action="">
            <div className="boxinsert">
            <label htmlFor="musicid">MusicID:</label>
            <input
                type="number"
                className="insertform"
                placeholder="Enter MusicID"
                onChange={(event) => {
                setMusicid(event.target.value)
                }}
            />
            </div>
            <div className="boxinsert">
            <label htmlFor="musicname">Music Name:</label>
            <input
                type="text"
                className="insertform"
                placeholder="Enter Music Name"
                onChange={(event) => {
                setMusicname(event.target.value)
                }}
            />
            </div>
            <div className="boxinsert">
            <label htmlFor="author">Artist:</label>
            <input
                type="text"
                className="insertform"
                placeholder="Enter Artist"
                onChange={(event) => {
                setAuthor(event.target.value)
                }}
            />
```



```
</div>
<div className="boxinsert">
<label htmlFor="apple">Apple Music:</label>
<input
    type="text"
    className="insertform"
    placeholder="Enter Apple Music"
    onChange={(event) => {
    setApple(event.target.value)
    }}
/>
</div>
<div className="boxinsert">
<label htmlFor="joox">Joox:</label>
<input
    type="text"
    className="insertform"
    placeholder="Enter Joox"
    onChange={(event) => {
    setJoox(event.target.value)
    }}
/>
</div>
<div className="boxinsert">
<label htmlFor="mv">MV:</label>
<input
    type="text"
    className="insertform"
    placeholder="Enter MV Link"
    onChange={(event) => {
    setMv(event.target.value)
    }}
/>
</div>
<button onClick={addMusic} class="btnadd">
Add new music
```

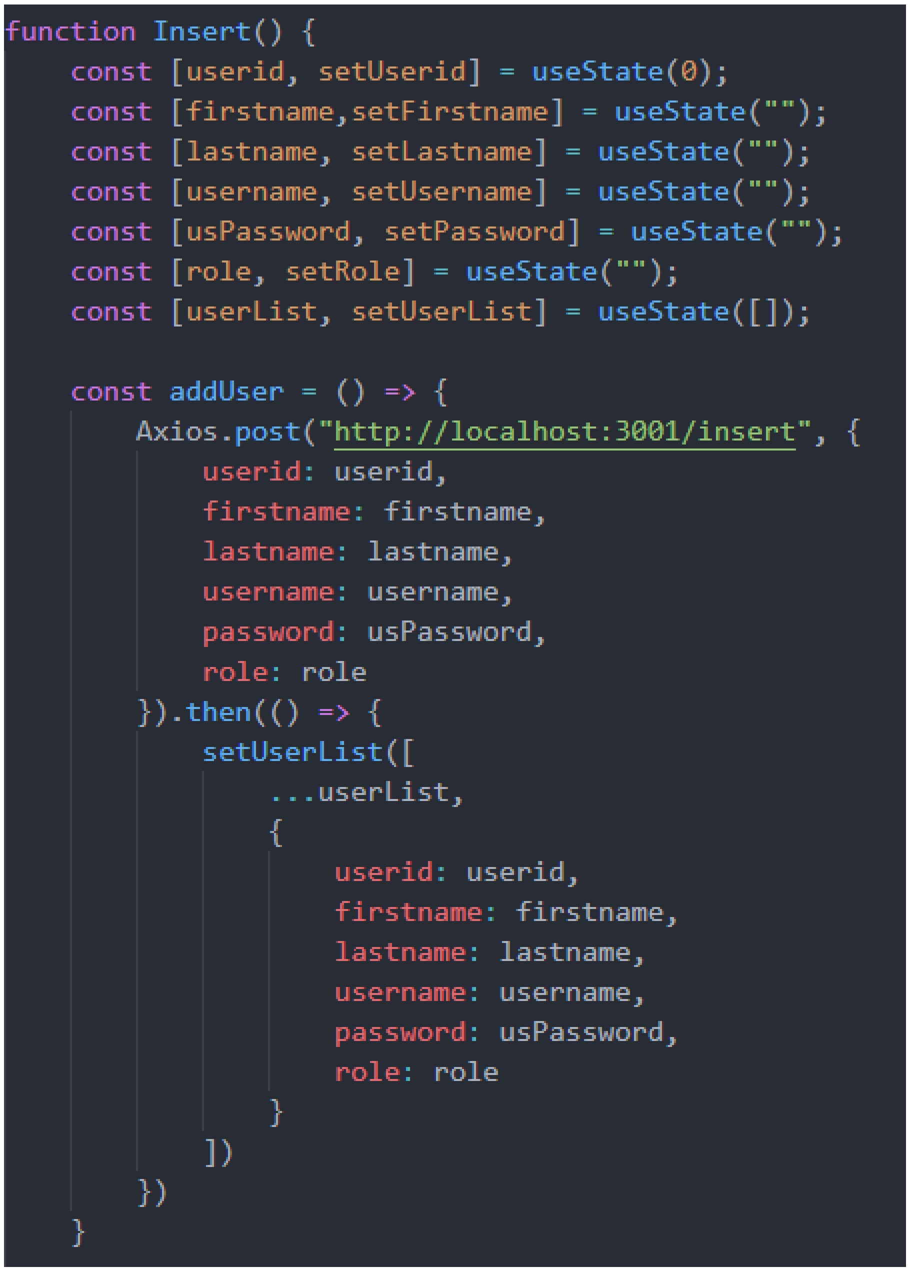
Admin can add new music by typing the data that admin would like to add in the text box and click the button to add new music.



