Web Application Detail and Code

Web App	Homepage	Method: GET
Detail	This code sets up a route that responds to GET requests at the root (/). It logs a message to the console, sends a status code of 200, and sends a file (home.html) as the response.	
Code	`router.get('/',(req,res) => {` `  console.log("Request at: /home")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/home.html`))` `})`	
Web App	Detail Page	Method: GET
Detail	This code sets up a route that response to GET request at the page of each album. It logs the message to the console, sends a status code of 200, and sends a file (detail.html) as the response.	
Code	`router.get('/album/:title',(req,res) =>{` `  var title = req.params.title` `  console.log(`Request at: /${title}`);` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/detail.html`))` `})`	
Web App	Search	Method: GET
Detail	This code sets up a route that responds to GET requests at search page, all user can use this page to search for album. It logs a message to the console, sends a status code of 200, and sends a file (search.html) as the response.	
Code	`router.get('/search',(req,res) => {` `  console.log("Request at: /search")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/search.html`))` `})`	

Web App	Searched album's detail	Method: GET
Detail	This code sets up a route that responds to GET requests at the detail of album which is looked for. It logs a message to the console, sends a status code of 200, and sends a file (detailSearch.html) as the response. The title parameter from the request URL is logged to the console.	
Code	`router.get('/search/:title',(req,res) => {` `  console.log("Request at: /"+req.params.title);` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/detailSearch.html`))` `})`	
Web App	searchDetail	Method: GET
Detail	This code sets up a route that responds to GET requests at /searchDetail. This page will show all products that match to the search categories. It logs a message to the console, sends a status code of 200, and sends a file (searchDetail.html) as the response.	
Code	`router.get('/searchDetail',(req,res) => {` `  console.log("Request at: /searchDetail");` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/searchDetail.html`))` `})`	
Web App	Login	Method: GET
Detail	This code sets up a route that responds to GET requests at /login. It logs a message to the console, sends a status code of 200, and sends a file (login.html) as the response.	
Code	`router.get('/login',(req,res) => {` `  console.log("Request at: /login")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/login.html`))` `})`	

Web App	AdminPortal	Method: GET

Detail	This code sets up a route that responds to GET requests at /adminPortal if admin input correct information. It logs a message to the console, sends a status code of 200, and sends a file (adminPortal.html) as the response.
Code	`router.get('/adminPortal',(req,res) => {` `  console.log("Request at: /adminPortal")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/adminPortal.html`))` `})` `})`
Web App	Account management — Method: GET
Detail	This code sets up a route that responds to GET requests at account management page, the page that admin can add new admin's info, update admin's info, and delete admin's info. It logs a message to the console, sends a status code of 200, and sends a file (accManagement.html) as the response.
Code	`router.get('/accManagement',(req,res) => {` `  console.log("Request at: /adminPortal")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/accManagement.html`))` `})`
Web App	Album management — Method: GET
Detail	This code sets up a route that responds to GET requests at /albumManagement. It logs a message to the console, sends a status code of 200, and sends a file (albumManagement.html) as the response.
Code	`router.get('/albumManagement',(req,res) => {` `  console.log("Request at: /albumManagement")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/albumManagement.html`))` `})`

Web App	Team	Method: GET

Detail	This code sets up a route that responds to GET requests at team page. It logs a message to the console, sends a status code of 200, and sends a file (team.html) as the response.
Code	`router.get('/team',(req,res) => {` `  console.log("Request at: /team")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/team.html`))` `})`
Web App	Admin's homepage — Method: GET
Detail	This code sets up a route that responds to GET requests at homepage of admin, which navigation bar will have button that can link admin to admin portal. It logs a message to the console, sends a status code of 200, and sends a file (adminhome.html) as the response.
Code	`router.get('/admin/home',(req,res) => {` `  console.log("Request at: /home")` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/adminhome.html`))` `})`
Web App	Admin search — Method: GET
Detail	This code sets up a route that responds to GET requests at search page for admin. It logs a message to the console, sends a status code of 200, and sends a file (adminsearch.html) as the response.
Code	`router.get('/admin/search',(req,res) => {` `  console.log("Request at: /admin/search");` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/adminsearch.html`))` `})`

