A

PROJECT REPORT ON

ONLINE MUSIC SYSTEM (BEATZZ)

By

GOPAL MALAVIYA (CE-75) (19CEUES049)
YASH PIPARIYA (CE-128) (19CEUEG033)
DARSH VAGHELA (CE-172) (19CEUBS089)

B.Tech CE Semester-VI
Subject: System Design Practice

Guided by:
Prof.Pandav K. Patel
Assistant Professor
Dept. of Comp. Engg.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of

System Design Practice and recorded in this journal is the

bonafide work of

GOPAL MALAVIYA (CE-75) (19CEUES049)
YASH PIPARIYA (CE-128) (19CEUEG033)
DARSH VAGHELA (CE-172) (19CEUBS089)

of B.Tech semester **VI** in the branch of **Computer Engineering**

during the academic year **2021-2022**.

Prof. Pandav K. Patel	Dr. C. K. Bhensdadia,
Assistant Professor,	Head,
Dept. of Computer Engg.,	Dept. of Computer Engg.,
Faculty of Technology	Faculty of Technology
Dharmsinh Desai University, Nadiad	Dharmsinh Desai University, Nadiad

Contents:

Abstract:

Beatzz is a collection of MP3 songs of different languages in one place. Where users can listen songs and make playlists of their favourite song. User need to login for access playlists.

Beatzz provides 24 hours customer service, and decrease the manual efforts and time. By giving facility of search user can find any songs by song name or artist name.

Introduction:

Our project is about ONLINE MUSIC SYSTEM. In this system user can listen song and search songs which she/he likes. User can also create their own playlists , edit playlists and delete playlist. for that, user have to must login. In this system there are mainly two type of user i) Admin ii) User.

Admin:

Admins have to first login with their username and password. After login admin can add songs , edit songs, delete songs. All user can see these songs , play songs which are added by admin . Admin can also create playlists, edit playlists, delete playlists . In this Playlist admin have to add some songs which is already added by them and also they can edit songs , delete songs from playlist.

User:

Users have to first login with their username and password. After login user can create their own playlists , edit playlists , delete playlists . After creation of playlist user have to add some songs in that playlist and also she/he can edit songs ,delete songs from that playlist. If user just want to listen song then there is no need to login.

Technology:

- React js
- Node js
- Express js
- Javascript

Platform:

- MongoDB
- Firebase
- HTML
- CSS
- Bootstrap

Tools:

- Visual Studio Code
- MongoDB Compass

Software Requirement Specification:

R.1 Manage Users

R.1.1 Registration

Description: User register themselves into the system by entering their required details.

Input: User details like name, email-id, password.

Processing: New account is created for the user and unique id is given to user to login.

Output: User gets id and password.

R.1.2 Login

Description: User login to the system by entering valid id and password.

Input: User's id and password.

Processing: Id and password are verified, if they are valid then access is given otherwise access is denied.

Output: User logged in to the system and home page opens.

R.2 Manage Songs

R.2.1 Add song

Description: Only admin can manage songs.

Input: Details of song to be added

Output: Item is added

R.2.2 Update song

Description: Only admin can manage songs.

Input: Song to be updated.

Output: Song is updated.

R.2.3 Delete song

Description: Only admin can manage songs.

Input: Select song .

Output: Song is deleted.

R.2.4 Play Song

Description: User can click on any song and song will play.

Input: Select song

Output: Song will paly

R.2.5 Search Song

Description: User can search their favourite song they interested by Song name or artist name and songs which are matched will be display.

Input: Query for searching song or artist.

Output: Song which are matched with query will display.

R.3 Manage Playlists

Description: Admin and user both can manage playlists but playlists which are created by admin visible to all user and admin & playlist created by user will visible only to that user.

R.3.1 Create playlist

Input: Details of playlist and songs to be added

Output: playlist is created

R.3.2 Update playlist

Input: playlist and songs to be updated.

Output: playlist and songs is updated.

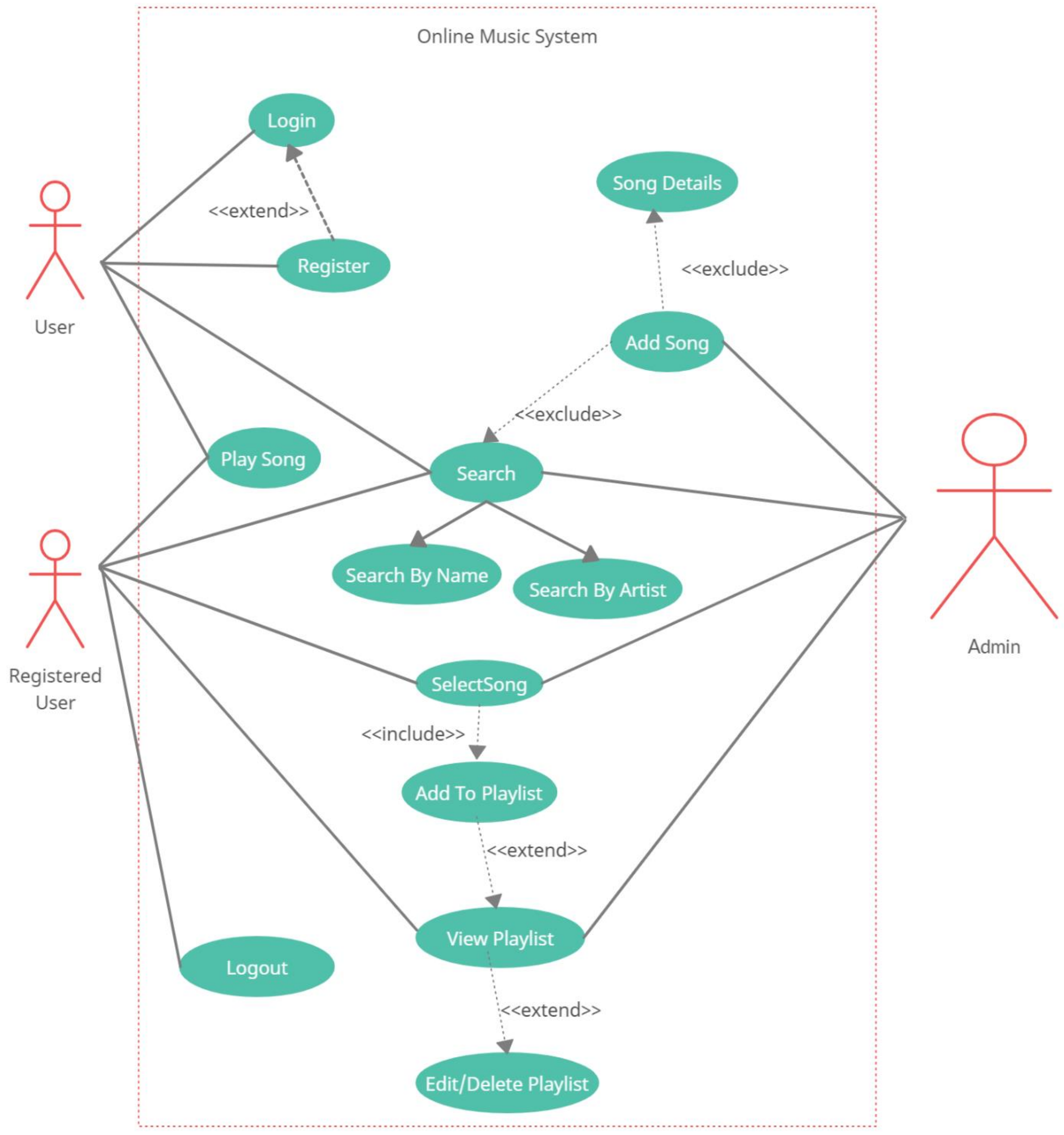
R.3.3 Delete playlist

Input: Select playlist.

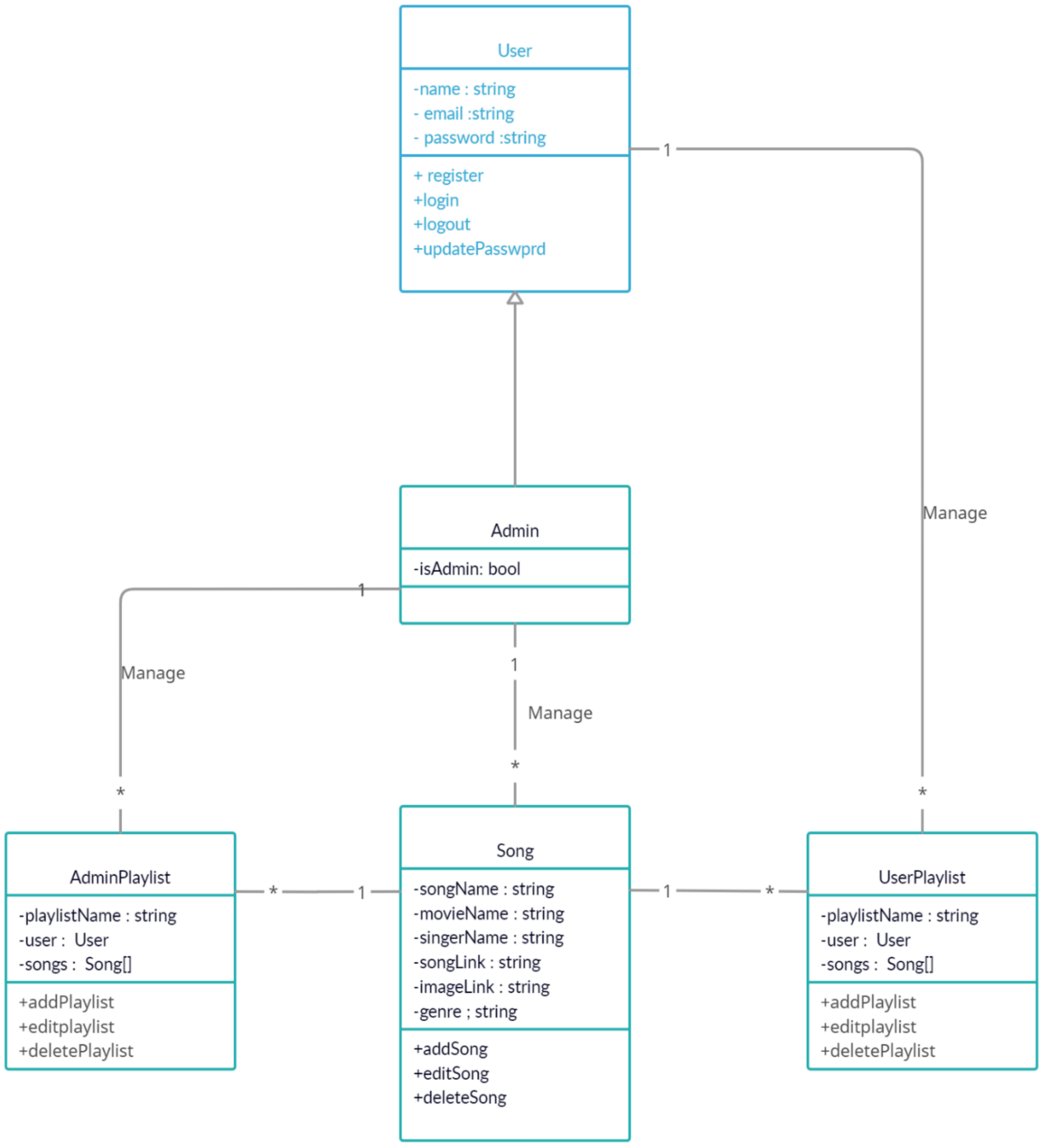
Output: playlist is deleted.