```
// Product Part
app.post("/insertMusic", (req, res) => {
  const musicid = req.body.musicid;
  const musicname = req.body.musicname;
  const author = req.body.author;
  const spotify = req.body.spotify;
  const apple = req.body.apple;
  const joox = req.body.joox;
  const mv = req.body.mv;
  data.query("INSERT INTO music (MusicID, MusicName, Author, Spotify, Apple, Joox, MV) VALUES(?, ?, ?, ?, ?, ?, ?)",
  [musicid, musicname, author, spotify, apple, joox, mv],
  (err, result) => {
    if(err){
      console.log(err);
    }else{
      res.send("New music has been add");
    }
  }
  );
})
```

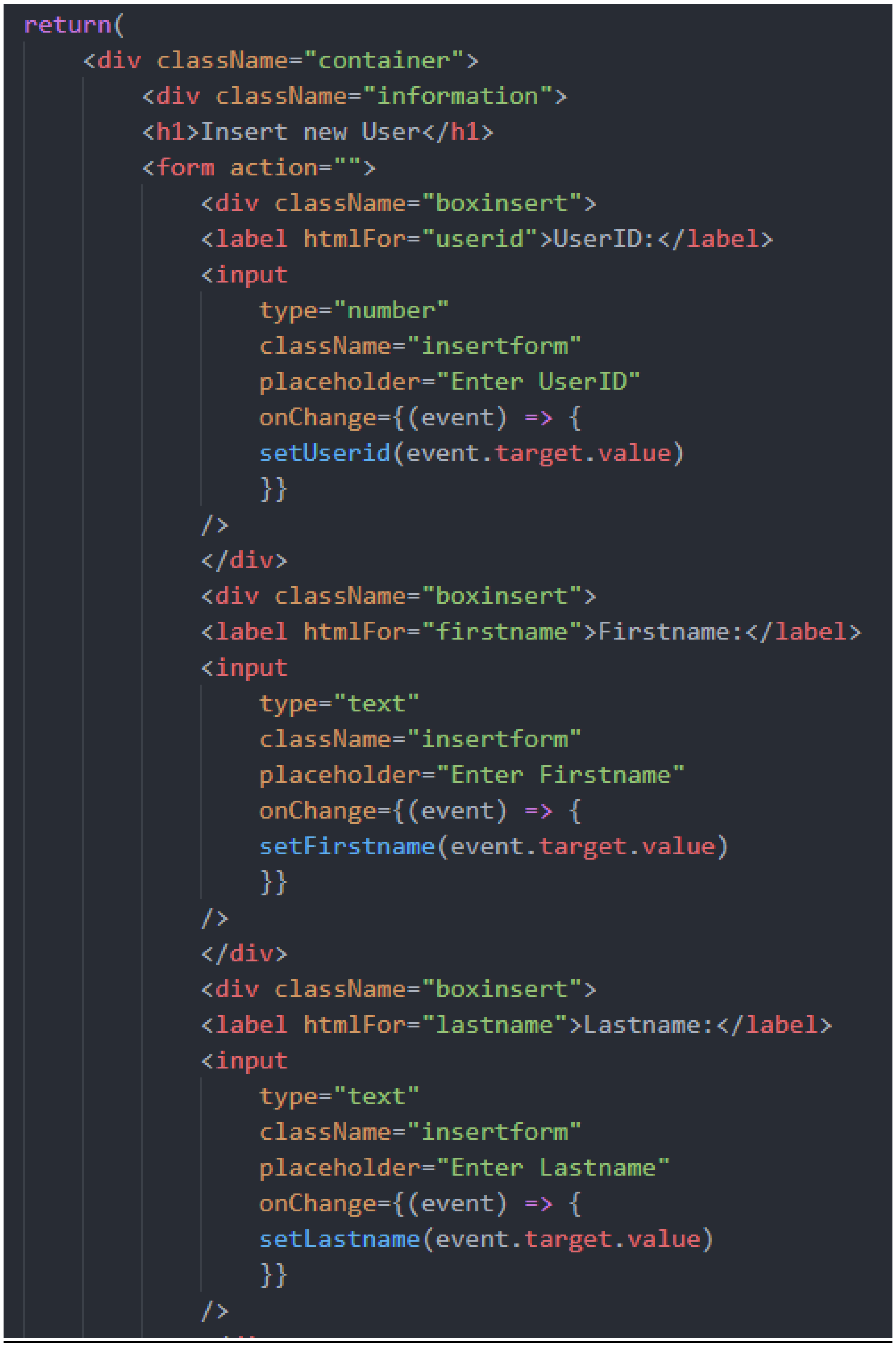
When the admin inserts new music to the database it will use “app.post” to create new music in the database.

Insert User

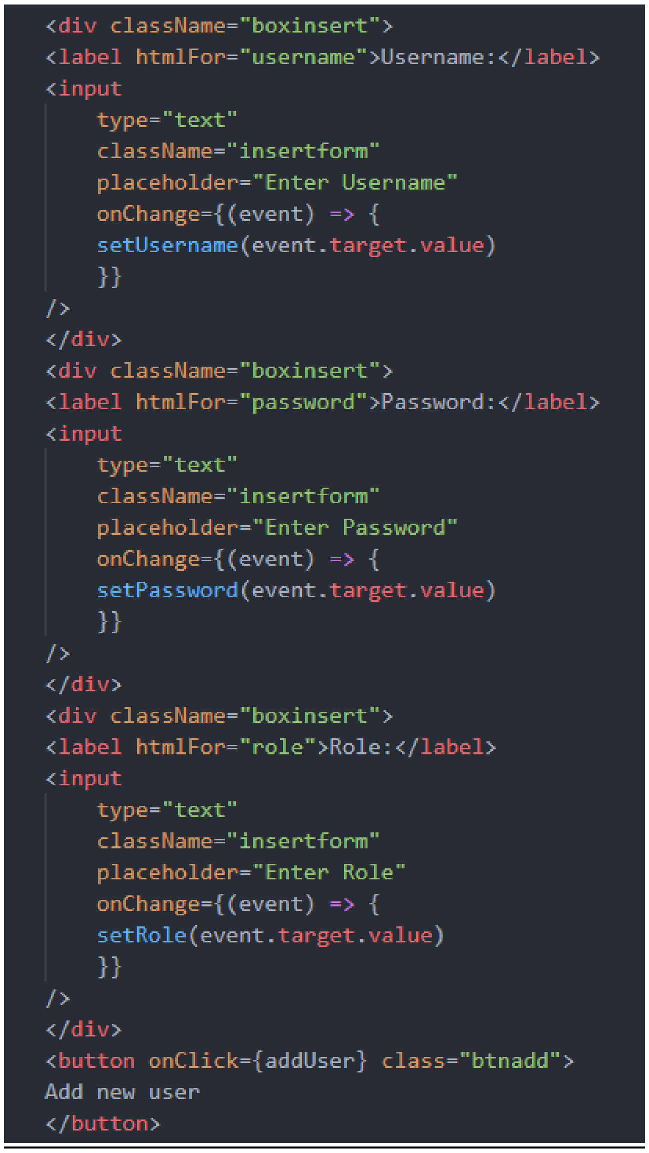


```
function Insert() {
    const [userid, setUserid] = useState(0);
    const [firstname,setFirstname] = useState("");
    const [lastname, setLastname] = useState("");
    const [username, setUsername] = useState("");
    const [usPassword, setPassword] = useState("");
    const [role, setRole] = useState("");
    const [userList, setUserList] = useState([]);

    const addUser = () => {
        Axios.post("http://localhost:3001/insert", {
            userid: userid,
            firstname: firstname,
            lastname: lastname,
            username: username,
            password: usPassword,
            role: role
        }).then(() => {
            setUserList([
                ...userList,
                {
                    userid: userid,
                    firstname: firstname,
                    lastname: lastname,
                    username: username,
                    password: usPassword,
                    role: role
                }
            ])
        })
    }
```

For the addUser function, it will use axios.post to post the new user that admin inserts to “localhost:3001/insert” to create a new user in the database.

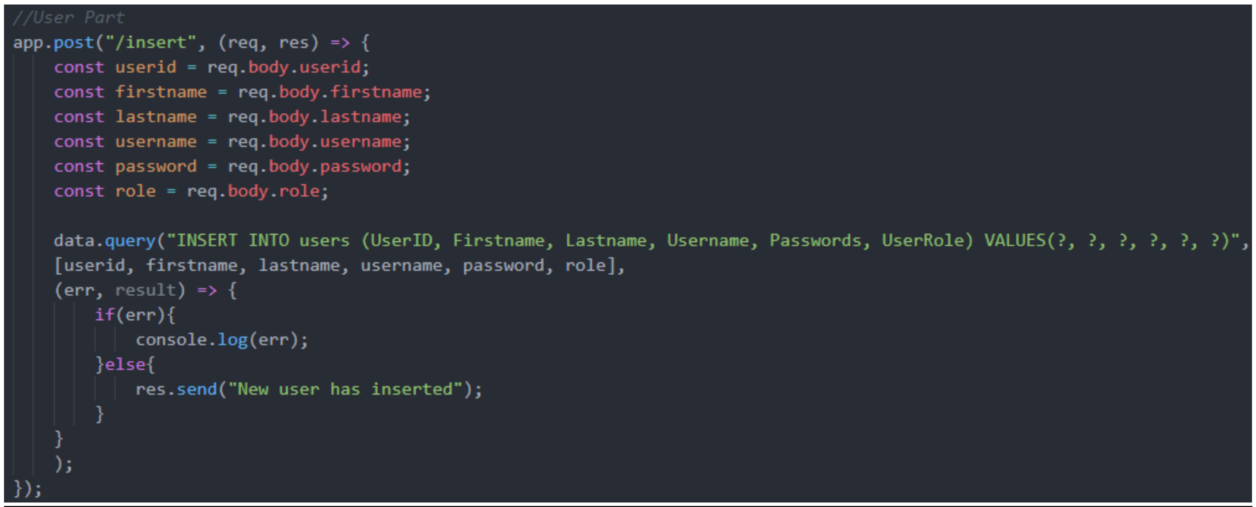