Web App	Searched album's detail for admin	Method: GET

Detail	This code sets up a route that responds to GET requests at searched album's detail page for admin. It logs a message to the console, sends a status code of 200, and sends a file (admindetailSearch.html) as the response.
Code	`router.get('/admin/search/:title',(req,res) => {` `  console.log("Request at: /admin/"+req.params.title);` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/admindetailSearch.html`))` `})`
Web App	Admin search detail | Method: GET
Detail	This code sets up a route that responds to GET requests at the page that admin search for product It logs a message to the console, sends a status code of 200, and sends a file (adminsearchDetail.html) as the response.
Code	`router.get('/admin/searchDetail',(req,res) => {` `  console.log("Request at: /admin/searchDetail");` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/adminsearchDetail.html`))` `})`
Web App	Admin team | Method: GET
Detail	This code sets up a route that responds to GET requests at team page of admin. It logs a message to the console, sends a status code of 200, and sends a file (adminteam.html) as the response.
Code	`router.get('/admin/team',(req,res) => {` `  console.log("Request at: /admin/team");` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/adminteam.html`))` `})`

Web App	Admin album title	Method: GET

Detail	This code sets up a route that responds to GET requests at album's detail page for admin. Admin can delete either whole album or each songs in this page. It logs a message to the console, sends a status code of 200, and sends a file (admindetail.html) as the response	
Code	`router.get('/admin/album/:title',(req,res) => {` `  console.log("Request at: /admin/album/"+req.params.title);` `  res.statusCode = 200; // Status 200: OK` `  res.sendFile(path.join(`${__dirname}/HTML/admindetail.html`))` `})`	
Web App	error	Method: USE
Detail	This code sets up a middleware function that responds to all types of HTTP requests at any route. It logs a message to the console, sends a file (error.html) as the response, and sets the status code to 404. The status code 404 the server could not find the requested resource.	
Code	`router.use((req,res,next) => {` `  console.log("404: Invalid accesssed")` `  res.sendFile(path.join(`${__dirname}/HTML/error.html`))` `  res.statusCode = 404;` `  })`	

Web Service Detail and Code

Web Service	CallItems	Method: GET
Detail	This code sets up a route that responds to GET requests at /callItems, executes a SQL query to select all records from the Album table, and sends the results as the response. If there's an error executing the query, it will be thrown.	
Code	`router.get('/callItems',(req,res) =>{` `  let album = \`SELECT * FROM Album\`;` `  connection.query( album, function (error, results) {` `  if (error) throw error;` `  return res.send(results)` `  })` `  })`	
Web Service	Call all albums' name	Method: GET
Detail	This code sets up a route that responds to GET requests at /callAlbum, executes a SQL query to select all albums' name, and sends the results as the response. If there's an error executing the query, it will be thrown.	
Code	`router.get('/callAlbum',(req,res) => {` `  connection.query('SELECT Title From album', function (error, results){` `  if (error) throw error;` `  return res.send(results);` `  });` `})`	
Web Service	Call detail of each album	Method: GET
Detail	This code sets up a route that responds to GET requests at /callDetail/:title, replaces all "%20" with a space in the title, executes a SQL query to select all records from the Song table that match the title, and sends the results as the response. If there's an error executing the query, it will be thrown.	
Code	`router.get('/callDetail/:title',(req,res) =>{` `  let title = req.params.title;` `  title = title.replaceAll("%20"," ");`	

<table>
<tr><td></td><td colspan="2">

```
        connection.query('SELECT * FROM Song s INNER JOIN Album a ON s.aTitle = a.Title WHERE a.Title =
?', title, function (error, results) {
            if (error) throw error;
            return res.send(results);
        });
    })
})
```

</td></tr>
<tr><td>Web Service</td><td>Call all admins’ info</td><td>Method: GET</td></tr>
<tr><td>Detail</td><td colspan="2">This code sets up a route that responds to GET requests at /callAdmin, executes a SQL query to select all records from the _admin table and sends the results as the response. If there’s an error executing the query, it will be thrown.</td></tr>
<tr><td>Code</td><td colspan="2">

```
router.get('/callAdmin',(req,res) =>{
        let query = `SELECT * FROM _admin`;
        connection.query( query ,function (error, results) {
            if (error) throw error;
            return res.send(results)
        })
    })
```

</td></tr>
<tr><td>Web Service</td><td>Call specific admin’s info</td><td>Method: GET</td></tr>
<tr><td>Detail</td><td colspan="2">This code sets up a route that responds to GET requests at /callAcc, executes a SQL query to select all records from the _admin table that match the query of request, and sends the results as the response. If there’s an error executing the query, it will be thrown.</td></tr>
<tr><td>Code</td><td colspan="2">

```
router.get('/callAcc',(req,res) =>{
        let user = req.query.Username
        connection.query( `SELECT * FROM _admin WHERE Username = ?`, user, function (error, results) {
            if (error) throw error;
            return res.send(results)
        })
    })
```