Design:

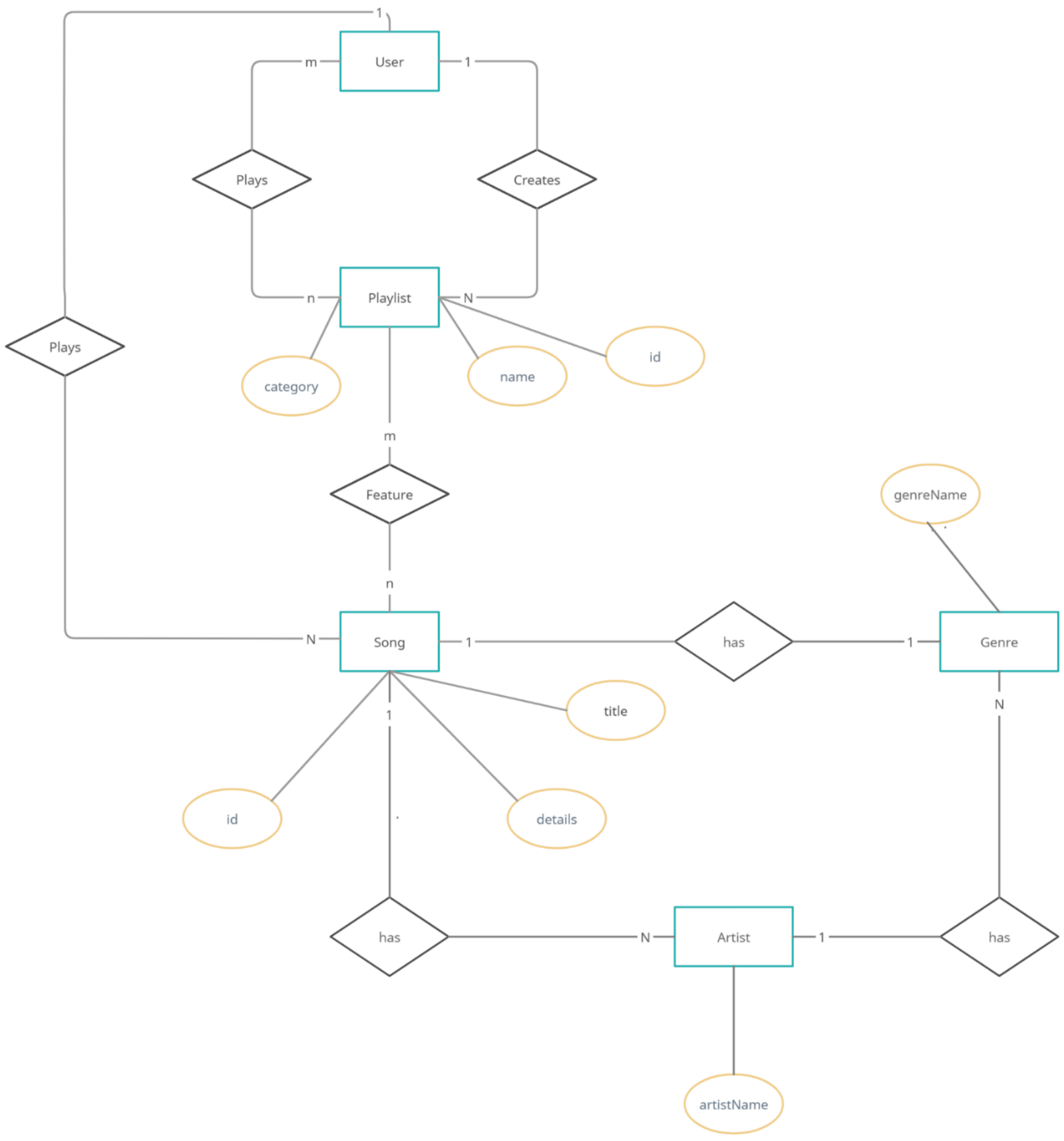
I) Use case Diagram:



II) Class Diagram:

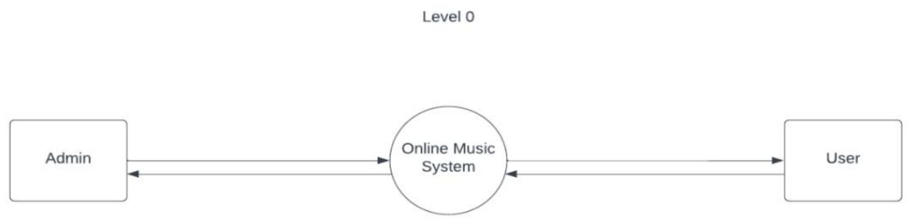


III) E-R Diagram:

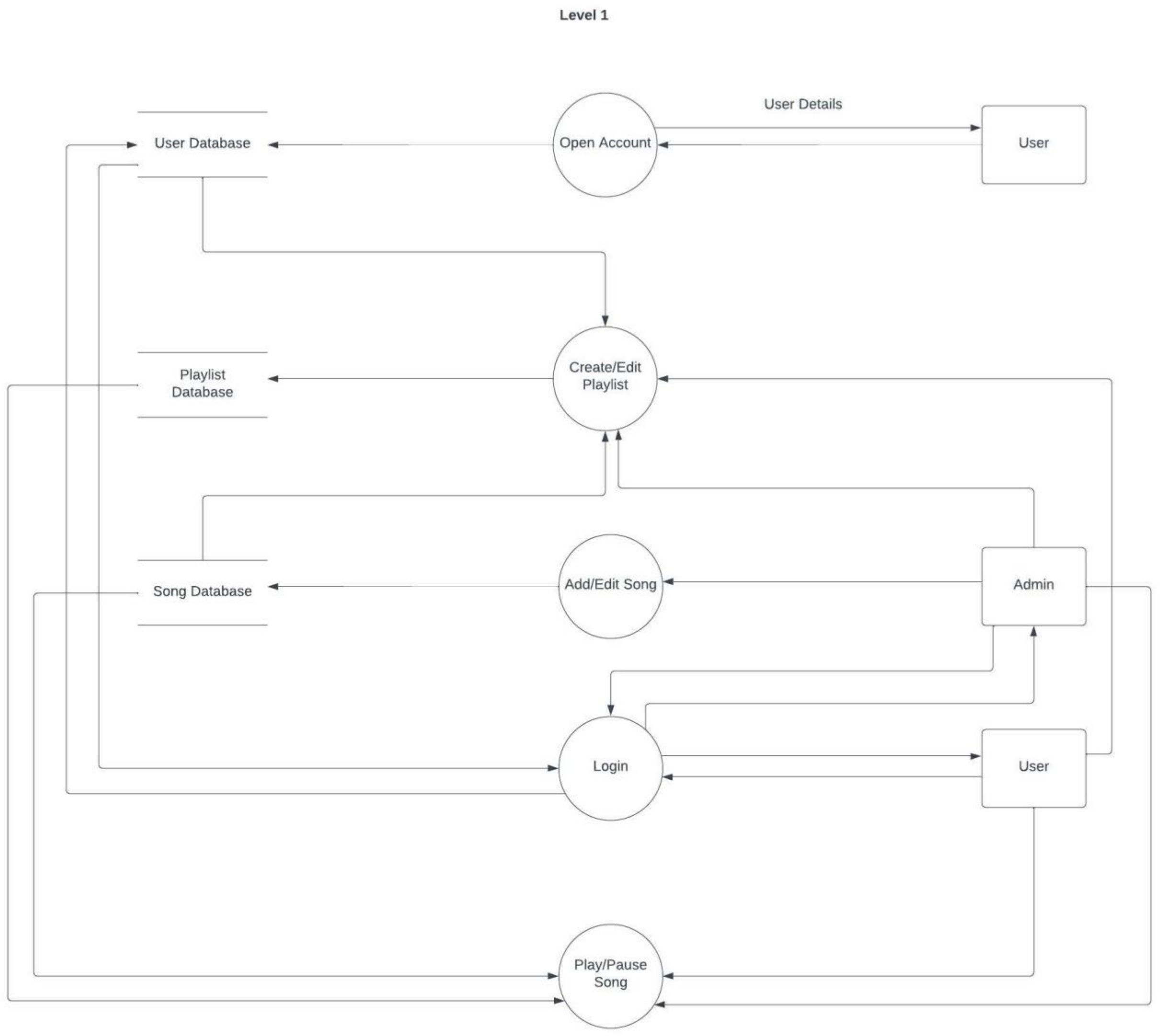


IV) DFD:

Level 0

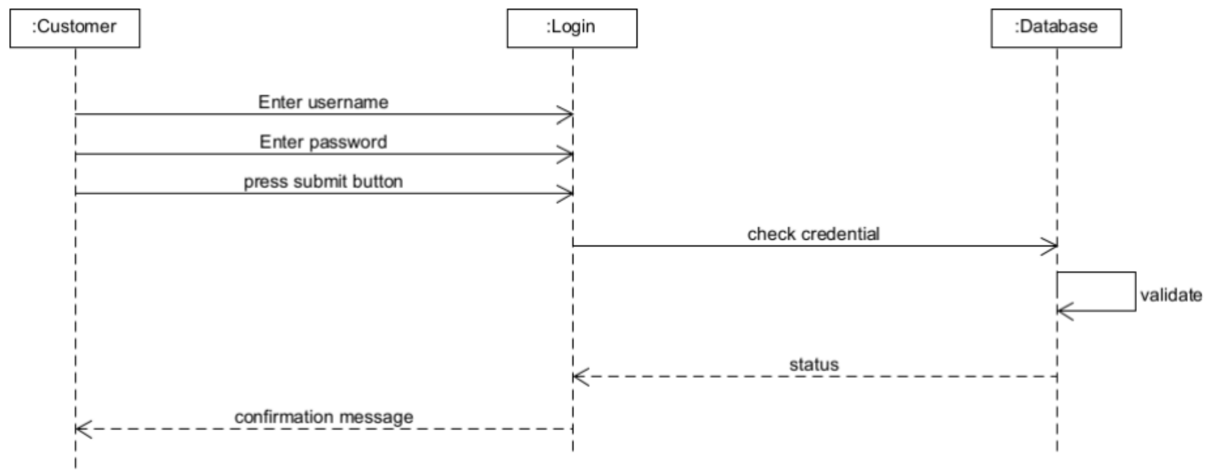


Level 1

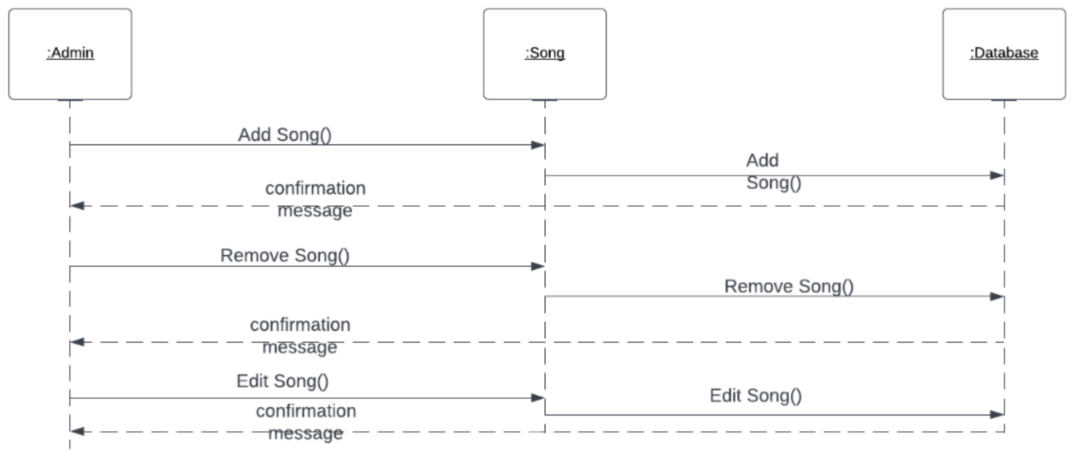


V)Sequence Diagram:

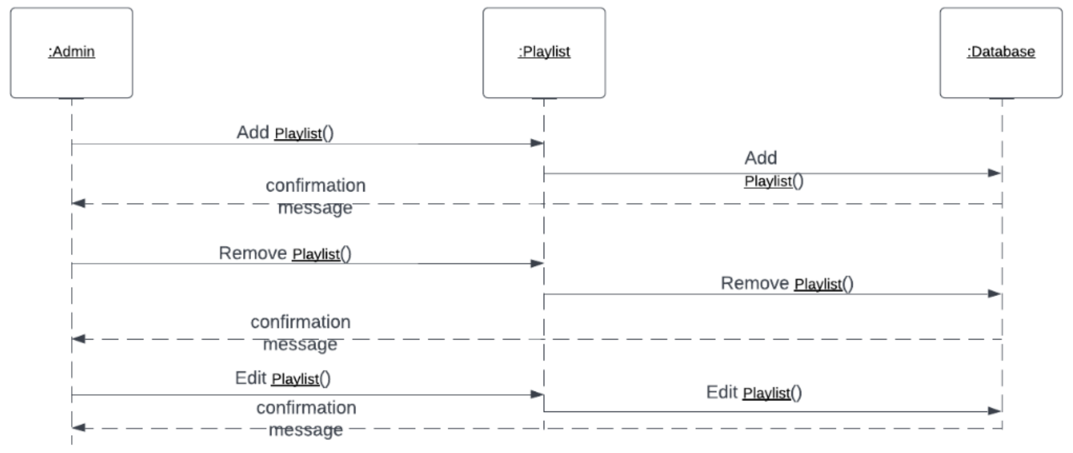
1)Login:



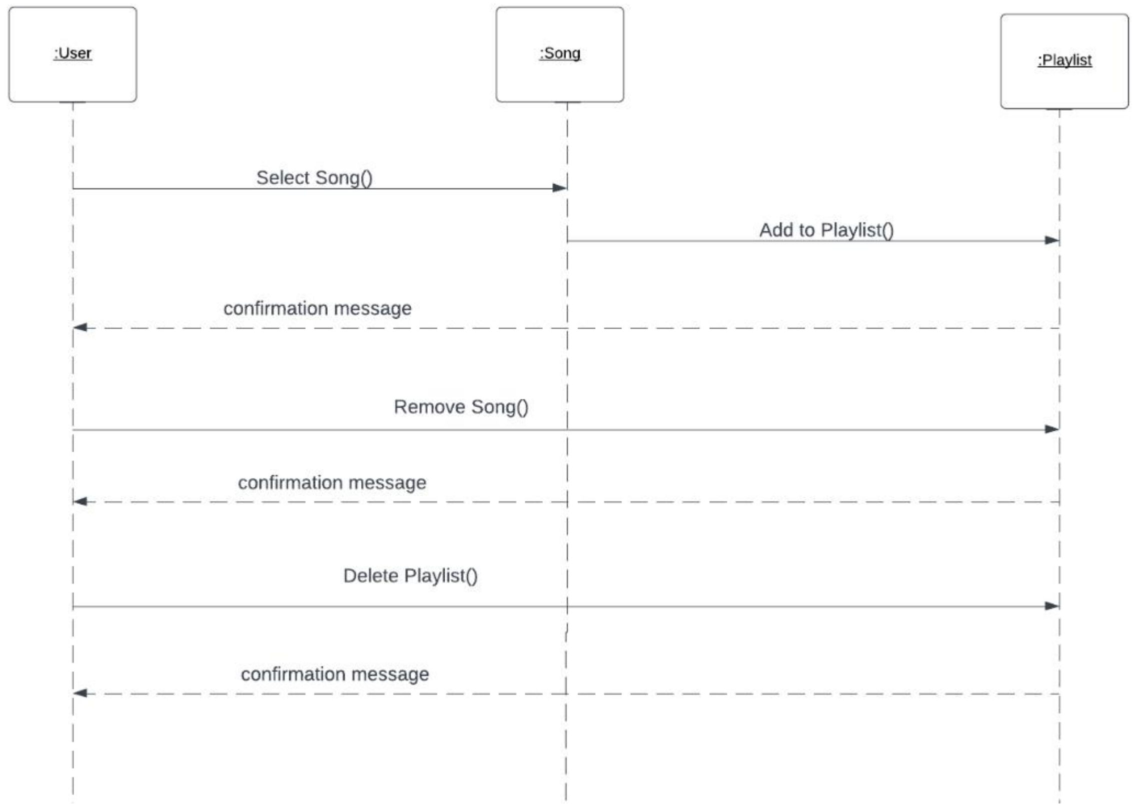
2)Manage Song By Admin:



3)Manage Playlist By Admin:

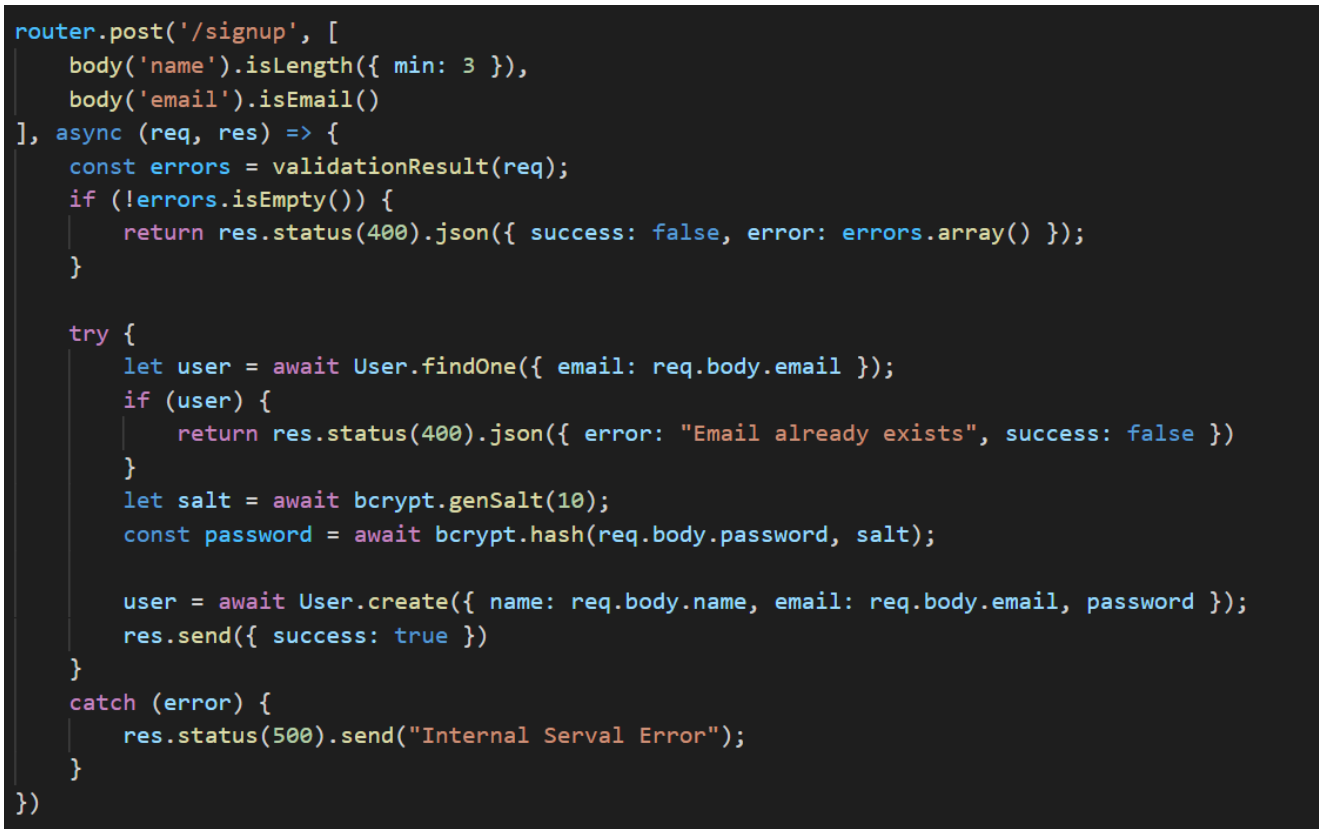


4)Manage Playlist By User:



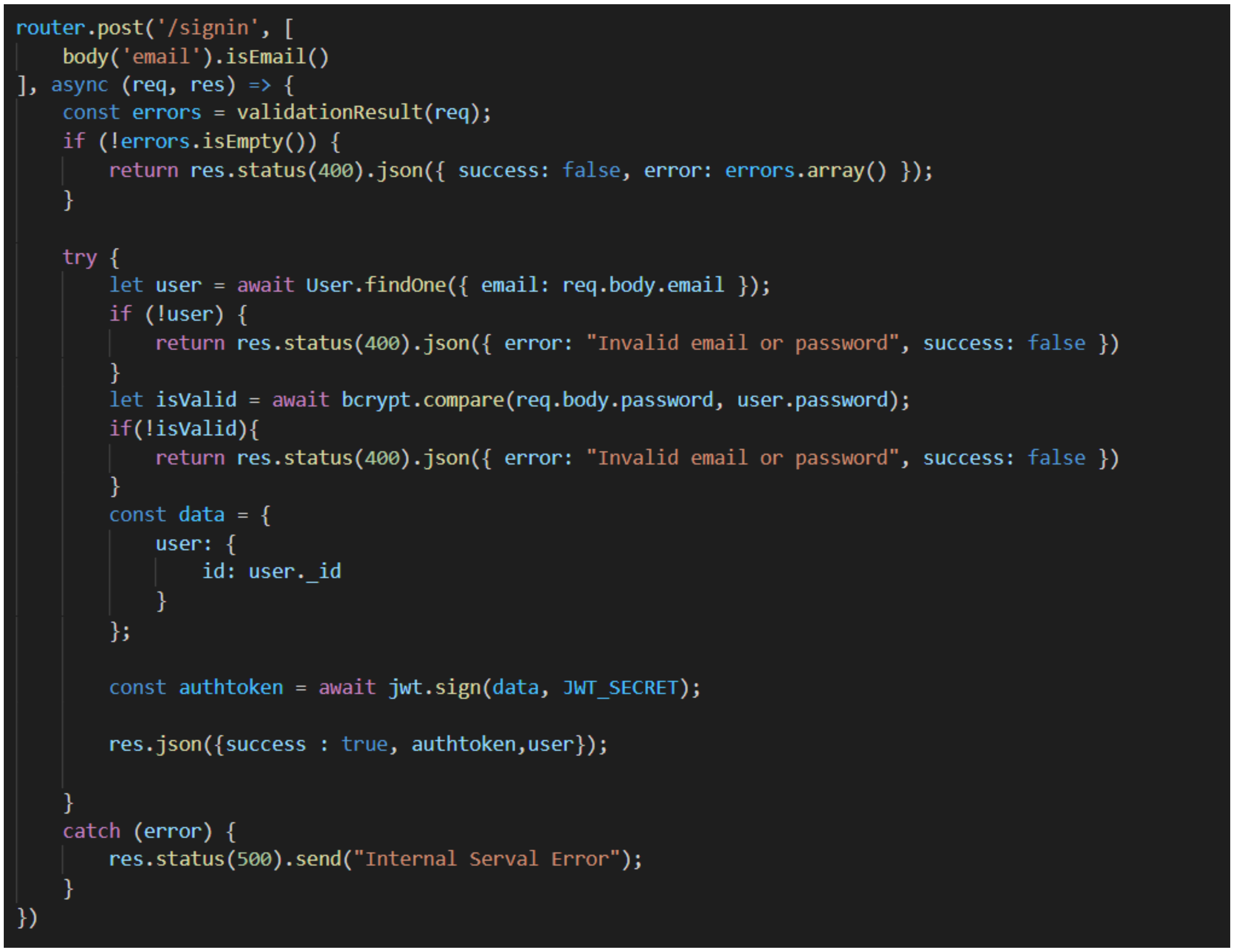
Implementation Details:

1) This function is use for signup .



```
router.post('/signup', [
    body('name').isLength({ min: 3 }),
    body('email').isEmail()
], async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({ success: false, error: errors.array() });
    }

    try {
        let user = await User.findOne({ email: req.body.email });
        if (user) {
            return res.status(400).json({ error: "Email already exists", success: false })
        }
        let salt = await bcrypt.genSalt(10);
        const password = await bcrypt.hash(req.body.password, salt);

        user = await User.create({ name: req.body.name, email: req.body.email, password });
        res.send({ success: true })
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }
})
```

2) This function is use for signin.



```
router.post('/signin', [
    body('email').isEmail()
], async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({ success: false, error: errors.array() });
    }

    try {
        let user = await User.findOne({ email: req.body.email });
        if (!user) {
            return res.status(400).json({ error: "Invalid email or password", success: false })
        }
        let isValid = await bcrypt.compare(req.body.password, user.password);
        if(!isValid){
            return res.status(400).json({ error: "Invalid email or password", success: false })
        }
        const data = {
            user: {
                id: user._id
            }
        };

        const authtoken = await jwt.sign(data, JWT_SECRET);

        res.json({success : true, authtoken,user});

    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }
})
```

3)This function is use for add songs by admin.

```
router.post('/addsong',authenticateuser, async (req, res) => {
    try {
       let song = await Song.create({ songName: req.body.songName, movieName:
req.body.movieName, singerName: req.body.singerName, songLink: req.body.songLink,
imageLink: req.body.imageLink, genre: req.body.genre });
        res.send({ success: true })
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }
})
```

4)This function is use for fetch song from database.

```
router.get('/fetchallsongs/:genre', async (req, res) => {
    const genre = req.params.genre;
    try {
        let songs = null;
        if (genre !== "all")
            songs = await Song.find({ genre: genre })
        else
            songs = await Song.find();

        res.json({ success: true, songs });
    }
    catch (error) {
        res.status(500).send("Internal Server Error");
    }
})
```

5)This function is use for delete song from database.

```
router.delete('/deletesong',authenticateuser, async (req, res) => {

    Song.deleteOne({ _id: req.body.id }).then((result) => {
        res.status(200).json(result)
    }).catch((err) => { console.warn(err) })
    console.log("item deleted")
})
```

6)This function is use for update song details.

```
router.post('/editsong',authenticateuser, async (req, res) => {
    try {
        await Song.updateOne({ _id: req.body.id }, { songName: req.body.songName,
movieName: req.body.movieName, singerName: req.body.singerName, genre:
req.body.genre });
        res.send({ success: true });
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }
})
```

7)This function is use for create playlist by user and admin both.

```
router.post('/createplaylistbyadmin', authenticateuser, async (req, res) => {
    try {
        await AdminPlaylist.create({ playlistName: req.body.playlistName, user:
req.user.id, songs: req.body.selectedSongs });
        res.send({ success: true })
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }

})

router.post('/createplaylistbyuser', authenticateuser, async (req, res) => {
    try {
        if (req.body.playlistId) {
            let playlist = await UserPlaylist.findOne({_id:req.body.playlistId})
            let song = playlist.songs.find(s => s == req.body.songId)

            if(song) {
                res.send({success: false, error:"Song already exists in
playlist!!"})
            }
            else{
                await UserPlaylist.updateOne({ _id: req.body.playlistId },
{$push:{ songs: req.body.songId }})
                playlist = await UserPlaylist.findOne({_id :
req.body.playlistId}).populate('songs')
                res.send({ success: true , playlist })
            }
        }
        else {
            let playlist = await UserPlaylist.create({ playlistName:
req.body.playlistName, user: req.user.id, songs: [req.body.songId] });
            playlist = await UserPlaylist.findOne({_id :
playlist._id}).populate('songs')
            res.send({ success: true, playlist })
        }
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }

})
```

8)This function is use for fetch playlist from database.

```
router.get('/fetchadminplaylists/:id', async (req, res) => {
    let id = req.params.id
    try {
        let playlists;
        if (id !== "all") {
            playlists = await AdminPlaylist.find({ _id: id }).populate('songs');
        }
        else {
            playlists = await AdminPlaylist.find().populate('songs');
        }
        res.send({ success: true, playlists })
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }
})
router.get('/fetchuserplaylists', authenticateuser, async (req, res) => {
    try {
        let playlists = await UserPlaylist.find({ user: req.user.id
}).populate('songs');
        res.send({ success: true, playlists })
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }
})
```

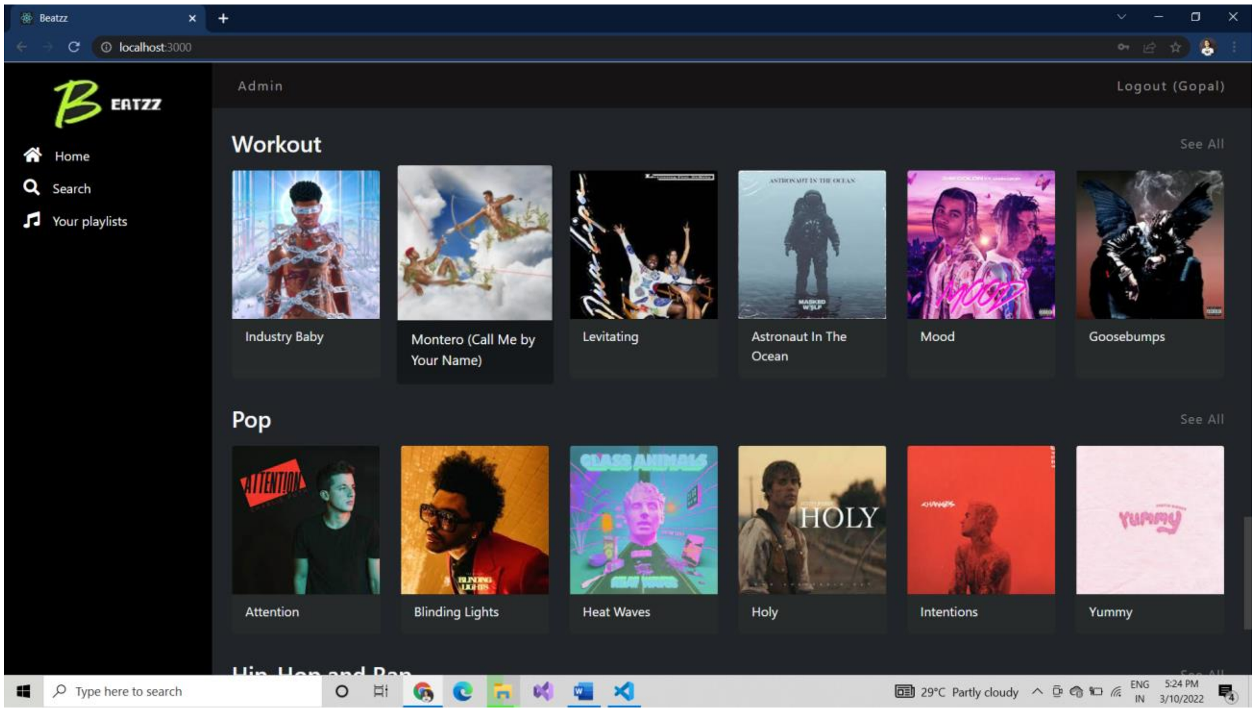
9)This function is use for delete and update playlist.