```
return(
    <div className="container">
        <div className="information">
        <h1>Insert new User</h1>
        <form action="">
            <div className="boxinsert">
            <label htmlFor="userid">UserID:</label>
            <input
                type="number"
                className="insertform"
                placeholder="Enter UserID"
                onChange={(event) => {
                setUserid(event.target.value)
                }}
            />
            </div>
            <div className="boxinsert">
            <label htmlFor="firstname">Firstname:</label>
            <input
                type="text"
                className="insertform"
                placeholder="Enter Firstname"
                onChange={(event) => {
                setFirstname(event.target.value)
                }}
            />
            </div>
            <div className="boxinsert">
            <label htmlFor="lastname">Lastname:</label>
            <input
                type="text"
                className="insertform"
                placeholder="Enter Lastname"
                onChange={(event) => {
                setLastname(event.target.value)
                }}
            />
```



```
<div className="boxinsert">
<label htmlFor="username">Username:</label>
<input
    type="text"
    className="insertform"
    placeholder="Enter Username"
    onChange={(event) => {
    setUsername(event.target.value)
    }}
/>
</div>
<div className="boxinsert">
<label htmlFor="password">Password:</label>
<input
    type="text"
    className="insertform"
    placeholder="Enter Password"
    onChange={(event) => {
    setPassword(event.target.value)
    }}
/>
</div>
<div className="boxinsert">
<label htmlFor="role">Role:</label>
<input
    type="text"
    className="insertform"
    placeholder="Enter Role"
    onChange={(event) => {
    setRole(event.target.value)
    }}
/>
</div>
<button onClick={addUser} class="btnadd">
Add new user
</button>
```

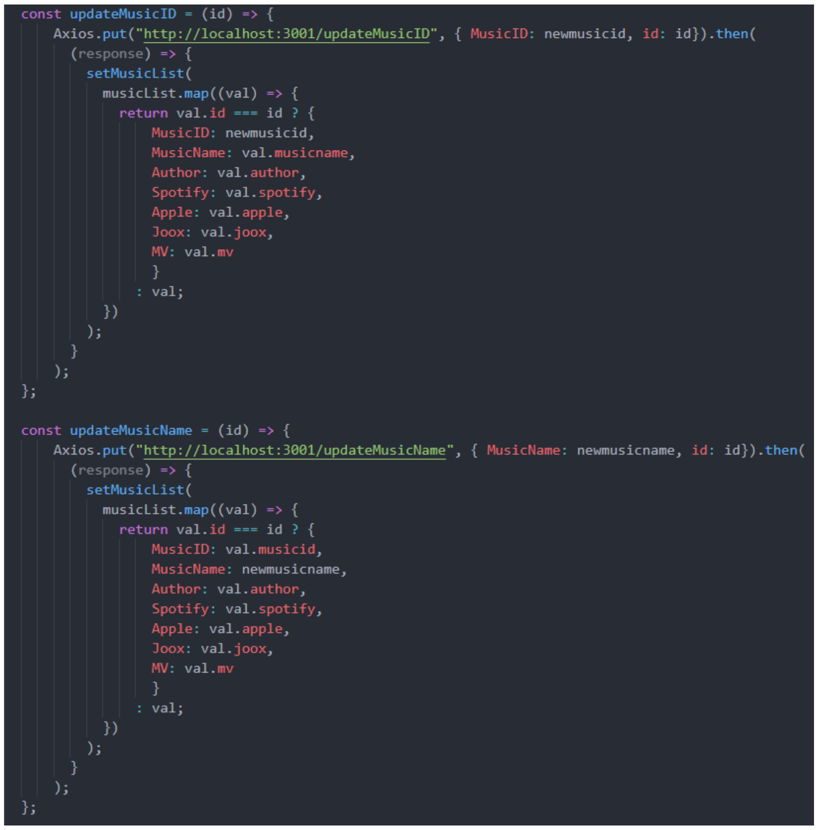
Admin can add new users by typing the data that admin would like to add in the text box and click the button to add a new user.



```
//User Part
app.post("/insert", (req, res) => {
    const userid = req.body.userid;
    const firstname = req.body.firstname;
    const lastname = req.body.lastname;
    const username = req.body.username;
    const password = req.body.password;
    const role = req.body.role;

    data.query("INSERT INTO users (UserID, Firstname, Lastname, Username, Passwords, UserRole) VALUES(?, ?, ?, ?, ?, ?)",
    [userid, firstname, lastname, username, password, role],
    (err, result) => {
        if(err){
            console.log(err);
        }else{
            res.send("New user has inserted");
        }
    }
    );
});
```

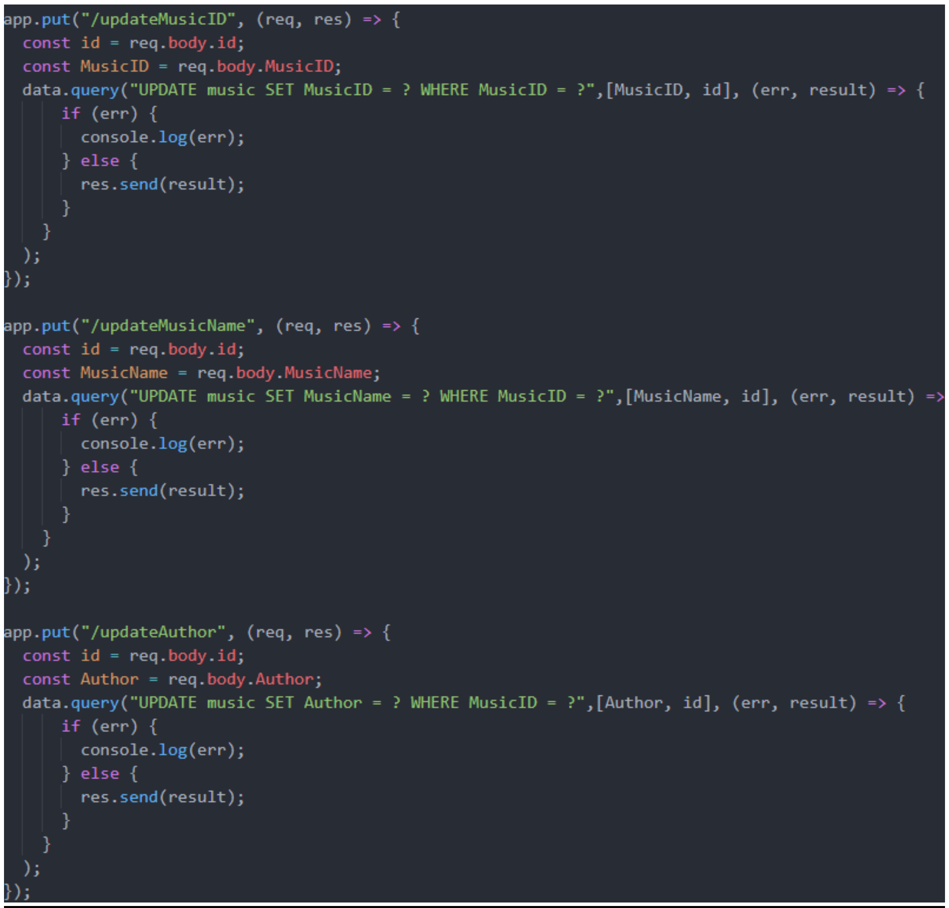
When the admin inserts a new user to the database it will use “app.post” to create a new user in the database.

Update product