</td></tr>
<tr><td>Web Service</td><td>Search for album</td><td>Method: GET</td></tr>
</table>

Detail	This code sets up a route that responds to GET requests at /callSearch, retrieves title, artist, year, type, lable, price, and the way to sort from the request, constructs a SQL query, executes the query, and sends the results as the response. If there's an error executing the query, it will be thrown.
Code	

```
router.get('/callSearch',(req,res) =>{
        console.log(req.query);
        let title = req.query.title;
        let artist = req.query.artist;
        let year = req.query.year;
        let type = req.query.eplp;
        let label = req.query.label;
        let lowprice = req.query.lowPrice;
        let highprice = req.query.highPrice;
        let sort = req.query.sort;
        let _query = `SELECT * FROM album WHERE 1=1`;

        if(title) _query += ` AND Title LIKE "%${title}%"`;
        if(artist) _query += ` AND contributeArtist LIKE "%${artist}%"`;
        if(year) _query += ` AND releaseYear = ${year}`;
        if(type) _query += ` AND albumType = "${type}"`;
        if(label) _query += ` AND label LIKE "%${label}%"`;
        if(lowprice && highprice){
            _query += ` AND Price BETWEEN ${lowprice} AND ${highprice}`;
        }else{
            if(lowprice) _query += ` AND Price >= ${lowprice}`;
            if(highprice) _query += ` AND Price <= ${highprice}`;
        }
        if(sort){
            switch (sort) {
                case "A2Z":
                    _query += ` ORDER BY Title ASC`
                    break;

                case "Z2A":
                    _query += ` ORDER BY Title DESC`
                    break;
```

	`` case "PriceASC": _query += ` ORDER BY Price ASC` break; case "PriceDESC": _query += ` ORDER BY Price DESC` break; } } console.log(_query) connection.query(_query, function (error, results) { if (error) throw error; return res.send(results) }) }) ```
Web Service	Add new admin — Method: POST
Detail	This code sets up a route that responds to POST requests at /addAdmin, retrieves the Admin object from the request body, constructs a SQL query to insert a new admin into the _admin table using the values from the Admin object, executes the query, and sends a success message as the response. If there's an error executing the query, it will be thrown.
Code	``` router.post('/addAdmin',(req,res) => { let value = req.body.Admin; let admin = req.body.Admin.Fname; connection.query(`INSERT INTO _admin VALUE ("${value.Fname}","${value.Lname}","${value._Username}","${value.Password}")`,function (error, results){ if (error) throw error; return res.send({error: false, data: results.affectedRows, message: `Admin ${admin} has been added on our system successfully`}) }); }) ```
Web Service	Add new album — Method: POST

Detail	This code listens for a POST request at the /addAlbum endpoint. When such a request is received, it reads the album object from the request body. It then runs a SQL query to insert the album object into the album table in the database. If there's an error during this process, it throws the error. If the insertion is successful, it sends back a response.	
Code	`router.post('/addAlbum',(req,res) => {` `let album = req.body.album;` `connection.query('INSERT INTO album SET ?', [album], function (error, results){` `if (error) throw error;` `return res.send({ error: false, data: results.affectedRows, message: \`Album ${album.Title} has been added successfully\`});` `});` `})`	
Web Service	Add new song to album	Method: POST
Detail	This code listens for a POST request at the /addSong endpoint. When such a request is received, it reads the song object from the request body. It then runs a SQL query to insert the song object into the song table in the database. If there's an error during this process, it throws the error. If the insertion is successful, it sends back a response	
Code	`router.post('/addSong',(req,res) => {` `let song = req.body.song;` `connection.query('INSERT INTO song SET ?', [song], function (error, results){` `if (error) throw error;` `return res.send({ error: false, data: results.affectedRows, message: \`Song ${song.songName} has been added successfully\`});` `});` `})`	
Web Service	Update existing admin	Method: PUT
Detail	This code listens for a PUT request at the /updateAdmin endpoint. When a request is received, it reads three values from the request body: the property to edit, the new value, and the username of the admin.	