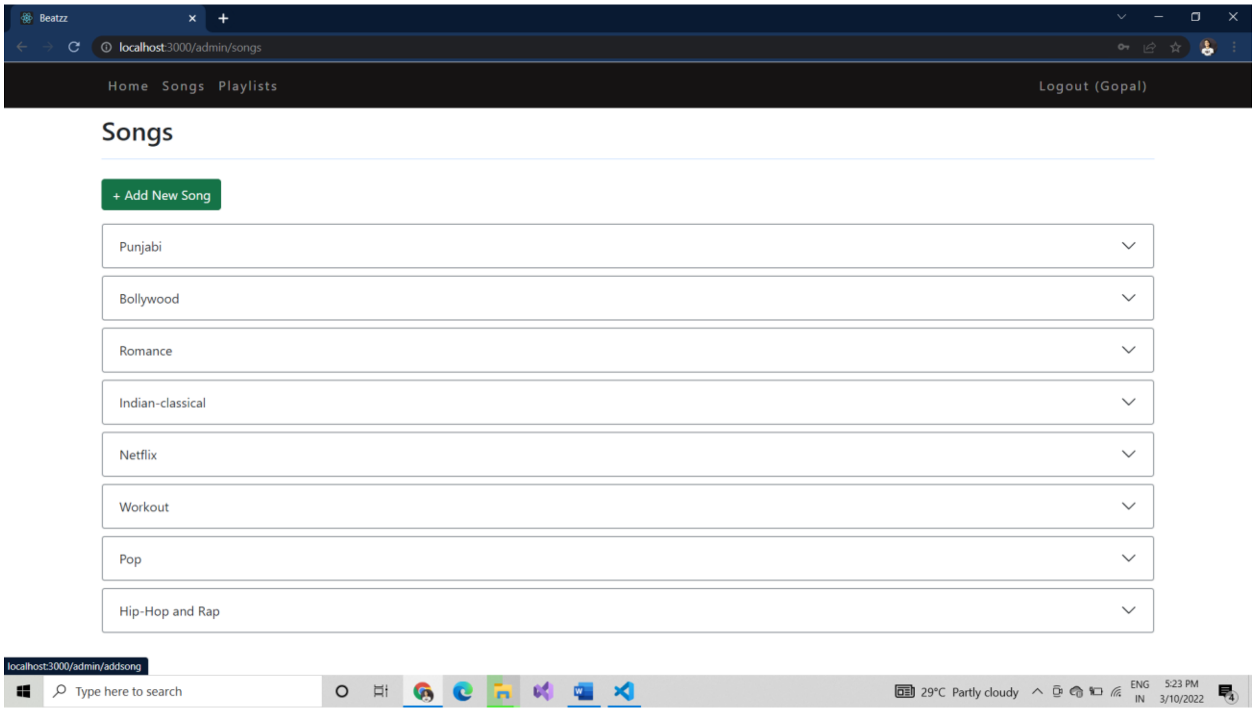
```
router.post('/editplaylist', authenticateuser, async (req, res) => {
    try {
        await AdminPlaylist.updateOne({ _id: req.body.playlistId }, {
playlistName: req.body.playlistName, songs: req.body.selectedSongs });
        res.send({ success: true })
    }
    catch (error) {
        res.status(500).send("Internal Serval Error");
    }

})
router.delete('/deleteadminplaylist', async (req, res) => {

    AdminPlaylist.deleteOne({ _id: req.body.id }).then((result) => {
        res.status(200).json(result)
    }).catch((err) => { console.warn(err) })
})


router.delete('/deleteuserplaylist', async (req, res) => {

    UserPlaylist.deleteOne({ _id: req.body.id }).then((result) => {
        res.status(200).json(result)
    }).catch((err) => { console.warn(err) })
})
```

10)This function is use for remove song from playlist.

```
router.post('/removesong', async (req, res) => {
    try{
        await UserPlaylist.updateOne({_id:req.body.playlistId},{$pull:{songs:
req.body.songId}})
        let playlist = await
UserPlaylist.findOne({_id:req.body.playlistId}).populate('songs')
        res.send({ success: true, playlist})
    }
    catch{
        ((err) => { console.warn(err) })
    }
})
```

Screenshots:





Songs

Punjabi

Bollywood

Romance

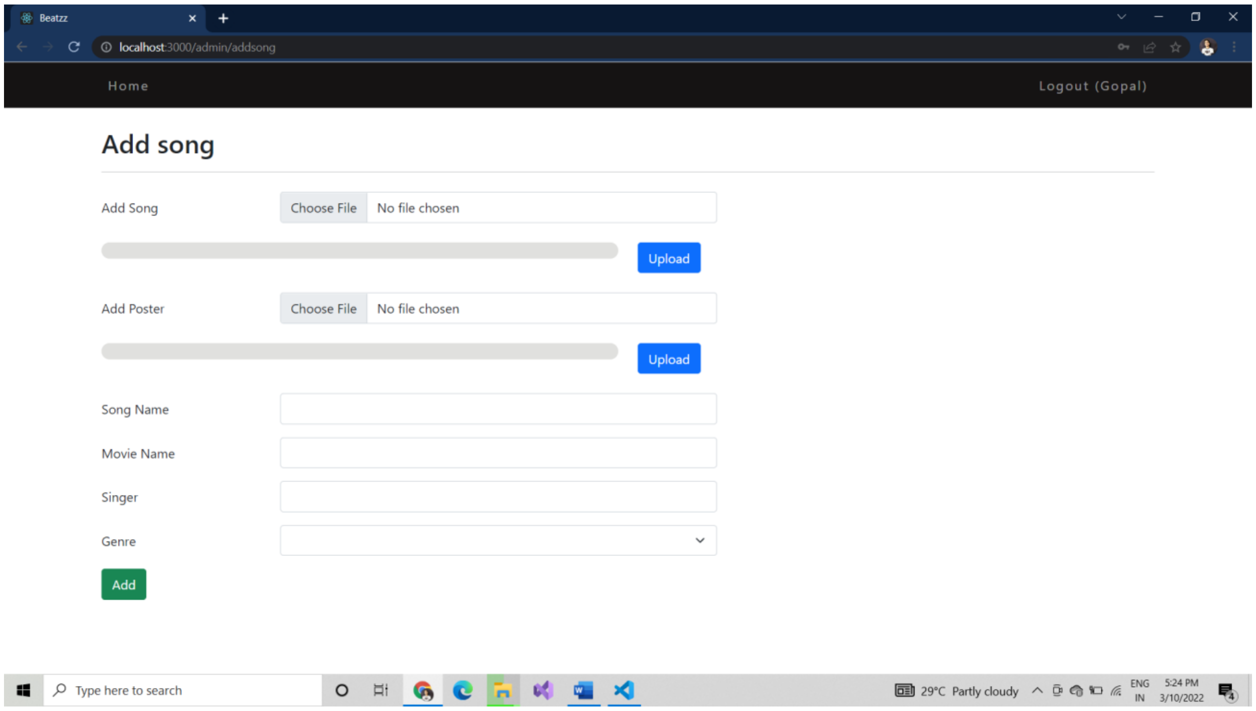
Indian-classical

Netflix

Workout

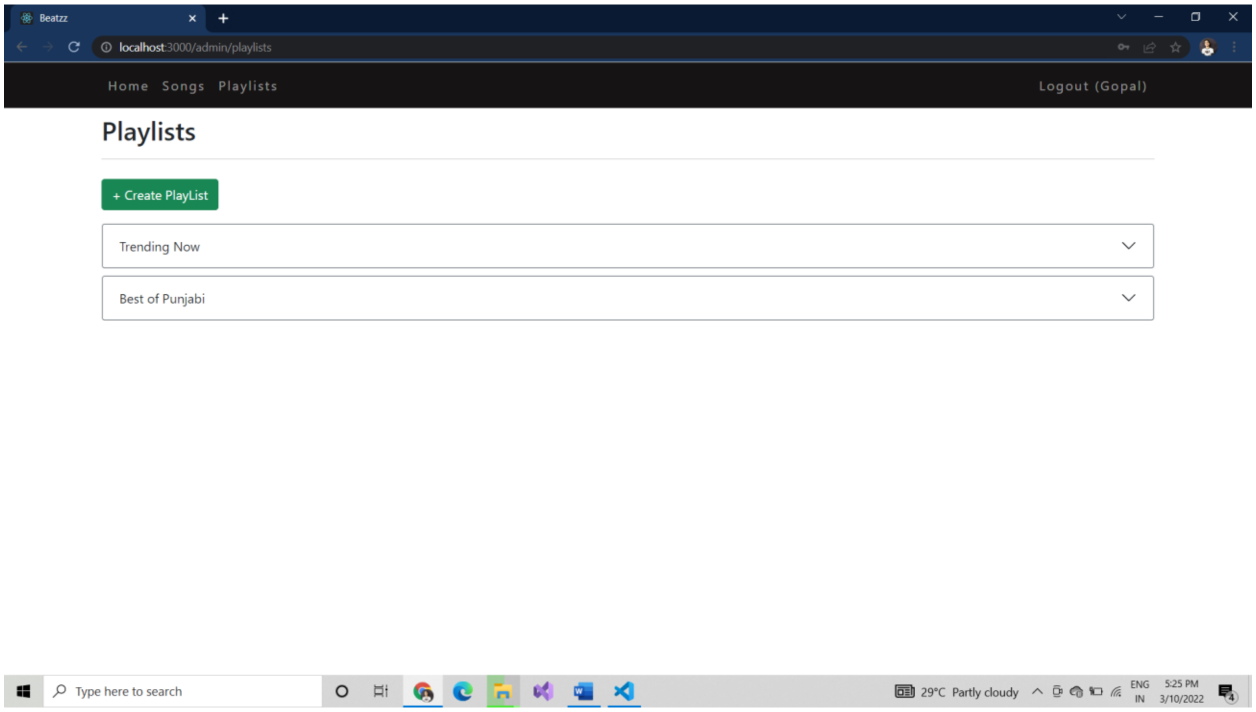
Pop

Hip-Hop and Rap



Add song

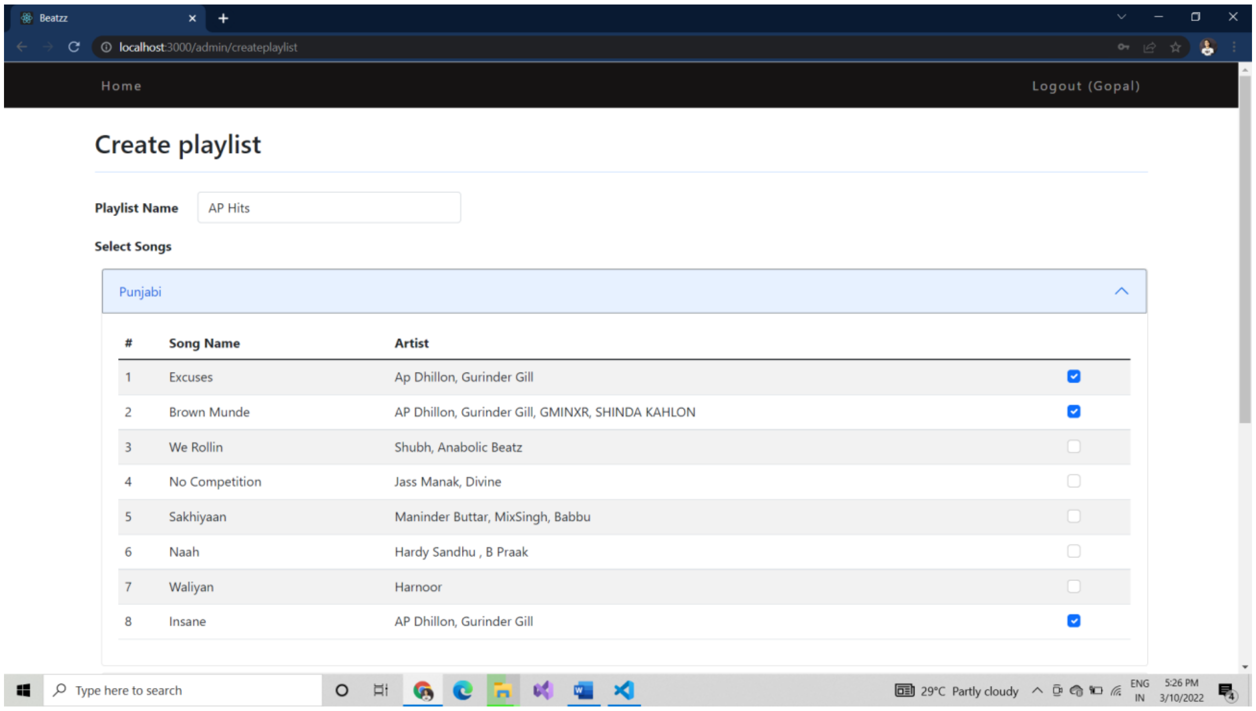
Add Song	Choose File No file chosen
Add Poster	Choose File No file chosen
Song Name	
Movie Name	
Singer	
Genre	



Playlists

Trending Now

Best of Punjabi



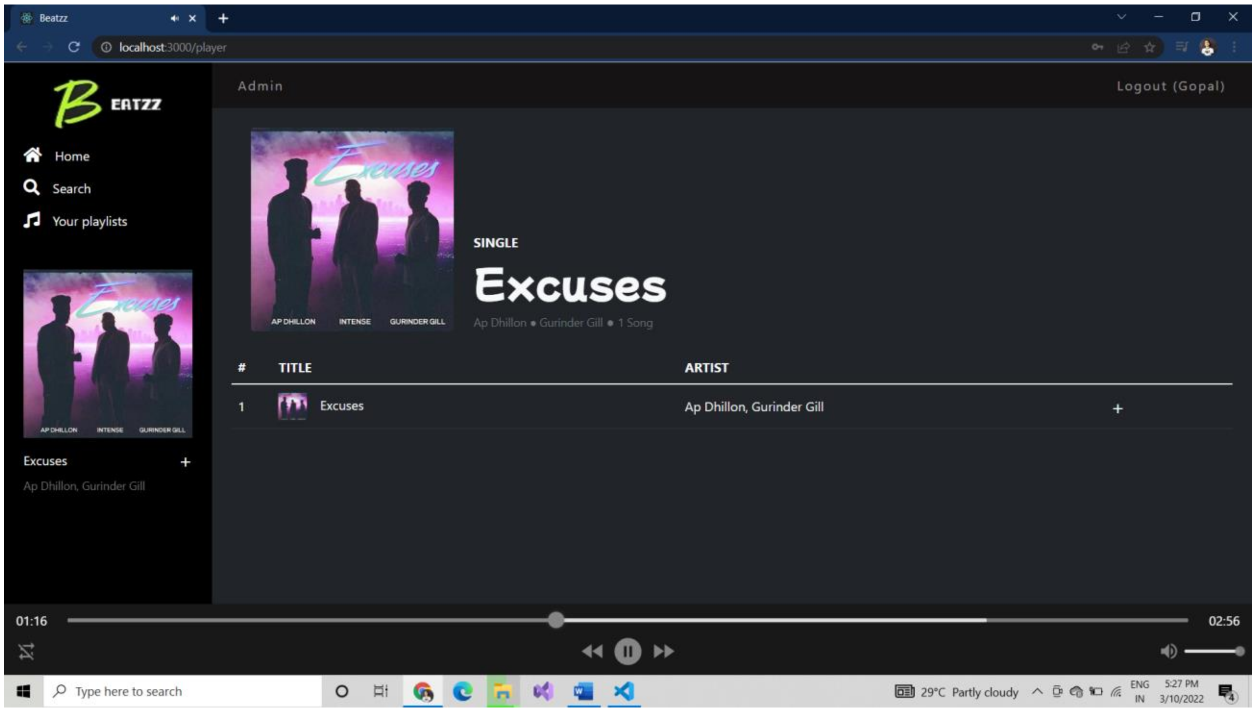
Create playlist

Playlist Name: AP Hits

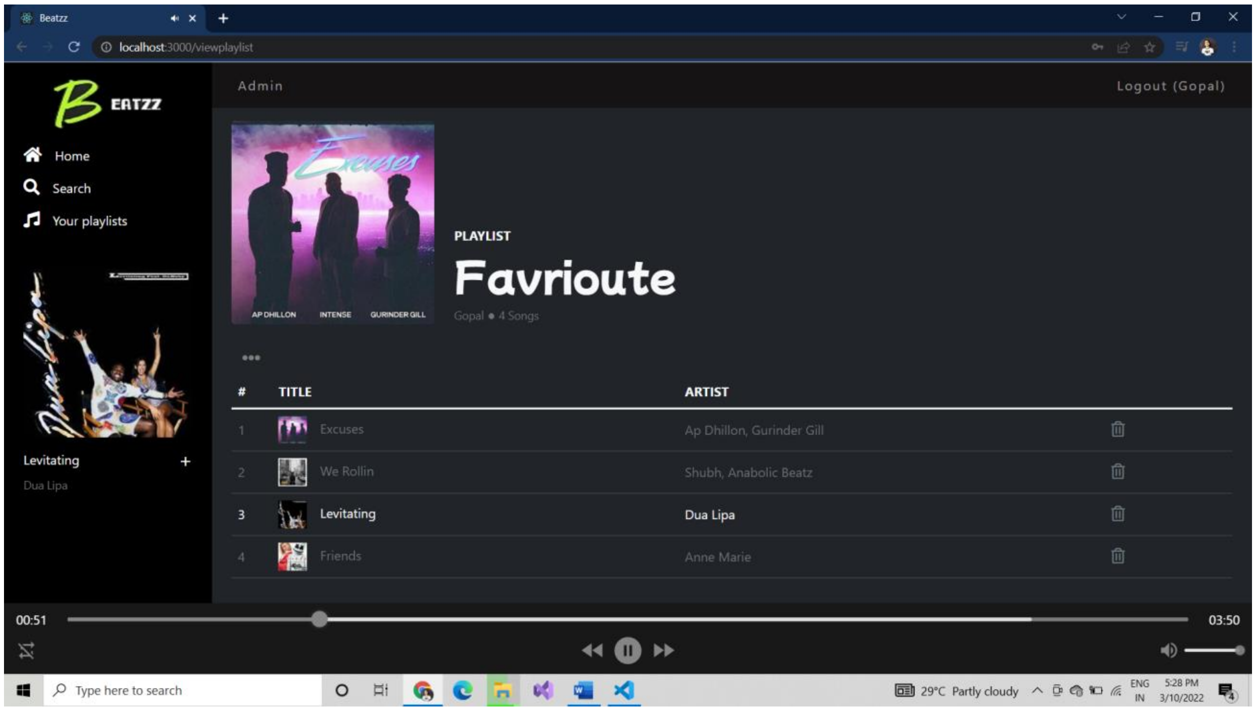
Select Songs

Punjabi

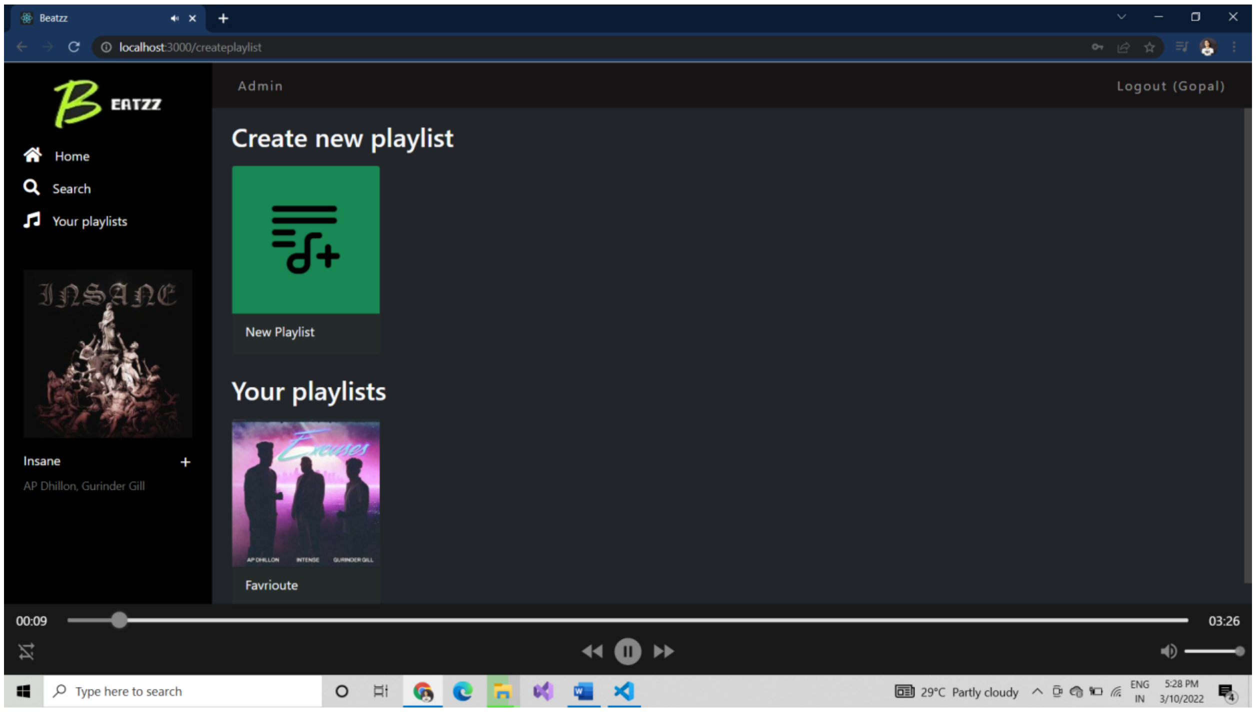
#	Song Name	Artist	
1	Excuses	Ap Dhillon, Gurinder Gill	✓
2	Brown Munde	AP Dhillon, Gurinder Gill, GMINXR, SHINDA KAHLON	✓
3	We Rollin	Shubh, Anabolic Beatz	
4	No Competition	Jass Manak, Divine	
5	Sakhiyaan	Maninder Buttar, MixSingh, Babbu	
6	Naah	Hardy Sandhu , B Praak	
7	Waliyan	Harnoor	
8	Insane	AP Dhillon, Gurinder Gill	✓