```
const updateMusicID = (id) => {
    Axios.put("http://localhost:3001/updateMusicID", { MusicID: newmusicid, id: id}).then(
      (response) => {
        setMusicList(
          musicList.map((val) => {
            return val.id === id ? {
                MusicID: newmusicid,
                MusicName: val.musicname,
                Author: val.author,
                Spotify: val.spotify,
                Apple: val.apple,
                Joox: val.joox,
                MV: val.mv
                }
              : val;
          })
        );
      }
    );
};

const updateMusicName = (id) => {
    Axios.put("http://localhost:3001/updateMusicName", { MusicName: newmusicname, id: id}).then(
      (response) => {
        setMusicList(
          musicList.map((val) => {
            return val.id === id ? {
                MusicID: val.musicid,
                MusicName: newmusicname,
                Author: val.author,
                Spotify: val.spotify,
                Apple: val.apple,
                Joox: val.joox,
                MV: val.mv
                }
              : val;
          })
        );
      }
    );
};
```

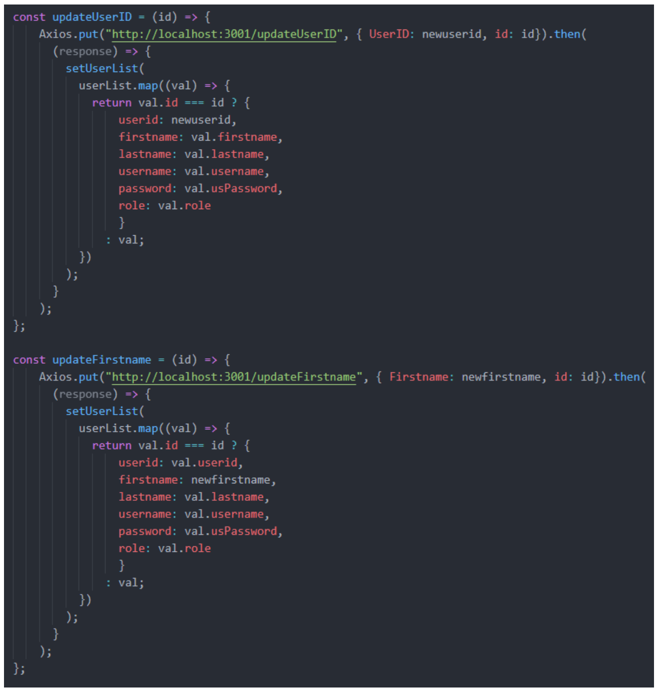
In the update product function, it will use axios.put to update the new product’s data that the admin chooses to update, and it will put to “localhost:3001” to update the database.



```
app.put("/updateMusicID", (req, res) => {
  const id = req.body.id;
  const MusicID = req.body.MusicID;
  data.query("UPDATE music SET MusicID = ? WHERE MusicID = ?",[MusicID, id], (err, result) => {
      if (err) {
        console.log(err);
      } else {
        res.send(result);
      }
    }
  );
});

app.put("/updateMusicName", (req, res) => {
  const id = req.body.id;
  const MusicName = req.body.MusicName;
  data.query("UPDATE music SET MusicName = ? WHERE MusicID = ?",[MusicName, id], (err, result) =>
      if (err) {
        console.log(err);
      } else {
        res.send(result);
      }
    }
  );
});

app.put("/updateAuthor", (req, res) => {
  const id = req.body.id;
  const Author = req.body.Author;
  data.query("UPDATE music SET Author = ? WHERE MusicID = ?",[Author, id], (err, result) => {
      if (err) {
        console.log(err);
      } else {
        res.send(result);
      }
    }
  );
});
```

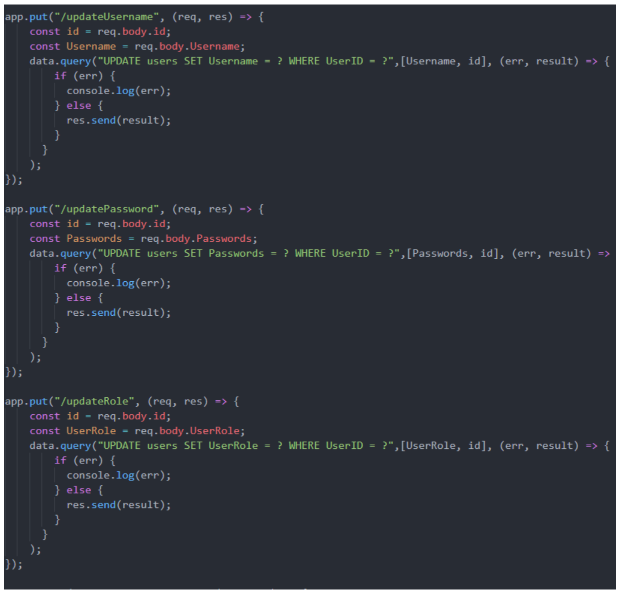
When the admin updates a product’s data to the database it will use “app.put” to update it in the database.

Update user