	It then runs a SQL query to update the specified property of the admin with the given username in the _admin table in the database. If there's an error during this process, it throws the error. If the update is successful, it sends back a response
Code	`router.put('/updateAdmin',(req,res) => {` `let property = req.body.Admin.Edit;` `let value = req.body.Admin.Value;` `let username = req.body.Admin.Username;` `connection.query(`UPDATE _admin SET ${property} = "${value}" WHERE Username = "${username}"`, function (error, results){` `if (error) throw error;` `return res.send({error: false, data: results.affectedRows, message: `${username}'s ${property} has become ${value}.`})` `});` `})`
Web Service	Update existing album — Method: PUT
Detail	This code listens for a PUT request at the /updateAlbum endpoint. When such a request is received, it reads three values from the request body: the property to edit, the new value, and the title of the album. These are expected to be under req.body.song.property, req.body.song.value, and req.body.song.title respectively. It then runs a SQL query to update the specified property of the album with the given title in the album table in the database. If there's an error during this process, it throws the error. If the update is successful, it sends back a response
Code	`router.put('/updateAlbum',(req,res) => {` `let property = req.body.song.property;` `let value = req.body.song.value;` `let album = req.body.song.title;` `connection.query(`UPDATE album SET ${property} = "${value}" WHERE Title = "${album}"`, function (error, results){` `if (error) throw error;` `return res.send({ error: false, data: results.affectedRows, message: `Album ${album}'s ${property} has become ${value}`});` `});`

	})	
Web Service	Update existing song	Method: PUT
Detail	This code listens for a PUT request at the /updateSong endpoint. When such a request is received, it reads four values from the request body: the name of the song, the property to edit, the new value, and the title of the album. These are expected to be under req.body.song.song, req.body.song.property, req.body.song.value, and req.body.song.title respectively. It then runs a SQL query to update the specified property of the song with the given name and album title in the song table in the database. If there's an error during this process, it throws the error. If the update is successful, it sends back a response	
Code	`router.put('/updateSong',(req,res) => {` `let songName = req.body.song.song;` `let property = req.body.song.property;` `let value = req.body.song.value;` `let title = req.body.song.title;` `connection.query(`UPDATE song SET ${property} = "${value}" WHERE songName = "${songName}" AND aTitle = "${title}"`, function (error, results){` `if (error) throw error;` `return res.send({ error: false, data: results.affectedRows, message: `Updated song`});` `});` `})`	
Web Service	Delete admin	Method: DELETE
Detail	This code listens for a DELETE request at the /deleteAdmin/:username endpoint. When such a request is received, it reads the username from the request parameters. It then runs a SQL query to delete the admin with the given username from the _admin table in the database. If there's an error during this process, it throws the error. If the deletion is successful, it sends back a response	
Code	`router.delete('/deleteAdmin/:username',(req,res) => {` `let username = req.params.username;` `connection.query('DELETE FROM _admin WHERE Username = ?', [username], function (error, results){` `if (error) throw error;`	

<table>
<tr><td></td><td colspan="2">

```
        return res.send({ error: false, data: results.affectedRows, message: `Admin ${username} has been deleted successfully.`});
        });
})
```

</td></tr>
<tr><td>Web Service</td><td>Delete album</td><td>Method: DELETE</td></tr>
<tr><td>Detail</td><td colspan="2">This code listens for a DELETE request at the /deleteAlbum/:title endpoint. When such a request is received, it reads the title from the request parameters. It then replaces any “%20” in the title with a space. It then runs a SQL query to delete the album with the given title from the album table in the database. If there’s an error during this process, it throws the error. If the deletion is successful, it sends back a response</td></tr>
<tr><td>Code</td><td colspan="2">

```
router.delete('/deleteAlbum/:title',(req,res) => {
        let Title = req.params.title;
        Title = Title.replaceAll("%20"," ");
        connection.query(`DELETE FROM album WHERE Title = "${Title}"`, function (error, results){
        if (error) throw error;
        return res.send({ error: false, data: results.affectedRows, message: Title});
        });
})
```

</td></tr>
<tr><td>Web Service</td><td>Delete song from album</td><td>Method: DELETE</td></tr>
<tr><td>Detail</td><td colspan="2">This code listens for a DELETE request at the /deleteSong endpoint.
When such a request is received, it reads the album and song from the request body. It then runs a SQL query to delete the song with the given name from the specified album in the song table in the database. If there’s an error during this process, it throws the error. If the deletion is successful, it sends back a response</td></tr>
<tr><td>Code</td><td colspan="2">

```
router.delete('/deleteSong',(req,res) => {
        let Album = req.body.album;
        let Song = req.body.song;
```

