

Web Information Management

MyMelody - Music Streaming Website

Suggestion Report

Group Number 25

Group Members- Neha Khairnar - 191081036

Prerana Shelke - 191081061

Abstract

A music Streaming website is an effective platform to listen to music. On this website, registered users can get access to multiple songs. This is also one of the users and administrator-friendly websites in which a user can create multiple playlists of his favorite songs. For users, it is easy to handle a playlist. Users can create a playlist, update and modify it at any time.

Introduction

Music streaming services are online applications that help you to listen to your favorite songs with ease. This program enables you to search for music by artist name, and album. You can use these apps to create and share your own playlist with other people. Enjoying music is unique to humans. Music floods the brain with a chemical called dopamine. Dopamine is the chemical in the brain associated with pleasure, motivation and reward. Studies have shown that certain pieces of classical music will have the same effect on everyone.

Music plays different roles in the lives of people. Some people listen to music for entertainment, some listen to music on their way to the gym or while travelling. Some people consider music as a thing that makes their boring lives fun and that's why because of various reasons given by various people music has become an integral and important part of our lives.

In today's era we have easy access to the internet which plays an important role in bringing the world closer. Music also plays a role in connecting the world because we believe that music has no language. Hence the aim of this project is to develop a music streaming website which will play an important role in connecting people and making the world a better place.

Problem Statement

The aim of this project is to develop a web application which can-

1. Display and play music
2. Perform user sign up and login activity.
3. Display user details.
4. Update password and email of the user.

5. Create Playlists.
6. Display artists and albums
7. Display Music files to the user
8. Provide a music playing facility to the user.
9. Music playing bar with play, pause and volume control button.
10. Add image for user created playlist.

Existing System

There are many existing music playing android and web applications available in the market these days. Some of the music streaming applications available in the market are spotify, soundcloud, gaana , apple music, soundcloud , wynk music, etc.

Some of the common features provided by available music streaming applications are-

1. User account creation.
2. Sign up and login facility.
3. Search for music , artists and albums.
4. Follow favourite artists.
5. Listen to their favourite music.
6. Create playlists of different artists and soundtracks.
7. User-friendly interface.
8. Settings, notifications and other basic facilities are provided.
9. Some music streaming services provide podcasts too.
10. Radio channels streaming is also provided by some music streaming web applications.

In this project we aim to include all the facilities provided by other music streaming websites and make it as user friendly as possible. Our database design follows the ACID properties of DBMS.

Tech Stack

- HTML and CSS - To design the frontend of the website.
- JavaScript - To write functions to play music and handle the backend data.
- Visual Studio Code- Editor for writing code.
- Node & Express-js: We used node and express-js to allow users to navigate through the website blazingly fast.
- MongoDB: we used MongoDB as our no-SQL database to store user data and storing the data related to music.
- React- To create web app and install required dependencies.

MERN Stack: MERN Stack is a Javascript Stack that is used for easier and faster deployment of full-stack web applications. MERN Stack comprises 4 technologies namely: [MongoDB](#), [Express](#), [React](#) and [Node.js](#). It is designed to make the development process smoother and easier.

Each of these 4 powerful technologies provides an end-to-end framework for the developers to work in and each of these technologies play a big part in the development of web applications.



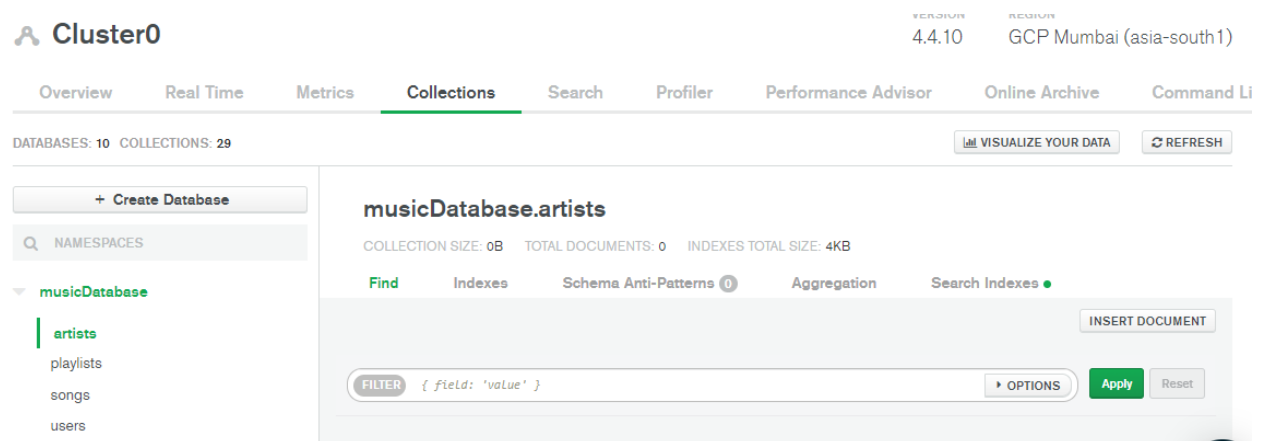
Procedure Followed To Create Website

1. Created an account on MongoDB Cloud ,created a new cluster and set required access details.
2. Created a MongoDB database named **musicDatabase** to store songs and user data. Created models by creating schemas for songs, playlist and user.

3. Established connection between MongoDB database and server of the web application.
4. Downloaded necessary dependencies and tools like Robo3T and Postman. Robo 3T, formerly known as Robomongo is a popular **resource for MongoDB hosting deployments**. It provides a Graphical User Interface (GUI) to interact with bricks of data through visual indicators rather than text based interface. It is free and lightweight. Postman is an application **used for API testing**. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.
5. Created MVC architecture to develop the website.
6. Technologies used are - **Express.js , Node.js, MongoDB, Nodemon, Cors, bcrypt**.
7. Established connection between web application's server and MongoDB cloud cluster which will make it handy to store data and help to manage the data effectively.
8. Created a react app named admin to add songs to the website and handle the data.
9. Established connection between website and firebase to store songs and user's data efficiently.