```
const updateUserID = (id) => {
    Axios.put("http://localhost:3001/updateUserID", { UserID: newuserid, id: id}).then(
      (response) => {
        setUserList(
          userList.map((val) => {
            return val.id === id ? {
                userid: newuserid,
                firstname: val.firstname,
                lastname: val.lastname,
                username: val.username,
                password: val.usPassword,
                role: val.role
                }
              : val;
          })
        );
      }
    );
};

const updateFirstname = (id) => {
    Axios.put("http://localhost:3001/updateFirstname", { Firstname: newfirstname, id: id}).then(
      (response) => {
        setUserList(
          userList.map((val) => {
            return val.id === id ? {
                userid: val.userid,
                firstname: newfirstname,
                lastname: val.lastname,
                username: val.username,
                password: val.usPassword,
                role: val.role
                }
              : val;
          })
        );
      }
    );
};
```

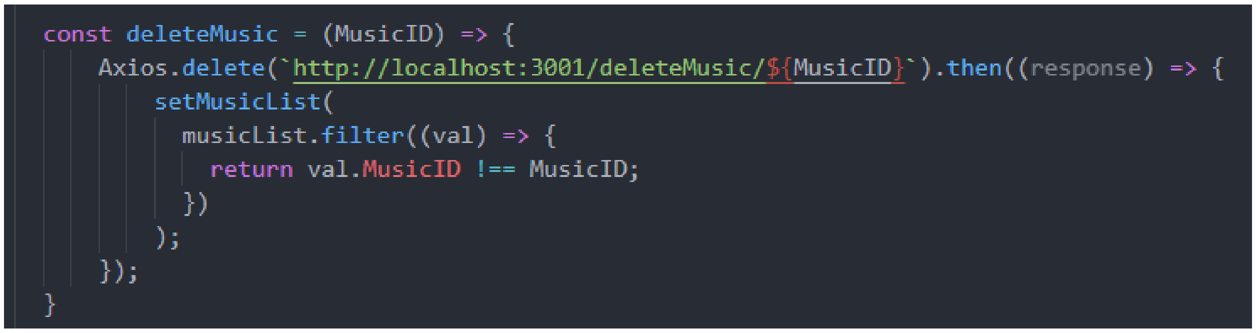
In the update user function, it will use axios.put to update the new user's data that the admin chooses to update, and it will put to "localhost:3001" to update the database.



```
app.put("/updateUsername", (req, res) => {
    const id = req.body.id;
    const Username = req.body.Username;
    data.query("UPDATE users SET Username = ? WHERE UserID = ?",[Username, id], (err, result) => {
        if (err) {
          console.log(err);
        } else {
          res.send(result);
        }
      }
    );
});

app.put("/updatePassword", (req, res) => {
    const id = req.body.id;
    const Passwords = req.body.Passwords;
    data.query("UPDATE users SET Passwords = ? WHERE UserID = ?",[Passwords, id], (err, result) =>
        if (err) {
          console.log(err);
        } else {
          res.send(result);
        }
      }
    );
});

app.put("/updateRole", (req, res) => {
    const id = req.body.id;
    const UserRole = req.body.UserRole;
    data.query("UPDATE users SET UserRole = ? WHERE UserID = ?",[UserRole, id], (err, result) => {
        if (err) {
          console.log(err);
        } else {
          res.send(result);
        }
      }
    );
});
```

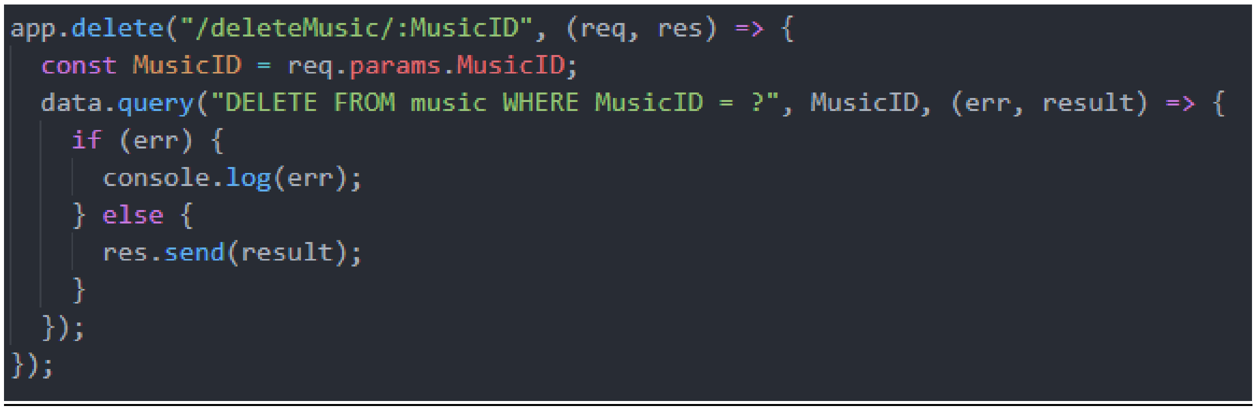
When the admin updates a user's data to the database it will use "app.put" to update it in the database.

Delete product



```
const deleteMusic = (MusicID) => {
    Axios.delete(`http://localhost:3001/deleteMusic/${MusicID}`).then((response) => {
        setMusicList(
          musicList.filter((val) => {
            return val.MusicID !== MusicID;
          })
        );
    });
}
```

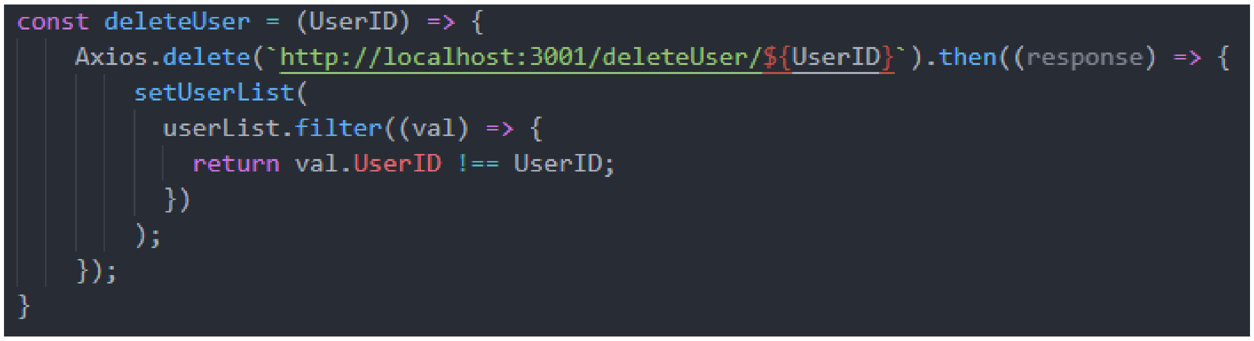
This function is made to fetch music data in port 3001 to set the MusicID that the admin would like to delete.



```
app.delete("/deleteMusic/:MusicID", (req, res) => {
  const MusicID = req.params.MusicID;
  data.query("DELETE FROM music WHERE MusicID = ?", MusicID, (err, result) => {
    if (err) {
      console.log(err);
    } else {
      res.send(result);
    }
  });
});
```

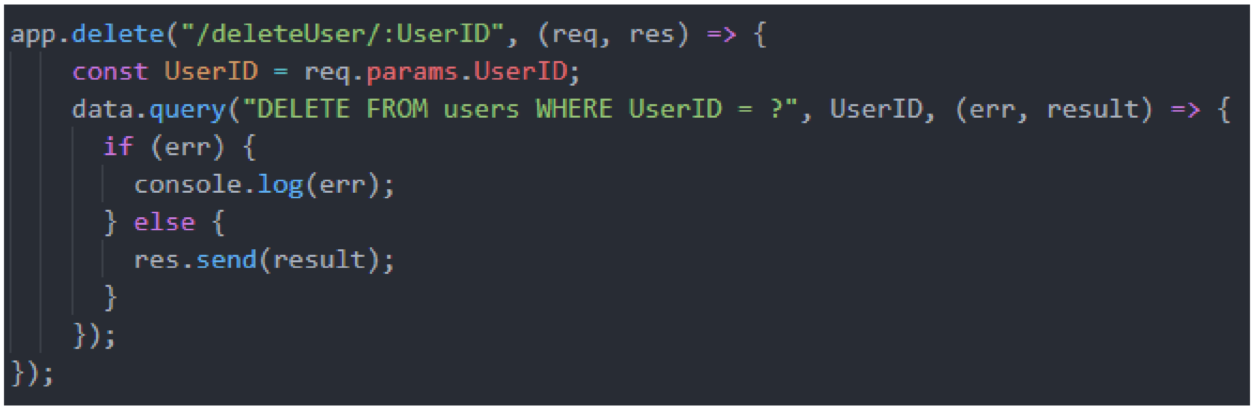
And when the admin chooses the music to delete, it will request to “app.delete” and delete the music in the database by using MusicID as a locator.

Delete user