</td></tr>
</table>

	connection.query('DELETE FROM song WHERE aTitle = ? AND songName = ?', [Album,Song], function (error, results){ if (error) throw error; return res.send({ error: false, data: results.affectedRows, message: `Song ${Song} from album ${Album} has been deleted successfully.`}); }); })
Web Service	Error response — Method: USE
Detail	It logs the message "404: Invalid accessed" to the console. It then sends an HTML file named error.html located in the Frontend/HTML directory relative to the current file (__dirname refers to the directory of the current file). Finally, it sets the HTTP status code of the response to 404, which stands for "Not Found".
Code	router.use((req,res,next) => { console.log("404: Invalid accesssed") res.sendFile(path.join(`${__dirname}/../Frontend/HTML/error.html`)) res.statusCode = 404; })

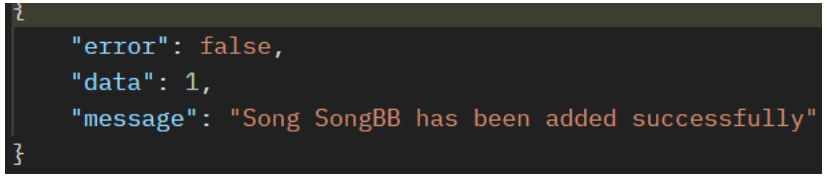
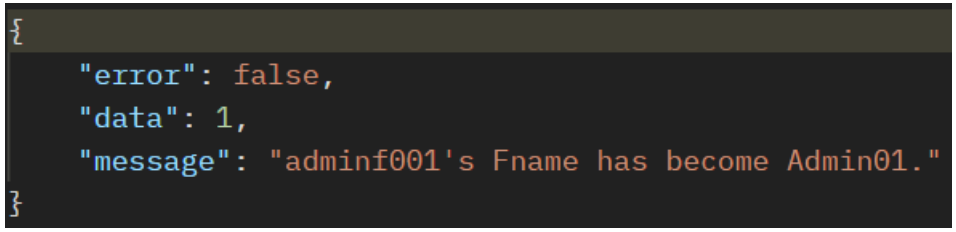
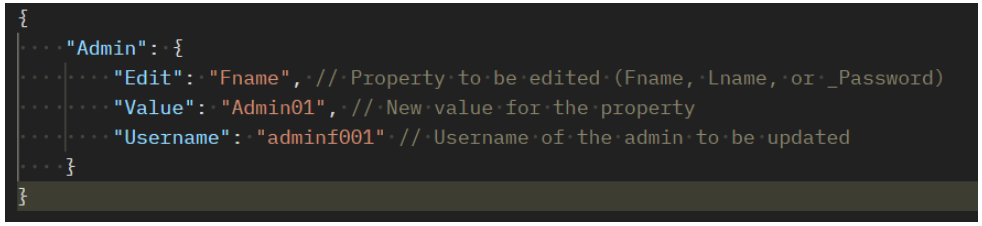
Postman Testing Results

Testing Result 1	"cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Album/polycat.jpg", "cassetteTape": 1, "vinylDisc": 0, "label": "Smallroom", "contributeArtist": "Polycat", "albumType": "LP", "releaseYear": 2016 "Title": "Are You Serious", "Price": 2400, "cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Album/Serious%20Bacon.jpg", "cassetteTape": 12, "vinylDisc": 20, "label": "Muzik Move Records", "contributeArtist": "Serious Bacon",
Body/Query	None
Request	GET:{http:localhost:3100/callItems}
Testing Result 2	"aTitle": "Are You Serious", "songName": "Phee-Phee-Tud-Wan-Hai-Noi", "Duration": "3:39", "_Key": "F#", "Title": "Are You Serious", "Price": 2400, "cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Album/Serious%20Bacon.jpg", "cassetteTape": 12, "vinylDisc": 20, "label": "Muzik Move Records", "contributeArtist": "Serious Bacon", "albumType": "LP", "releaseYear": 2022 "aTitle": "Are You Serious", "songName": "Mai-Yak-Fung", "Duration": "4:38", "_Key": "E", "Title": "Are You Serious", "Price": 2400, "cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Album/Serious%20Bacon.jpg", "cassetteTape": 12, "vinylDisc": 20, "label": "Muzik Move Records", "contributeArtist": "Serious Bacon", "albumType": "LP", "releaseYear": 2022
Parameter	Album title such as Are you serious
Request	GET:{http:localhost:3100/callDetail/Are You Serious}