Database Design

Name of database:- musicDatabase



User Collection

```
users
myFirstDatabase
sample_airbnb
sample_analytics
sample_geospatial
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
```

QUERY RESULTS 1-3 OF 3

```
_id: ObjectId("61cd966836d3e4e567be3ee9")
name: "neha"
date: "22"
email: "nehak2209pc@gmail.com"
gender: "female"
isAdmin: true
> likedSongs: Array
month: "september"
password: "Neha@123"
> playlist: Array
year: "2001"
```

```
_id: ObjectId("61cdd18257188fac7e27ee26")
name: "Nehaa"
email: "nmarkchantaedojimink@gmail.com"
password: "52b$105feTJNkb/xhFIdAR6kltVwez04npGypnWY1XsurvLmijPs7oCHscq"
gender: "female"
month: "09"
date: "22"
year: "2001"
> likedSongs: Array
> playlists: Array
```

Songs Collection

artists

playlists

songs

users

myFirstDatabase

sample_airbnb

sample_analytics

sample_geospatial

sample_mflix

sample_restaurants

sample_supplies

sample_training

sample_weatherdata

FILTER

{ field: 'value' }

OPTIONS

Apply

Reset

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("61cdd63a57188fac7e27ee41")
name: "sorry"
artist: "Justin Bieber"
song: "https://firebasestorage.googleapis.com/v0/b/music-application-efeca.ap..."
img: "https://firebasestorage.googleapis.com/v0/b/music-application-efeca.ap..."
duration: "199"
__v: 0
```

```
_id: ObjectId("61cdd79c57188fac7e27eeb4")
name: "Baby"
artist: "Justin Bieber"
song: "https://firebasestorage.googleapis.com/v0/b/music-application-efeca.ap..."
img: "https://firebasestorage.googleapis.com/v0/b/music-application-efeca.ap..."
duration: "0"
v: 0
```

Playlist Collection

The screenshot shows the MongoDB Compass web interface. On the left, a sidebar lists collections under 'musicDatabase', including 'artists', 'playlists' (highlighted), 'songs', and 'users'. Below this are other databases like 'myFirstDatabase' and 'sample_airbnb'. The main panel shows a query filter: '{ field: 'value' }'. Below the filter, it displays 'QUERY RESULTS 1-9 OF 9'. The first result is a document with the following structure:

```

{
  "_id": ObjectId("61cdd66057188fac7e27ee53"),
  "name": "Neha",
  "user": ObjectId("61cdd18257188fac7e27ee26"),
  "desc": "By Nehaa",
  "songs": Array
    [
      {
        "ing": "https://firebasestorage.googleapis.com/v0/b/music-application-efeca.ap...",
        "_v": 6
      }
    ]
}

```

At the bottom right, there is a small circular icon with a person silhouette.

Data Tables

1. users Table

Name	DataType
name	String
email	String
password	String
gender	String
month	String
date	String
year	String
likedSongs	Array of string

playlist	Array of string
isAdmin	Boolean

2. playlist Table


Name	Data Type
name	String
user	objectId
desc	string
songs	Array of strings
img (Image)	String

3. song Table

Name	Data Type
name	String
artist	String
song	String
img	String
duration	String

Models

- Song Model

```
server > models >  song.js > ...  
1  const mongoose = require("mongoose");  
2  const Joi = require("joi");  
3  
4  const songSchema = new mongoose.Schema({  
5    name: { type: String, required: true },  
6    artist: { type: String, required: true },  
7    song: { type: String, required: true },  
8    img: { type: String, required: true },  
9    duration: { type: String, required: true },  
10 });  
11  
12 const validate = (song) => {  
13   const schema = Joi.object({  
14     name: Joi.string().required(),  
15     artist: Joi.string().required(),  
16     song: Joi.string().required(),  
17     img: Joi.string().required(),  
18     duration: Joi.number().required(),  
19   });  
20   return schema.validate(song);  
21 };  
22  
23 const Song = mongoose.model("song", songSchema);  
24  
25 module.exports = { Song, validate };  
26
```

- User Model

```
const mongoose = require("mongoose");
const jwt = require("jsonwebtoken");
const Joi = require("joi");
const passwordComplexity = require("joi-password-complexity");

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, unique: true, required: true },
  password: { type: String, required: true },
  gender: { type: String, required: true },
  month: { type: String, required: true },
  date: { type: String, required: true },
  year: { type: String, required: true },
  likedSongs: { type: [String], default: [] },
  playlists: { type: [String], default: [] },
  isAdmin: { type: Boolean, default: false },
});

userSchema.methods.generateAuthToken = function () {
  const token = jwt.sign(
    { _id: this._id, name: this.name, isAdmin: this.isAdmin },
    process.env.JWTPRIVATEKEY,
    { expiresIn: "7d" }
  );
  return token;
};
```

```
17 });
18
19 userSchema.methods.generateAuthToken = function () {
20   const token = jwt.sign(
21     { _id: this._id, name: this.name, isAdmin: this.isAdmin },
22     process.env.JWTPRIVATEKEY,
23     { expiresIn: "7d" }
24   );
25   return token;
26 };
27
28 const validate = (user) => {
29   const schema = Joi.object({
30     name: Joi.string().min(5).max(10).required(),
31     email: Joi.string().email().required(),
32     password: passwordComplexity().required(),
33     month: Joi.string().required(),
34     date: Joi.string().required(),
35     year: Joi.string().required(),
36     gender: Joi.string().valid("male", "female", "non-binary").required(),
37   });
38   return schema.validate(user);
39 };
40
41 const User = mongoose.model("user", userSchema);
42
43 module.exports = { User, validate };
44
```

- Playlist Model (song's playlist)

```
const mongoose = require("mongoose");
const Joi = require("joi");

const ObjectId = mongoose.Schema.Types.ObjectId;

const playlistSchema = new mongoose.Schema({
  name: { type: String, required: true },
  user: { type: ObjectId, ref: "user", required: true },
  desc: { type: String },
  songs: { type: Array, default: [] },
  img: { type: String },
});

const validate = (playList) => {
  const schema = Joi.object({
    name: Joi.string().required(),
    user: Joi.string().required(),
    desc: Joi.string().allow(""),
    songs: Joi.array().items(Joi.string()),
    img: Joi.string().allow(""),
  });
  return schema.validate(playList);
};

const PlayList = mongoose.model("playList", playlistSchema);

module.exports = { PlayList, validate };
```

Firestore Connection Details(Client and Admin)

```
firebase.js x
client > src > firebase.js > ...
1  import { initializeApp } from "firebase/app";
2  import { getStorage } from "firebase/storage";
3
4  const firebaseConfig = {
5    apiKey: process.env.REACT_APP_API_KEY,
6    authDomain: process.env.REACT_APP_AUTH_DOMAIN,
7    projectId: process.env.REACT_APP_PROJECT_ID,
8    storageBucket: process.env.REACT_APP_STORAGE_BUCKET,
9    messagingSenderId: process.env.REACT_APP_MESSAGING_SENDER_ID,
10   appId: process.env.REACT_APP_APP_ID,
11 };
12
13 const app = initializeApp(firebaseConfig);
14 const storage = getStorage(app, process.env.REACT_APP_BUCKET_URL);
15 export default storage;
16
```