```
const deleteUser = (UserID) => {
    Axios.delete(`http://localhost:3001/deleteUser/${UserID}`).then((response) => {
        setUserList(
          userList.filter((val) => {
            return val.UserID !== UserID;
          })
        );
    });
}
```

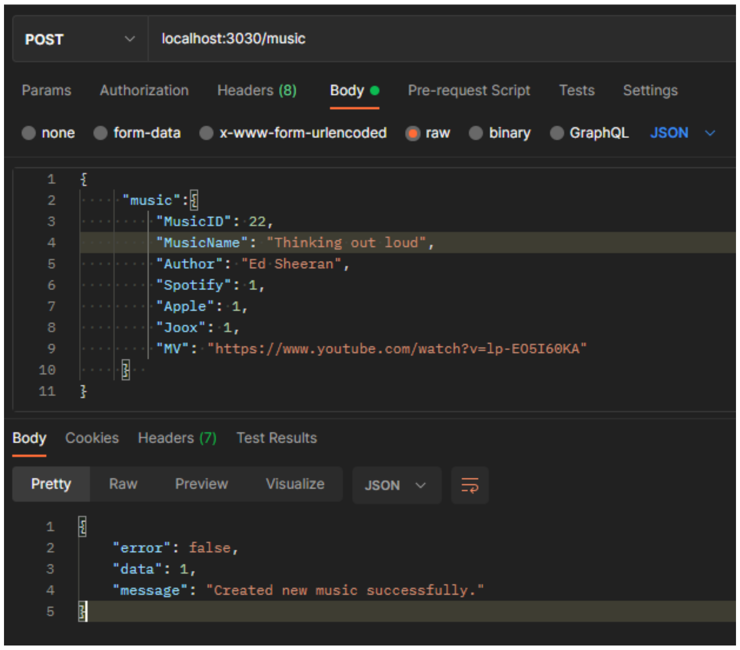
This function is made to fetch user data in port 3001 to set the UserID that the admin would like to delete.



```
app.delete("/deleteUser/:UserID", (req, res) => {
    const UserID = req.params.UserID;
    data.query("DELETE FROM users WHERE UserID = ?", UserID, (err, result) => {
      if (err) {
        console.log(err);
      } else {
        res.send(result);
      }
    });
});
```

And when the admin chooses the user to delete, it will request to “app.delete” and delete the user in the database by using UserID as a locator.

Postman Tester (Music Part)

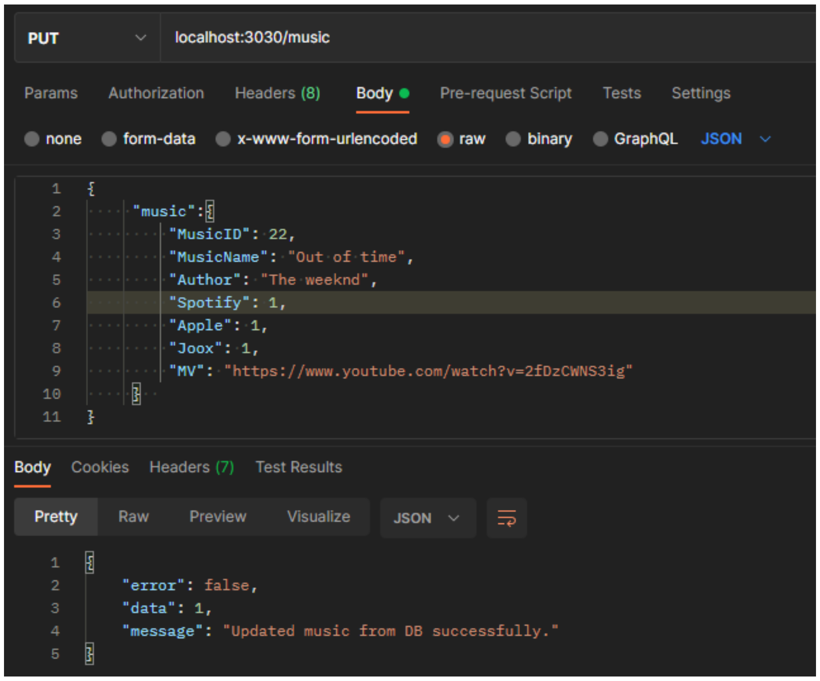


localhost:3030/music

```
{
    "music":{
        "MusicID": 22,
        "MusicName": "Thinking out loud",
        "Author": "Ed Sheeran",
        "Spotify": 1,
        "Apple": 1,
        "Joox": 1,
        "MV": "https://www.youtube.com/watch?v=lp-EO5I60KA"
    }
}
```

```
{
    "error": false,
    "data": 1,
    "message": "Created new music successfully."
}
```

POST: This is a post method for music service of our website that administrators can be able to add music information to the database.

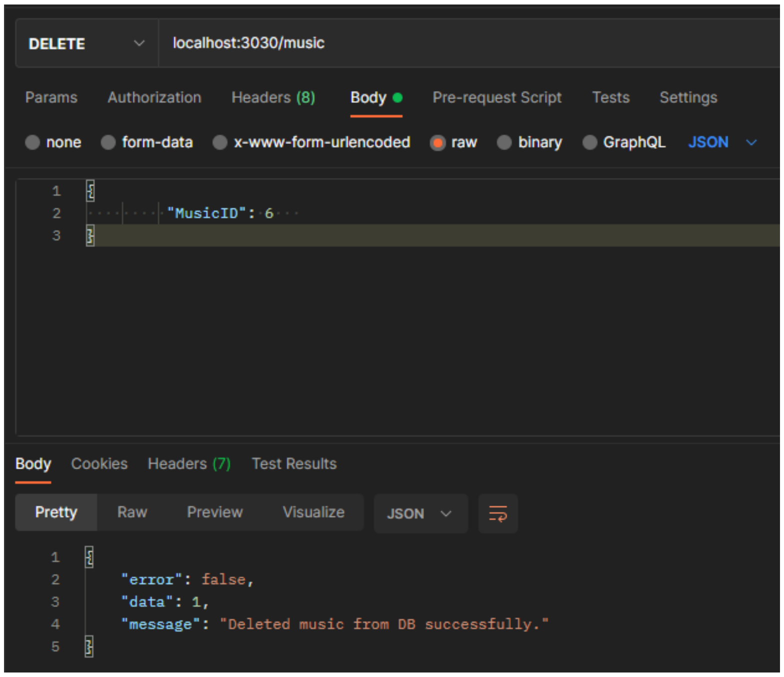


localhost:3030/music