Testing Result 3	{ "Fname": "Admin1", "Lname": "Fake", "Username": "adminf001", "_Password": "admin1" }, { "Fname": "Admin2", "Lname": "Fake", "Username": "adminf002", "_Password": "admin2" }, { "Fname": "Jidapa", "Lname": "Kraisangka", "Username": "Jidapa.kra", "_Password": "adminPa" },
Body/Query	None
Request	GET:{http://localhost/3100/callAdmin}
Testing Result 4	[{ "Fname": "Admin2", "Lname": "Fake", "Username": "adminf002", "_Password": "admin2" }]
Body/Query	Admin’s username such as adminf002
Request	GET:{http://localhost:3100/callAcc?Username=adminf002}

Testing Result 5	[{ "Title": "Reun Pae Volume 6", "Price": 1200, "cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Album/reun.png", "cassetteTape": 0, "vinylDisc": 5, "label": "Smallroom", "contributeArtist": "Tattoo Colour", "albumType": "LP", "releaseYear": 2022 }, { "Title": "B", "Price": 2200, "cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Album/B.png", "cassetteTape": 0, "vinylDisc": 48, "label": "Smallroom", "contributeArtist": "Slur", "albumType": "LP", "releaseYear": 2015
Body/Query	Search criteria such as label=smallroom, eplp=LP, and sort=PriceASC
Request	GET:{http://localhost:3100/callSearch?title=&artist=&year=&eplp=LP&label=Smallroom&lowPrice=&highPrice=&sort=PriceASC}
Testing Result 6	{ "Title": "80 Kisses" }, { "Title": "Are You Serious" }, { "Title": "B" }, { "Title": "BLISS" }, { "Title": "D Gerrard" }, { "Title": "ETC Studio Live Session" }, { "Title": "ICE Saranyu"
Body/Query	None

Request	GET:{http://localhost:3100/callAlbum}
Testing Result 7	{ "error": false, "data": 1, "message": "Admin Admin100 has been added on our system successfully" }
Body/Query	{ "Admin": { "Fname": "Admin100", "Lname": "Makesd", "Username": "adminkwer", "_Password": "admin0213354" } }
Request	POST: {http://localhost:3100/addAdmin}
Testing Result 8	{ "error": false, "data": 1, "message": "Album Sunny has been added successfully"
Body/Query	{ "album":{ "Title":"Sunny", "Price":1000, "cover": "https://raw.githubusercontent.com/SunnyRichman/ICTLife/FESTAProject/Team%20members/Sunny.jpg", "cassetteTape": 15, "vinylDisc": 10, "label": "ICT", "contributeArtist":"Sunny Richman", "albumType": "LP", "releaseYear": "2002" } }
Request	POST:{http://localhost:3100/addAlbum}

Testing Result 9	
Body/Query	
Request	POST:{http://localhost:3100/addSong}
Testing Result 10	
Body/Query	
Request	PUT:{ http://localhost:3100/updateAdmin }

Testing Result 11	{ "error": false, "data": 1, "message": "Album Sunny's vinylDisc has become 10" }
Body/Query	{ "song":{ "title":"Sunny", "value":"10", "property":"vinylDisc" } }
Request	PUT:{ http://localhost:3100/updateAlbum }
Testing Result 12	{ "error": false, "data": 1, "message": "Updated song" }
Body/Query	{ "song": { "property":"Duration", "song":"Insomnia", "title": "BLISS", "value": "4:52" } }
Request	PUT:{ http://localhost:3100/updateSong }

Testing Result 13	{ "error": false, "data": 1, "message": "Admin adminf001 has been deleted successfully." }
Parameter	Admin's username such as adminf001
Request	DELETE:{ http://localhost:3100/deleteAdmin/adminf001}
Testing Result 14	{ "error": false, "data": 1, "message": "Sunny" }
Parameter	Album title such as Sunny
Request	DELETE:{ http://localhost:3100/deleteAlbum/Sunny}
Testing Result 15	{ "error": false, "data": 1, "message": "Song SongBB from album B has been deleted successfully." }
Body/Query	{ "album": "B", "song": "SongBB" }
Request	DELETE:{ http://localhost:3100/deleteSong}