```
.env x
client > .env
1  REACT_APP_API_URL = "http://localhost:8080/api"
2  REACT_APP_BUCKET_URL="gs://music-application-efeca.appspot.com"
3  REACT_APP_API_KEY = "AIzaSyA0t7URjzzKCnI6fZ1ZQbAmHxzsfUBmRHs"
4  REACT_APP_AUTH_DOMAIN = "music-application-efeca.firebaseio.com"
5  REACT_APP_PROJECT_ID = "music-application-efeca"
6  REACT_APP_STORAGE_BUCKET = "music-application-efeca.appspot.com"
7  REACT_APP_MESSAGING_SENDER_ID = "379702883079"
8  REACT_APP_APP_ID = "1:379702883079:web:a58bb32a043eda97f04ba2"
9  REACT_APP_DATABASE_URL = "https://music-application-efeca-default-rtdb.asia-southeast1.firebaseio.com"
10 REACT_APP_MEASUREMENT_ID = "G-193NY7WM4M"
11
```

MongoDB Connection Details(Server)

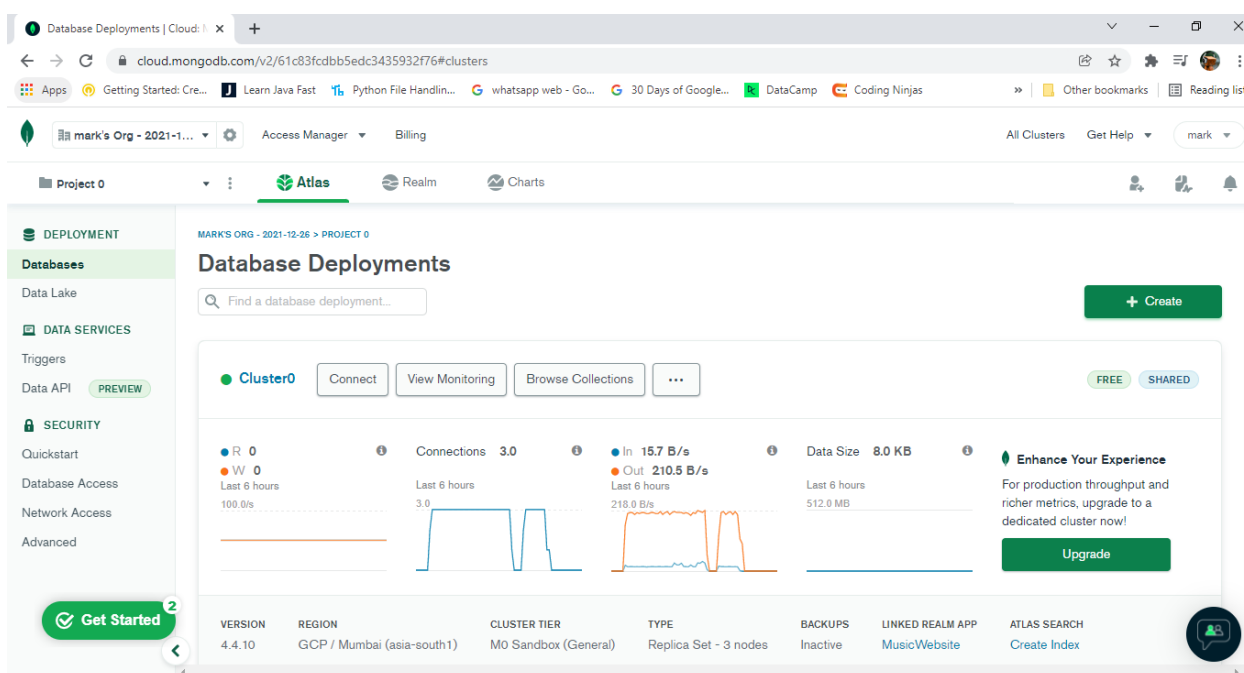
```