```
{
    "music":{
        "MusicID": 22,
        "MusicName": "Out of time",
        "Author": "The weeknd",
        "Spotify": 1,
        "Apple": 1,
        "Joox": 1,
        "MV": "https://www.youtube.com/watch?v=2fDzCWNS3ig"
    }
}
```

```
{
    "error": false,
    "data": 1,
    "message": "Updated music from DB successfully."
}
```

PUT: This is an put method for the music service of our website that administrators can be able to update music information from the database.


DELETE localhost:3030/music

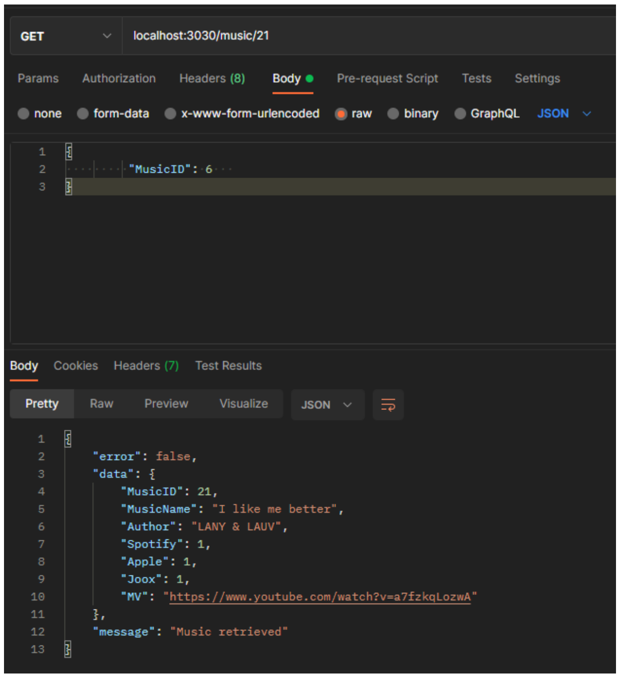
Body: raw JSON

```
{
    "MusicID": 6
}
```

Response:

```
{
    "error": false,
    "data": 1,
    "message": "Deleted music from DB successfully."
}
```

DELETE: This is a delete method for the music service of our website that administrators can be able to delete music information from the database.

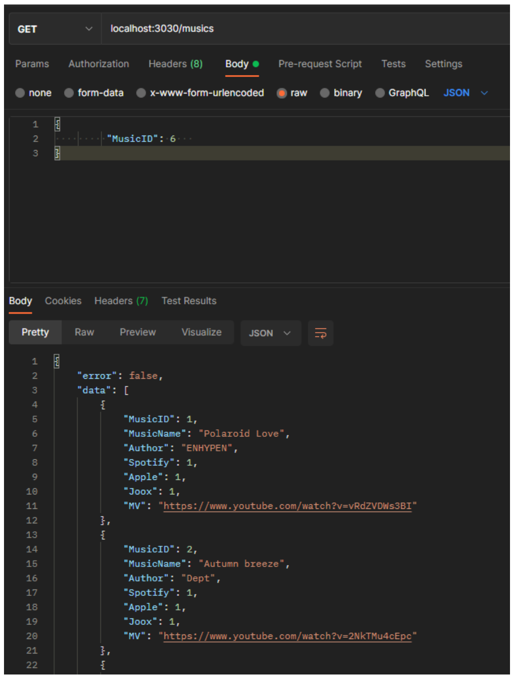

GET localhost:3030/music/21

Body: raw JSON

```
{
    "MusicID": 6
}
```

Response:

```
{
    "error": false,
    "data": {
        "MusicID": 21,
        "MusicName": "I like me better",
        "Author": "LANY & LAUV",
        "Spotify": 1,
        "Apple": 1,
        "Joox": 1,
        "MV": "https://www.youtube.com/watch?v=a7fzkqLozwA"
    },
    "message": "Music retrieved"
}
```


GET localhost:3030/musics

Body: raw JSON

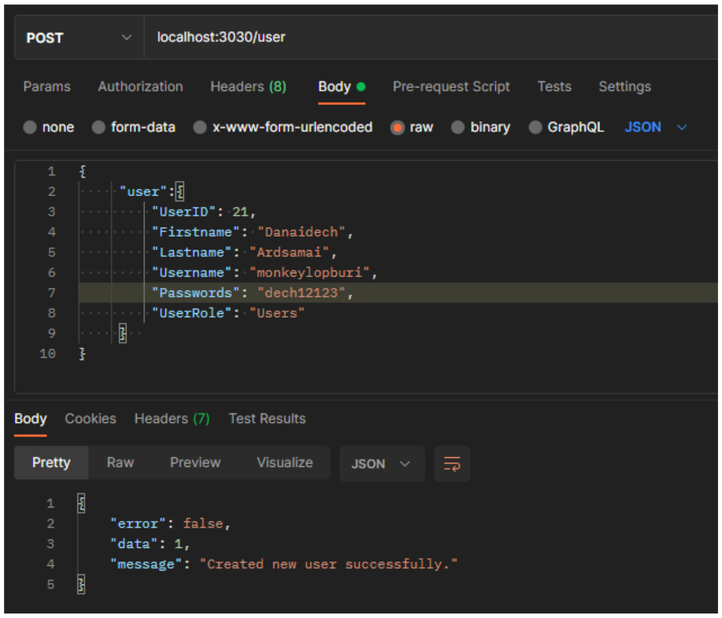
```
{
    "MusicID": 6
}
```

Response:

```
{
    "error": false,
    "data": [
        {
            "MusicID": 1,
            "MusicName": "Polaroid Love",
            "Author": "ENHYPEN",
            "Spotify": 1,
            "Apple": 1,
            "Joox": 1,
            "MV": "https://www.youtube.com/watch?v=vRdZVDWs3BI"
        },
        {
            "MusicID": 2,
            "MusicName": "Autumn breeze",
            "Author": "Dept",
            "Spotify": 1,
            "Apple": 1,
            "Joox": 1,
            "MV": "https://www.youtube.com/watch?v=2NkTMu4cEpc"
        },
        {
```

GET: This is a get method for music service of our website that administrators can be able to view music information from the database.

Postman Tester (User part)



POST localhost:3030/user

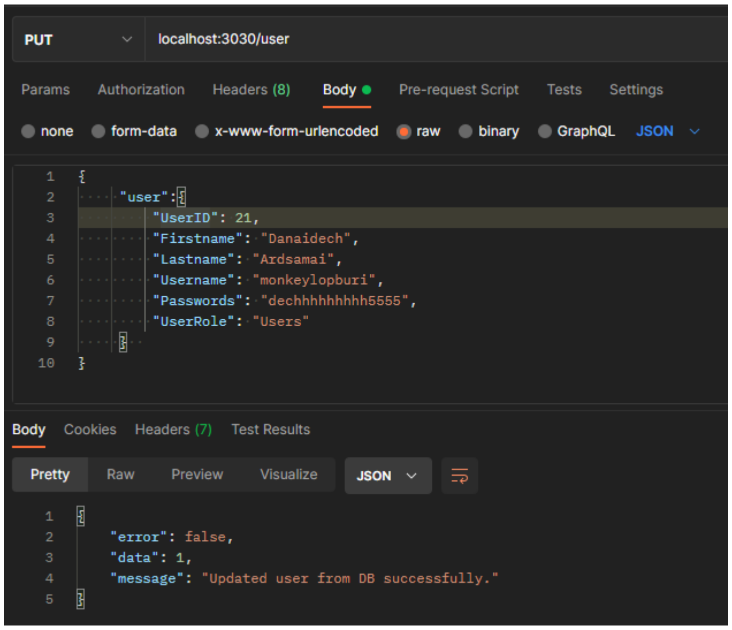
raw JSON