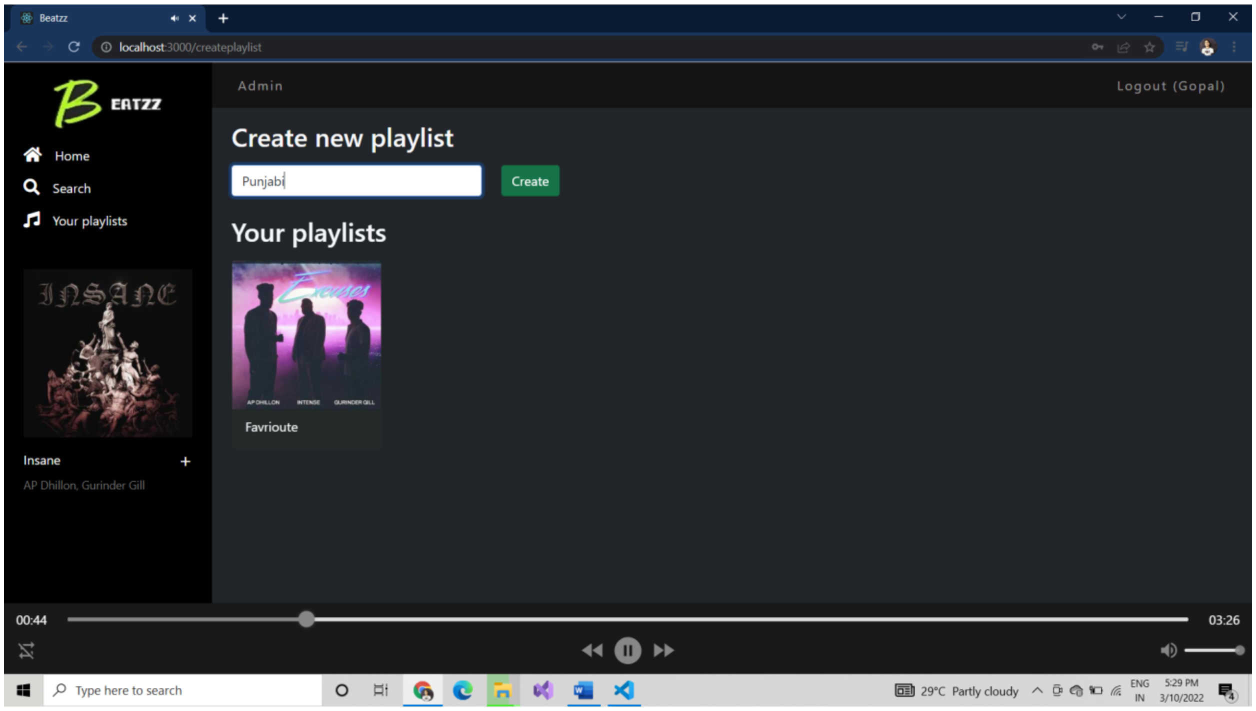


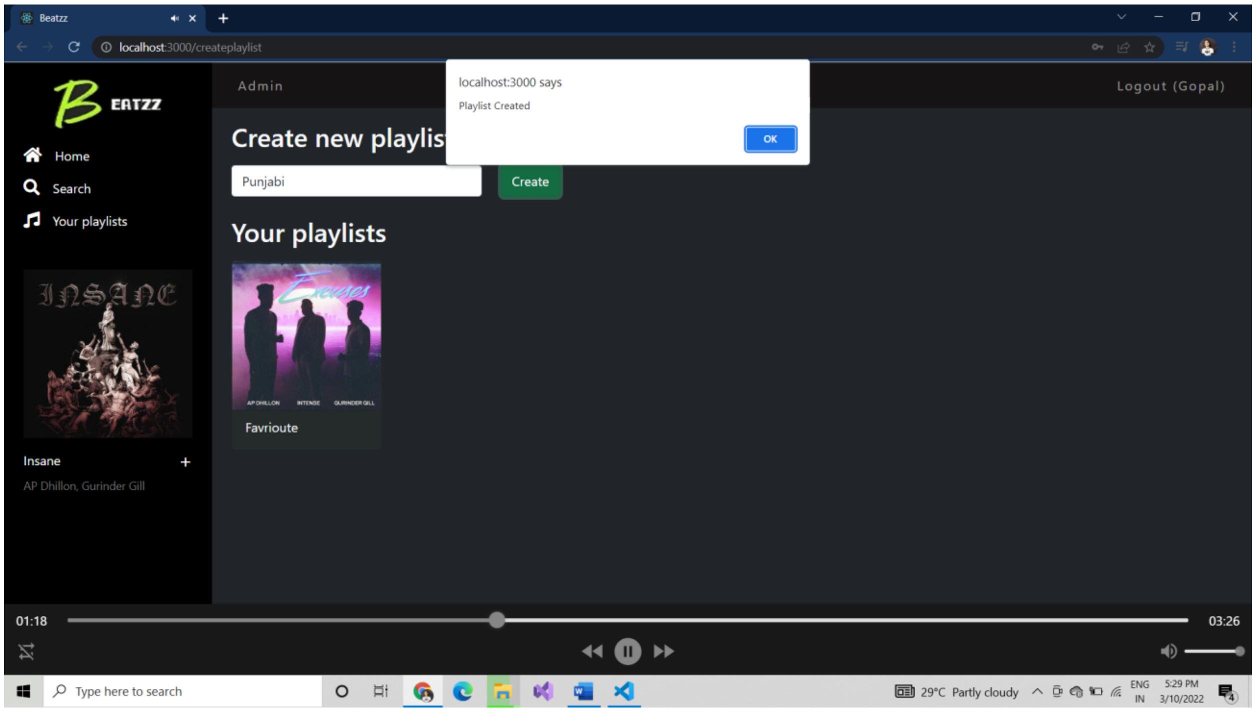
#	TITLE	ARTIST
1	Excuses	Ap Dhillon, Gurinder Gill



#	TITLE	ARTIST
1	Excuses	Ap Dhillon, Gurinder Gill
2	We Rollin	Shubh, Anabolic Beatz
3	Levitating	Dua Lipa
4	Friends	Anne Marie