.env
server > .env
1 DB = mongodb+srv://neha:Neha%40123@cluster0.cczfw.mongodb.net/musicDatabase?retryWrites=true&w=majority
2 JWTPRIVATEKEY= 1234jfoehmwoc
3 SALT = 10

```

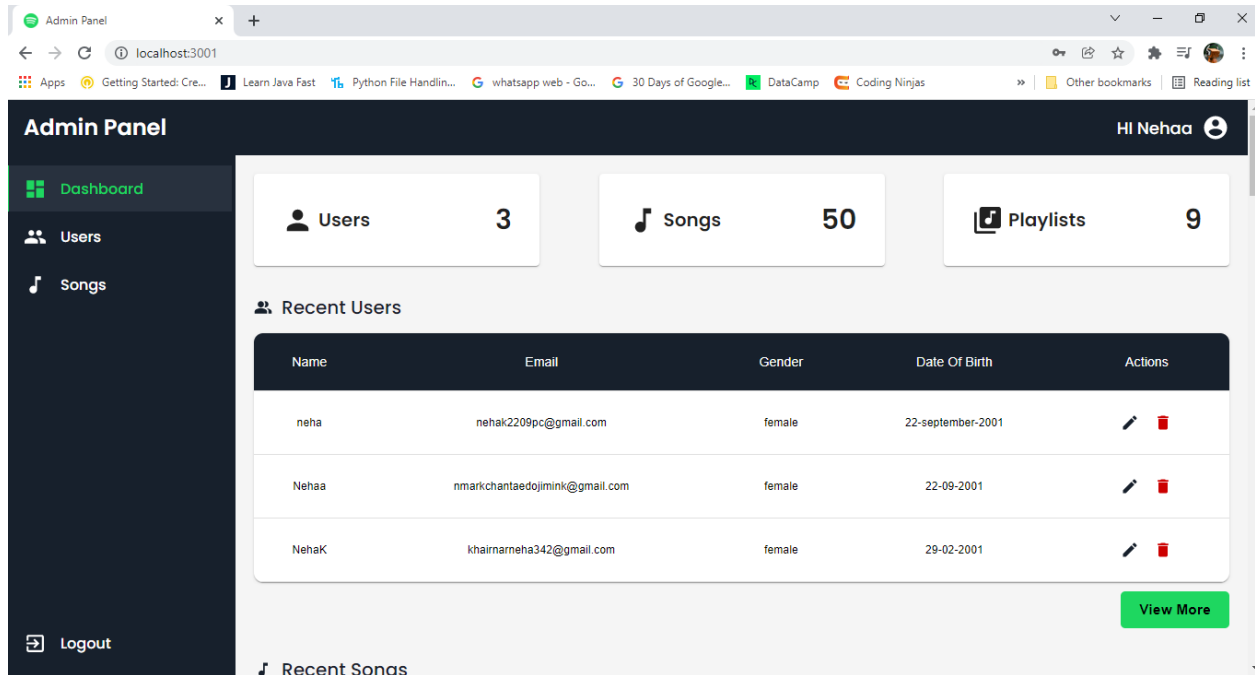
MongoDB cloud cluster

- MongoDB cloud is used to create databases by using cloud database to store data.
- Here we are using Atlas to create clusters and store data.
- **MongoDB Atlas** is a multi-cloud application data platform. At its core is our fully managed cloud database for modern applications. Atlas is the best way to run MongoDB, the leading non-relational database. Atlas is free to use.
- In the screenshot below **Cluster0** is the cluster created to store data. This cluster is connected to our web server using special token provided by MongoDB cloud database.



Result

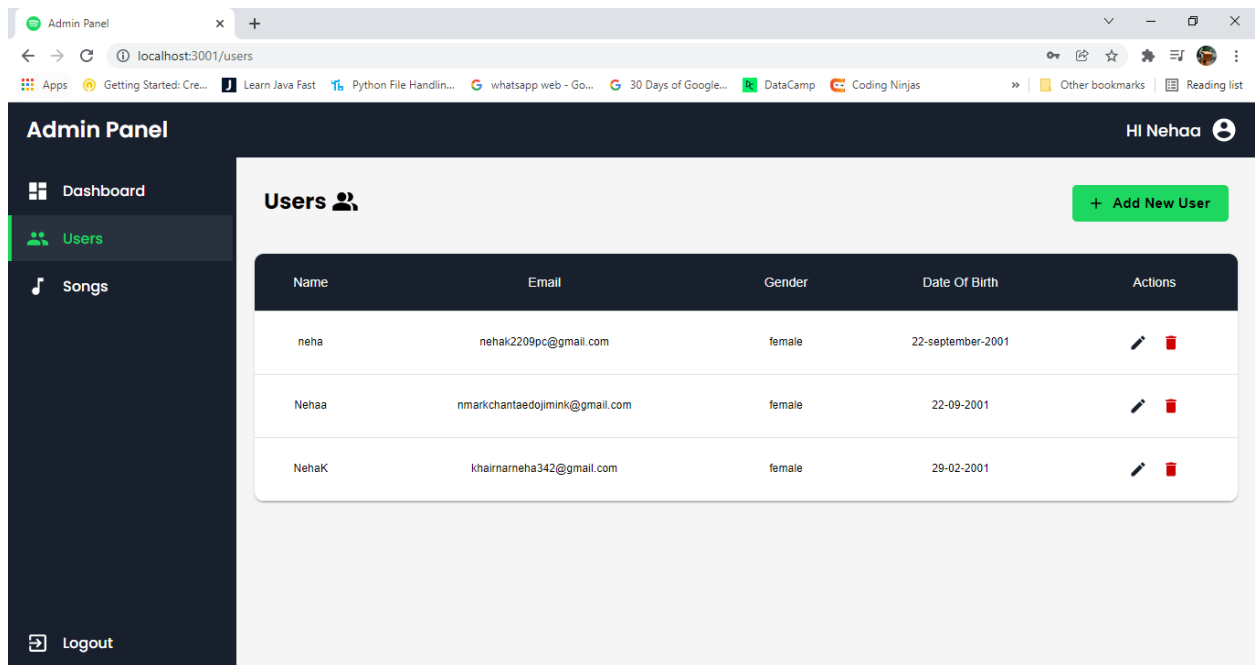
• Admin Panel (Dashboard)