```
{
    "user":{
        "UserID": 21,
        "Firstname": "Danaidech",
        "Lastname": "Ardsamai",
        "Username": "monkeylopburi",
        "Passwords": "dech12123",
        "UserRole": "Users"
    }
}
```

Body JSON

```
{
    "error": false,
    "data": 1,
    "message": "Created new user successfully."
}
```

POST: This is a post method for music service of our website that administrators can be able to add user information to the database.



PUT localhost:3030/user

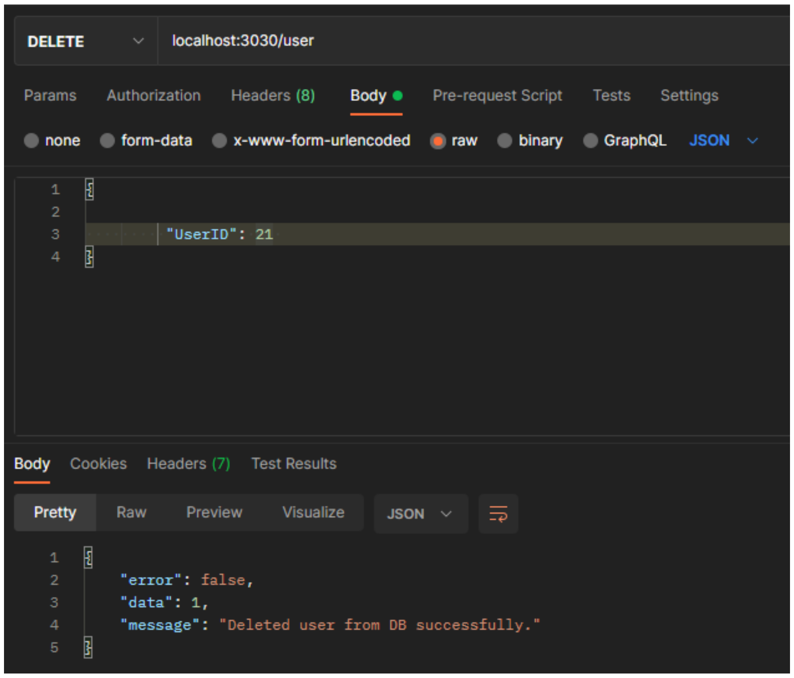
raw JSON

```
{
    "user":{
        "UserID": 21,
        "Firstname": "Danaidech",
        "Lastname": "Ardsamai",
        "Username": "monkeylopburi",
        "Passwords": "dechhhhhhhhh5555",
        "UserRole": "Users"
    }
}
```

Body JSON

```
{
    "error": false,
    "data": 1,
    "message": "Updated user from DB successfully."
}
```

PUT: This is an put method for the music service of our website that administrators can be able to update user information from the database.



```
DELETE  localhost:3030/user
```

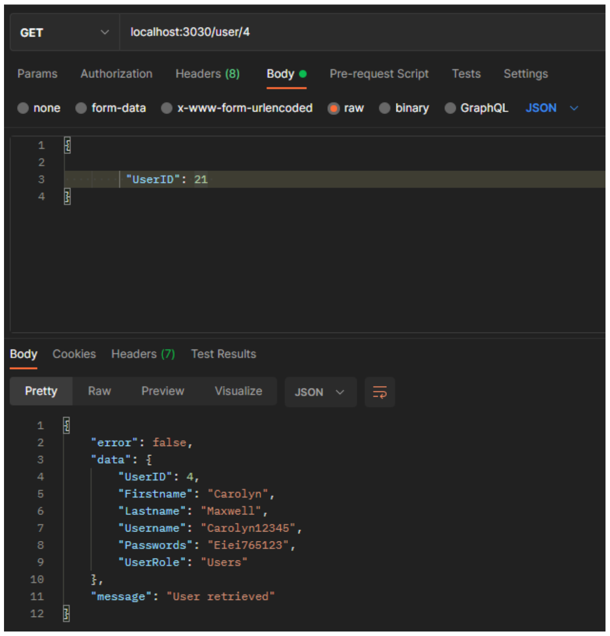
Body: raw, JSON

```
{

        "UserID": 21
}
```

Response Body (JSON):

```
{
    "error": false,
    "data": 1,
    "message": "Deleted user from DB successfully."
}
```

DELETE: This is a delete method for the music service of our website that administrators can delete user information from the database.



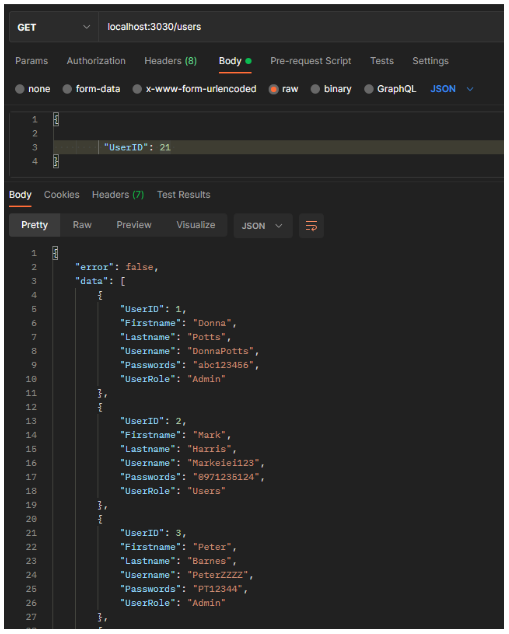
```
GET  localhost:3030/user/4
```

Body: raw, JSON

```
{

        "UserID": 21
}
```

Response Body (JSON):

```
{
    "error": false,
    "data": {
        "UserID": 4,
        "Firstname": "Carolyn",
        "Lastname": "Maxwell",
        "Username": "Carolyn12345",
        "Passwords": "Eiei765123",
        "UserRole": "Users"
    },
    "message": "User retrieved"
}
```



```
GET  localhost:3030/users
```

Body: raw, JSON

```
{

        "UserID": 21
}
```

Response Body (JSON):

```
{
    "error": false,
    "data": [
        {
            "UserID": 1,
            "Firstname": "Donna",
            "Lastname": "Potts",
            "Username": "DonnaPotts",
            "Passwords": "abc123456",
            "UserRole": "Admin"
        },
        {
            "UserID": 2,
            "Firstname": "Mark",
            "Lastname": "Harris",
            "Username": "Markeiei123",
            "Passwords": "0971235124",
            "UserRole": "Users"
        },
        {
            "UserID": 3,
            "Firstname": "Peter",
            "Lastname": "Barnes",
            "Username": "PeterZZZZ",
            "Passwords": "PT12344",
            "UserRole": "Admin"
        },
```

GET: This is a get method for music service of our website that administrators can be able to view user information from the database.