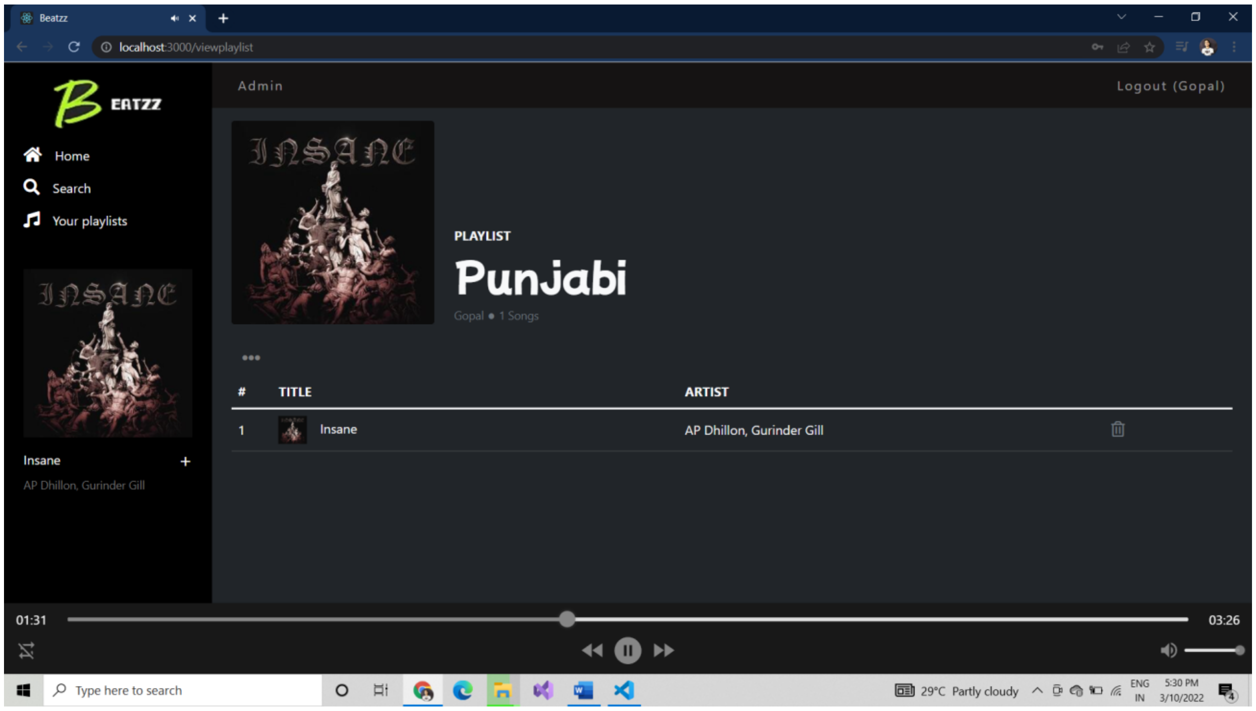




localhost:3000 says

Playlist Created

OK

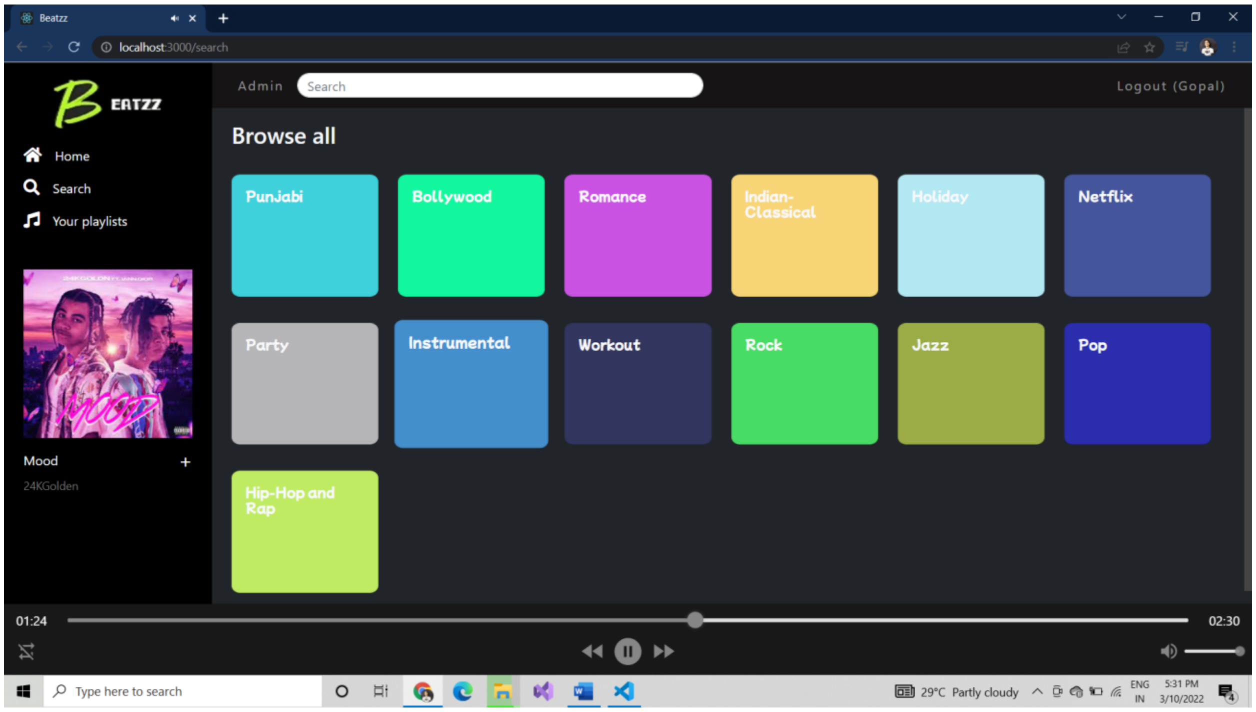


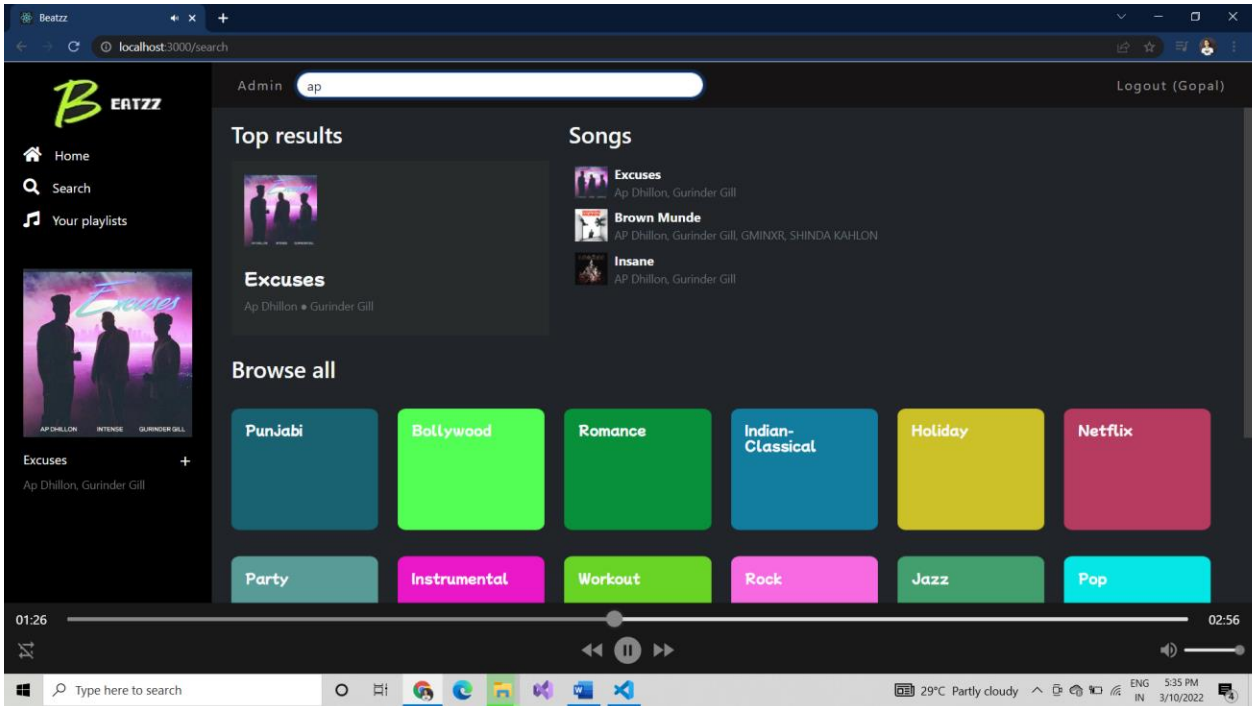
PLAYLIST

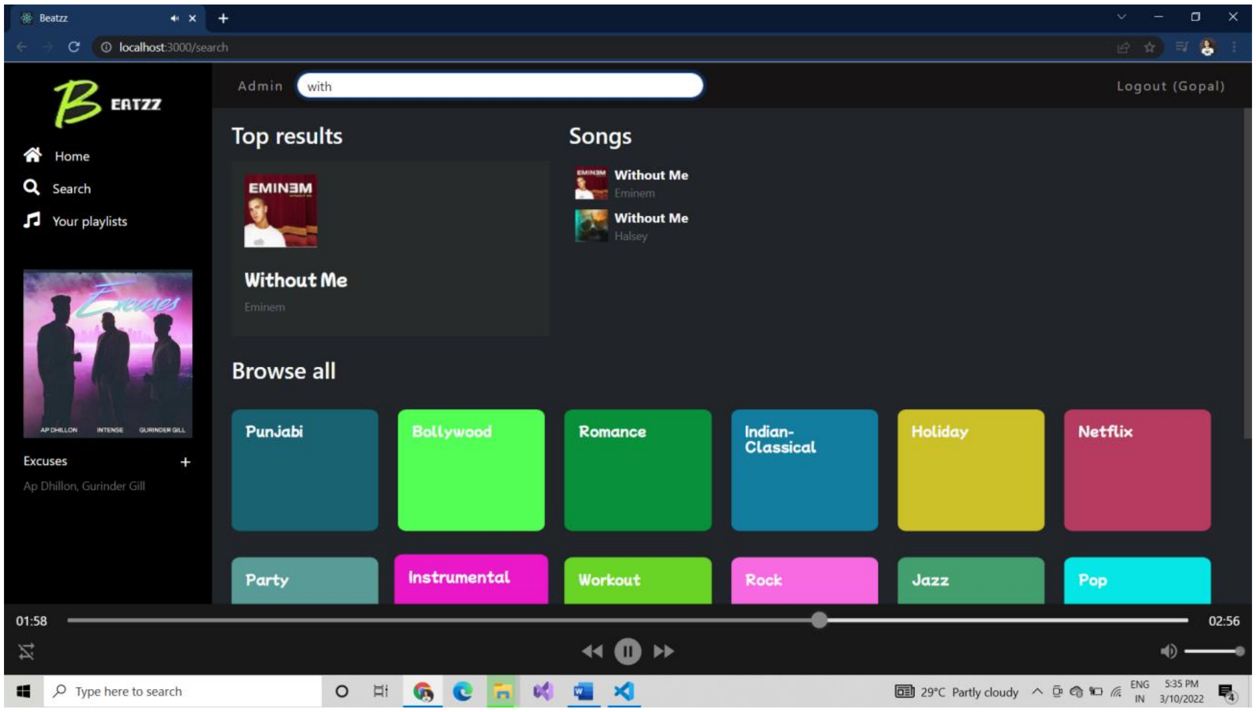
Punjabi

Gopal ● 1 Songs

#	TITLE	ARTIST
1	Insane	AP Dhillon, Gurinder Gill




Beatzz
localhost:3000/search
EATZZ
Home
Search
Your playlists
Admin
ap
Logout (Gopal)
Top results
Excuses
Ap Dhillon • Gurinder Gill
Songs
Excuses
Ap Dhillon, Gurinder Gill
Brown Munde
AP Dhillon, Gurinder Gill, GMINXR, SHINDA KAHLON
Insane
AP Dhillon, Gurinder Gill
Browse all
Punjabi
Bollywood
Romance
Indian-Classical
Holiday
Netflix
Party
Instrumental
Workout
Rock
Jazz
Pop
Excuses
Ap Dhillon, Gurinder Gill
01:26
02:56
Type here to search
29°C Partly cloudy
5:35 PM
3/10/2022


Beatzz
localhost:3000/search
EATZZ
Home
Search
Your playlists
Admin
with
Logout (Gopal)
Top results
Without Me
Eminem
Songs
Without Me
Eminem
Without Me
Halsey
Browse all
Punjabi
Bollywood
Romance
Indian-Classical
Holiday
Netflix
Party
Instrumental
Workout
Rock
Jazz
Pop
Excuses
Ap Dhillon, Gurinder Gill
01:58
02:56
Type here to search
29°C Partly cloudy
5:35 PM
3/10/2022

Testing:

Sr. No	Test Scenario	Expected Result	Actual Result	Status
1.	Sign Up without required field or existing email.	User Should not able to Signup.	User will get error message on the same page	Success
2.	Signup with not valid Username , email or password	User should not able to Signup.	User will get error message on the same page.	Success
3.	Login with incorrect email or password	User should not able to Login	User will get error message.	Success
4.	Login with correct credentials.	User should able to Login.	User will logged in and redirect to Home page.	Success
5.	If Admin is logged in.	Admin should able to manage Admin panel.	Admin will be able to manage Songs/Playlists	Success
6.	If User is logged in.	User should able to create/edit/delete playlists.	User will be able to create new playlists and User can add song to existing Playlists.	Success

<u>Conclusion</u>:

All the modules are implemented after understanding all models and diagram of the system.

Modules which implemented successfully are as below:

- Login
- Registration
- Reset password
- Play song
- Search song by song name or artist name
- Create new playlist
- Add song to existing playlist

After the implementation all modules were working properly.

<u>Limitation:</u>

- Admin can upload one song at a time rather than whole playlist or album.
- Artists don't have any account

<u>Future Extension:</u>

- Recently played songs or playlist will be display on home page.
- User will be able to shuffle all songs which are in particular playlist.
- Search module will be improve.
- User will be able to download songs for free.
- User will be able to give a **LIKE** to any song and liked songs will be saved.

Bibliography:

References:

- React JS
- Node JS
- Express JS
- Mongoose JS
- JavaScript
- Bootstrap
- Stack overflow
- react-h5-audio-player
- w3schools