The screenshot shows the Admin Panel Dashboard. The left sidebar contains links for Dashboard, Users, Songs, and Logout. The main content area displays three summary cards: Users (3), Songs (50), and Playlists (9). Below these is a 'Recent Users' section with a table listing three users: neha, Nehaa, and NehaK. Each user entry includes their email, gender, date of birth, and actions (edit and delete). A 'View More' button is located at the bottom right of the table.

Name	Email	Gender	Date Of Birth	Actions
neha	nehak2209pc@gmail.com	female	22-september-2001	Edit Delete
Nehaa	nmarkchantaedojimink@gmail.com	female	22-09-2001	Edit Delete
NehaK	khairneha342@gmail.com	female	29-02-2001	Edit Delete

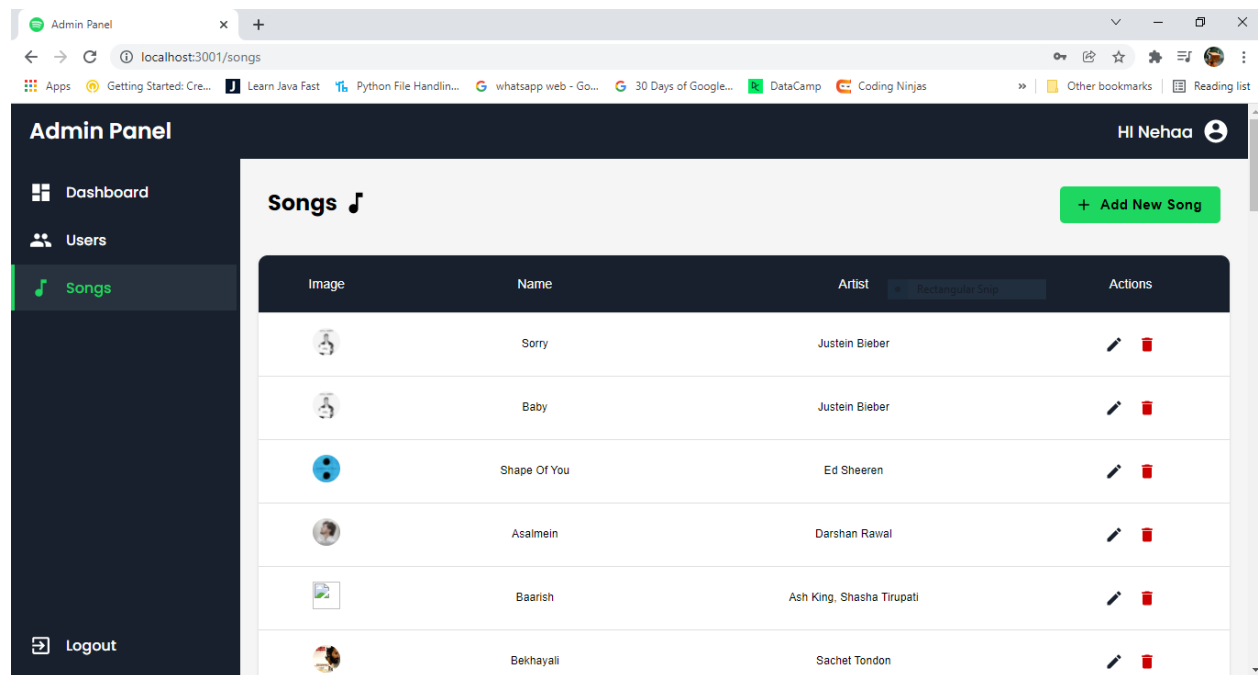
• Admin Panel (Users)



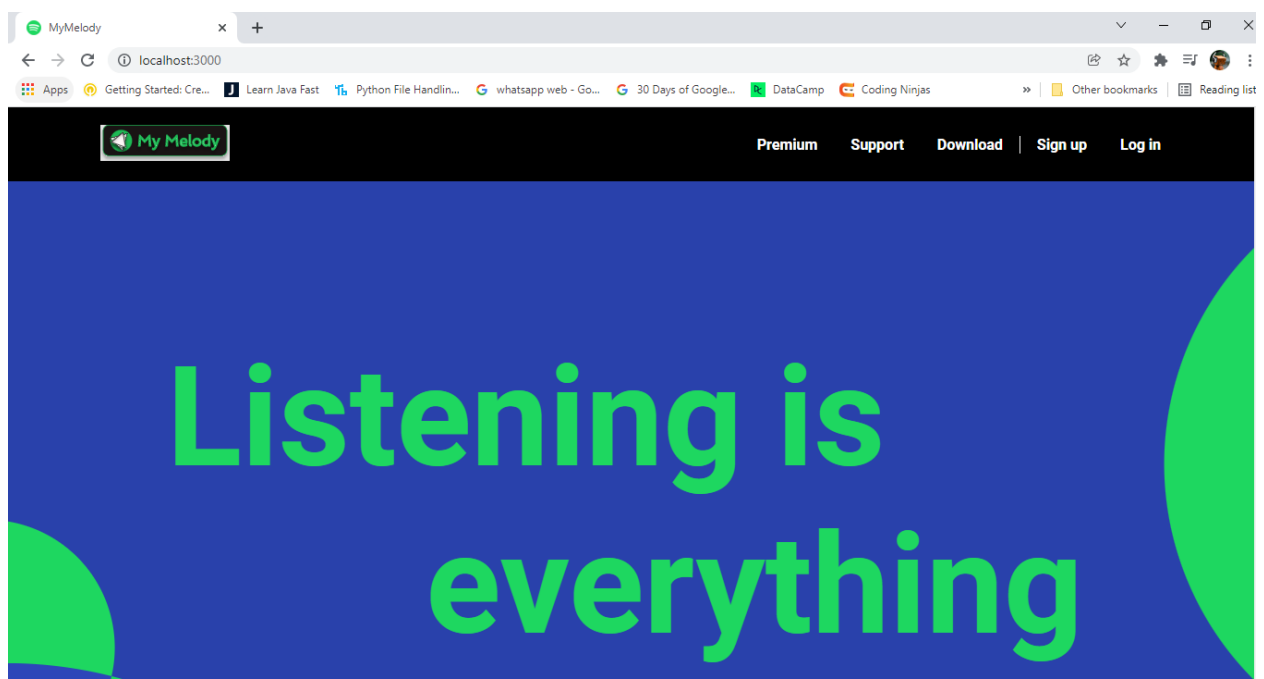
The screenshot shows the Admin Panel Users page. The left sidebar contains links for Dashboard, Users, Songs, and Logout. The main content area displays a 'Users' section with a table listing three users: neha, Nehaa, and NehaK. Each user entry includes their email, gender, date of birth, and actions (edit and delete). A '+ Add New User' button is located at the top right of the table.

Name	Email	Gender	Date Of Birth	Actions
neha	nehak2209pc@gmail.com	female	22-september-2001	Edit Delete
Nehaa	nmarkchantaedojimink@gmail.com	female	22-09-2001	Edit Delete
NehaK	khairneha342@gmail.com	female	29-02-2001	Edit Delete

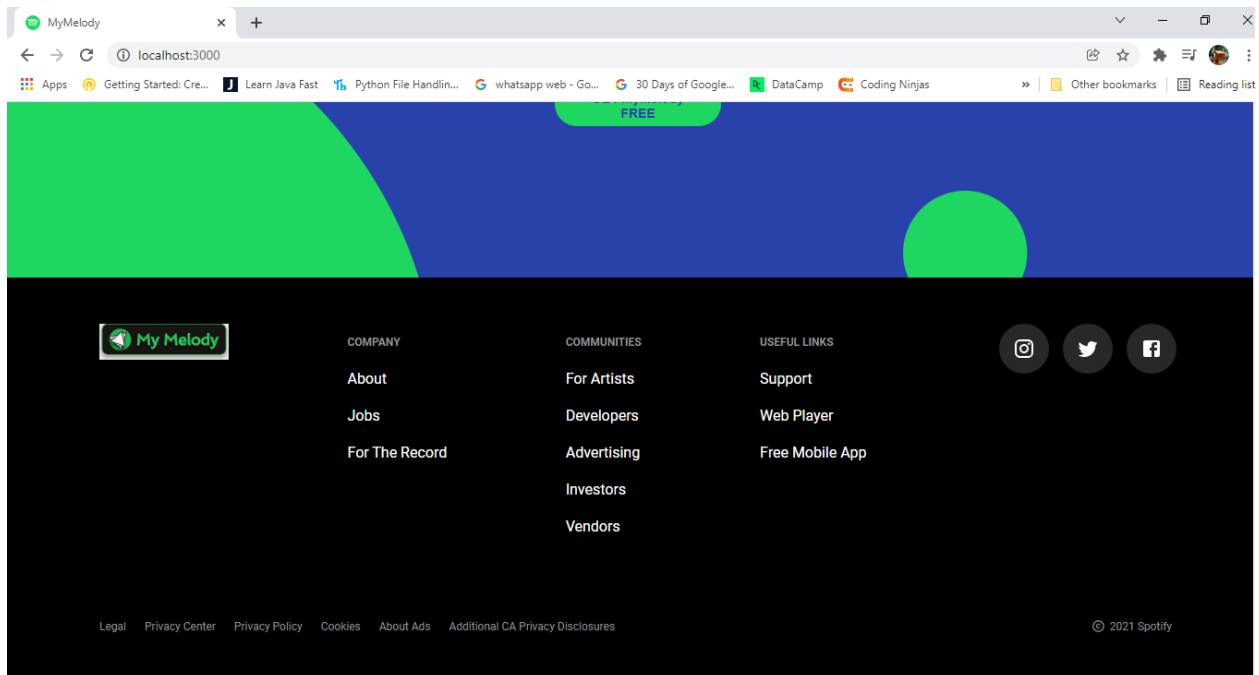
• Admin Panel (Songs)



• Home Screen

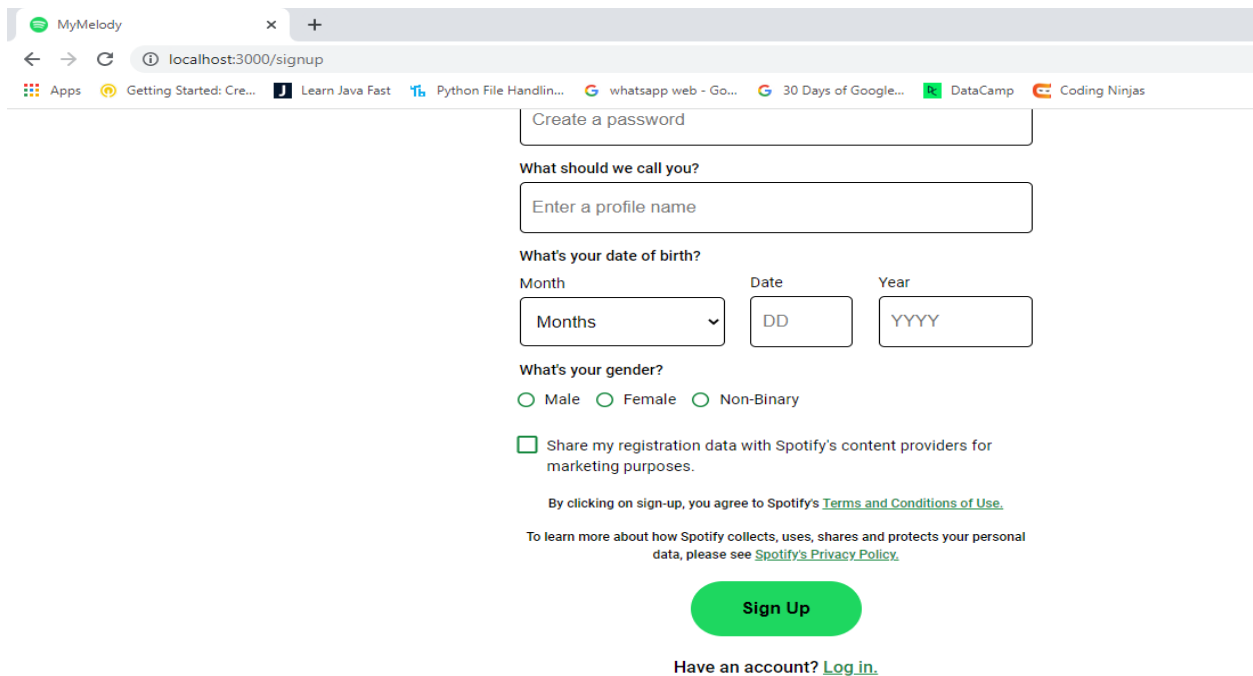


● Footer of Home Screen



● Sign Up Screen

A screenshot of the MyMelody sign-up screen. The browser's address bar shows 'localhost:3000/signup'. The page features the MyMelody logo at the top, followed by the text 'Sign up for free to start listening.' Below this is a blue button labeled 'Sign up with Facebook'. A horizontal line with the word 'or' in the center separates this from the email sign-up section. The email sign-up section is titled 'Sign up with your email address' and contains three input fields: 'What's your email?' (placeholder: 'Enter your email'), 'Create a password' (placeholder: 'Create a password'), and 'What should we call you?' (placeholder: 'Enter a profile name').



MyMelody

localhost:3000/signup

Create a password

What should we call you?

Enter a profile name

What's your date of birth?

Month: Months, Date: DD, Year: YYYY

What's your gender?

☐ Male ☐ Female ☐ Non-Binary

☐ Share my registration data with Spotify's content providers for marketing purposes.

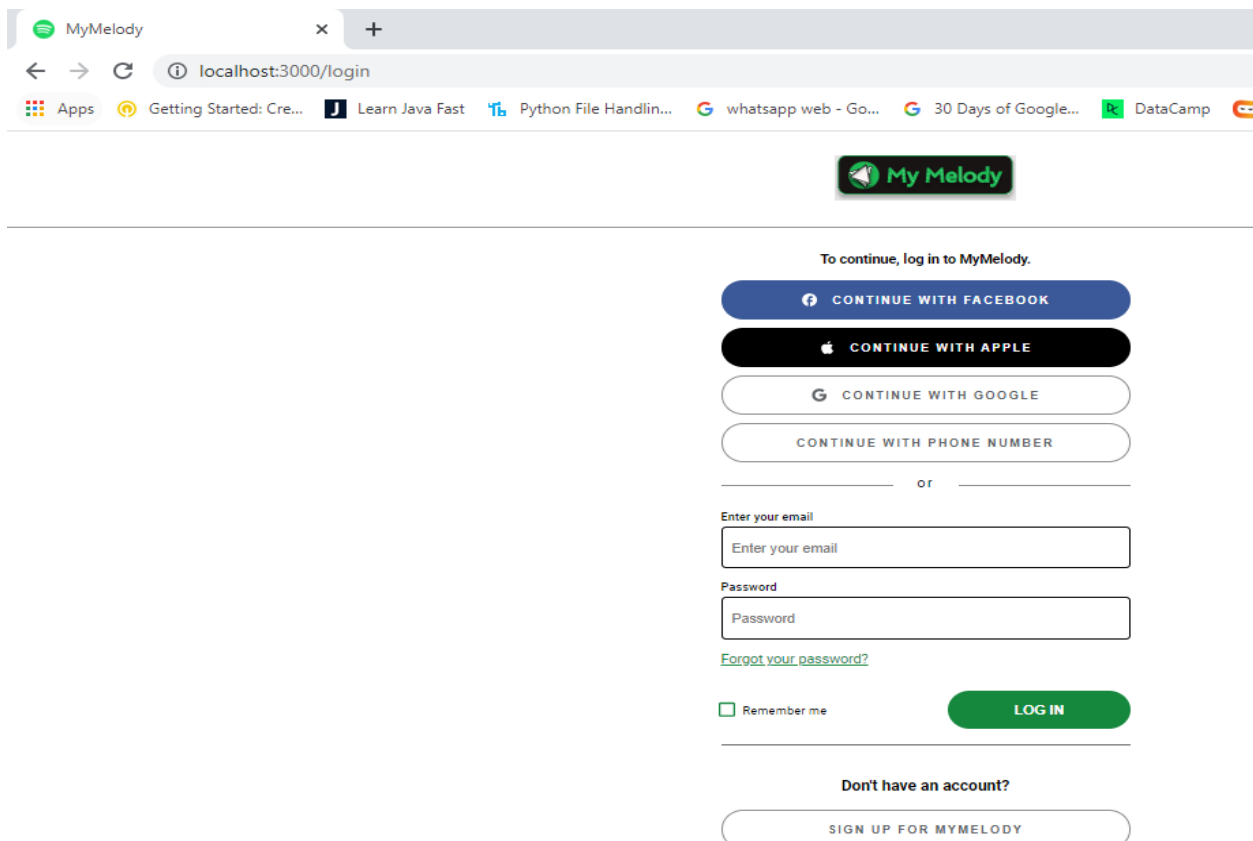
By clicking on sign-up, you agree to Spotify's [Terms and Conditions of Use](#).

To learn more about how Spotify collects, uses, shares and protects your personal data, please see [Spotify's Privacy Policy](#).

Sign Up

Have an account? [Log in](#).

● Log In Screen



MyMelody

localhost:3000/login

My Melody

To continue, log in to MyMelody.

CONTINUE WITH FACEBOOK

CONTINUE WITH APPLE

CONTINUE WITH GOOGLE

CONTINUE WITH PHONE NUMBER

or

Enter your email

Enter your email

Password

Password

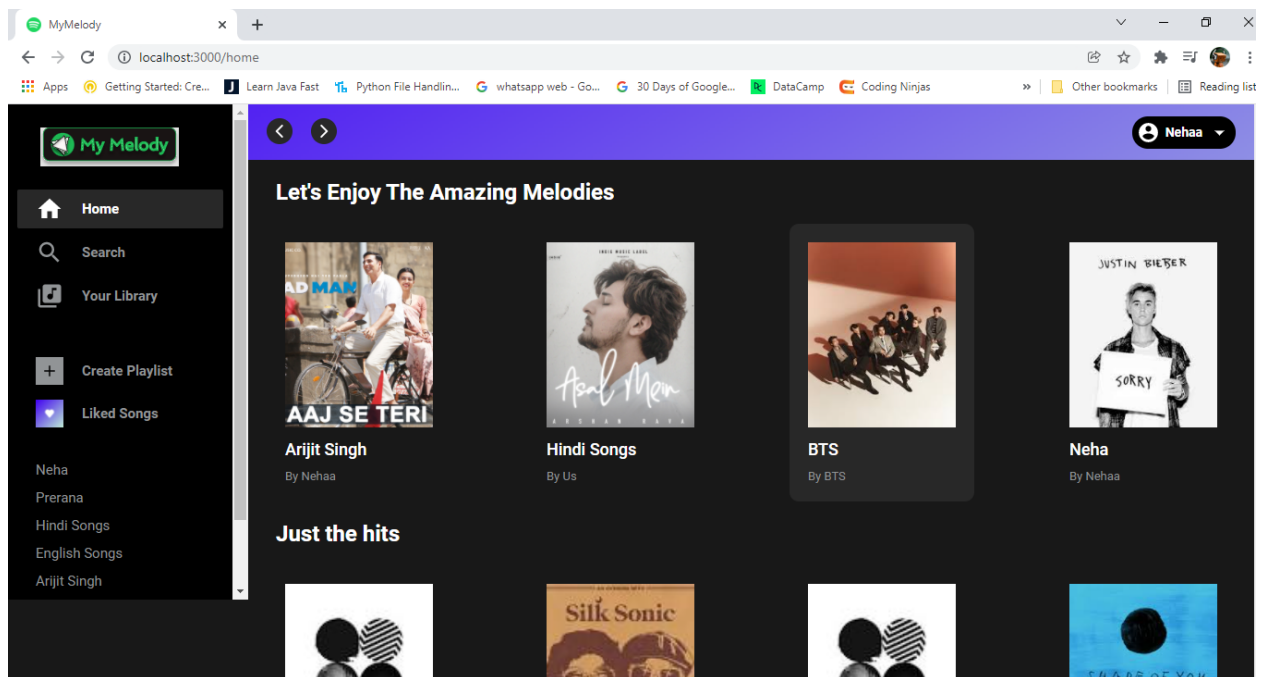
[Forgot your password?](#)

☐ Remember me **LOG IN**

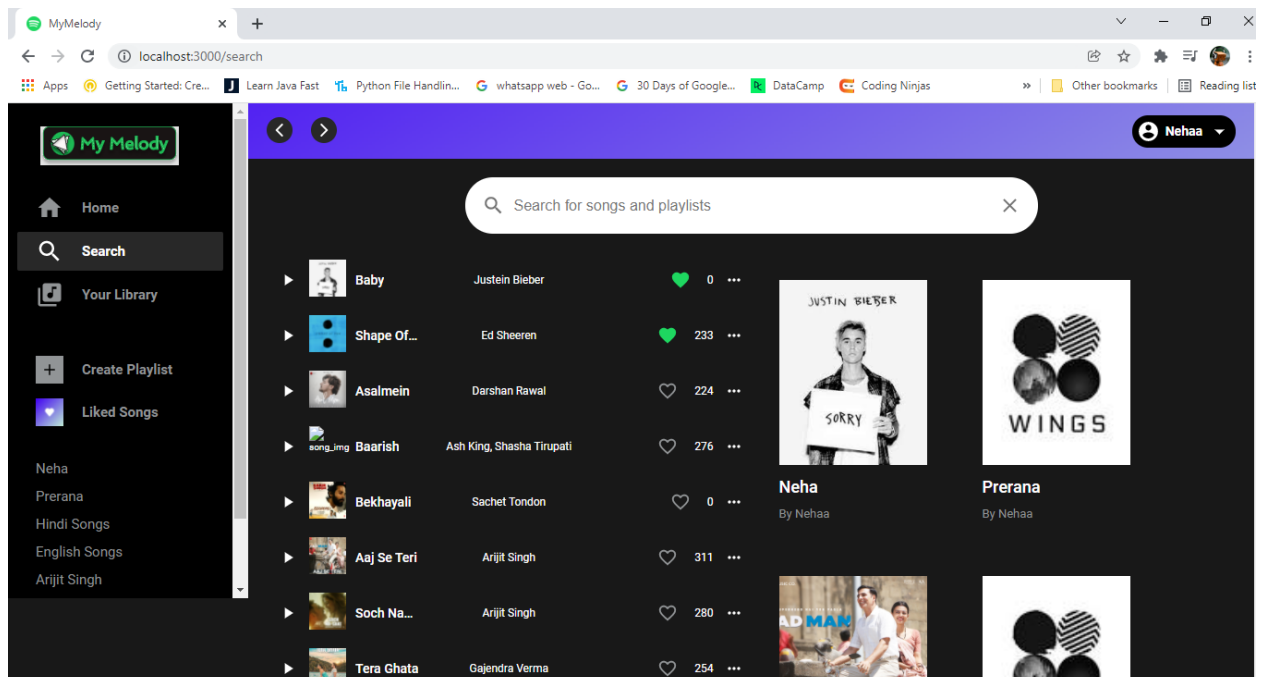
Don't have an account?

SIGN UP FOR MYMELODY

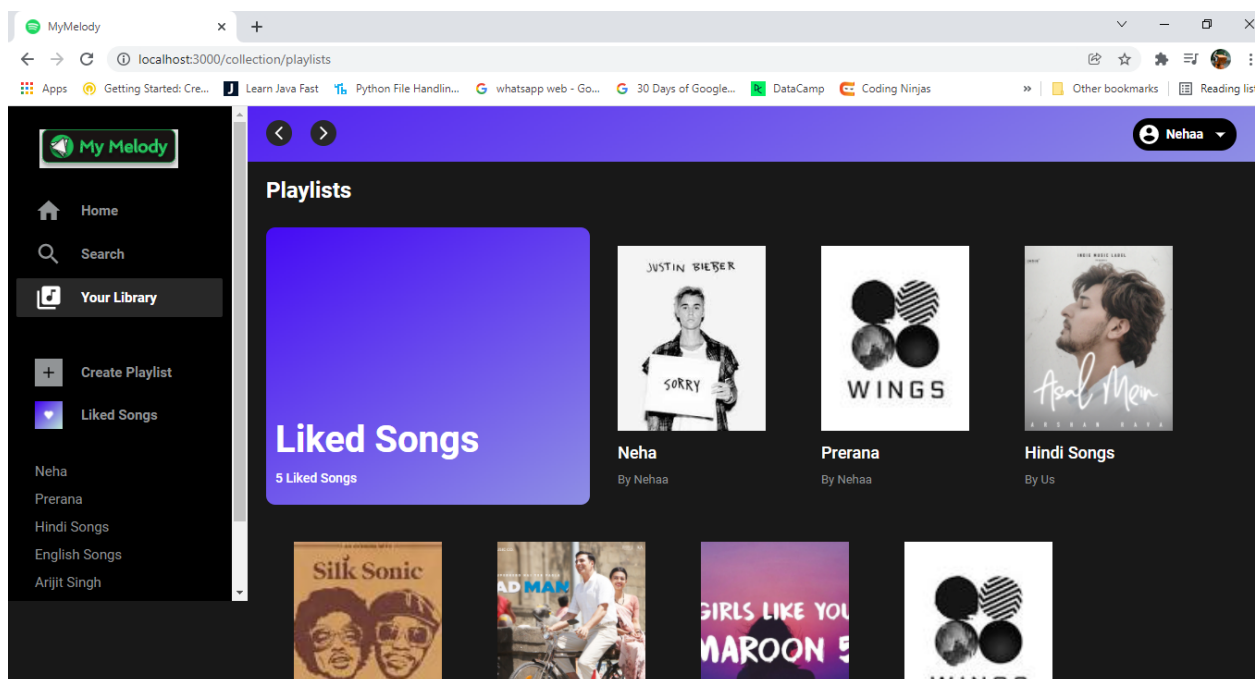
• Main Screen of music website



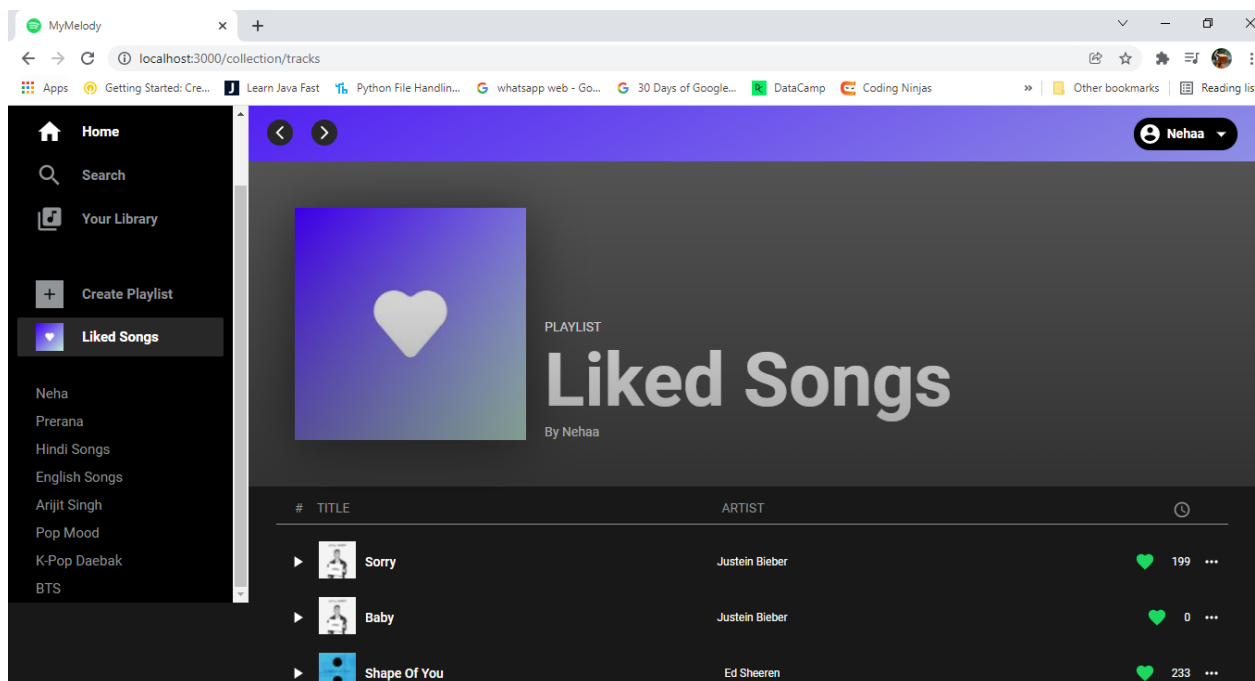
• Search Screen



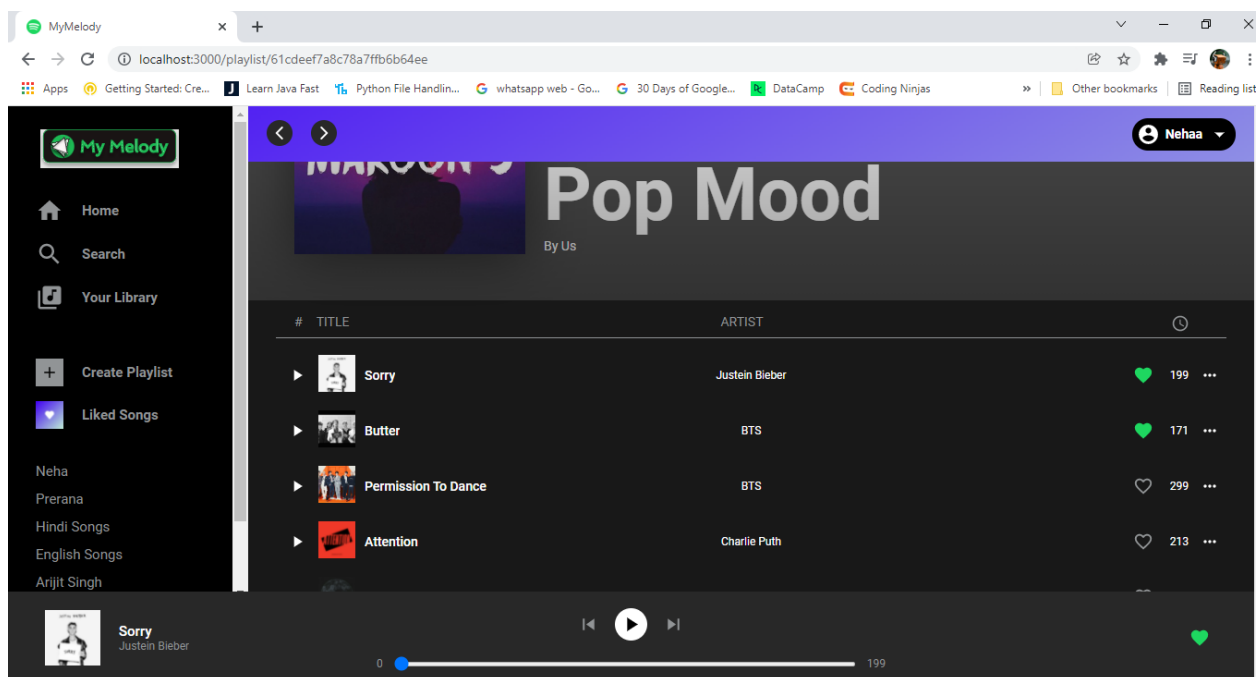
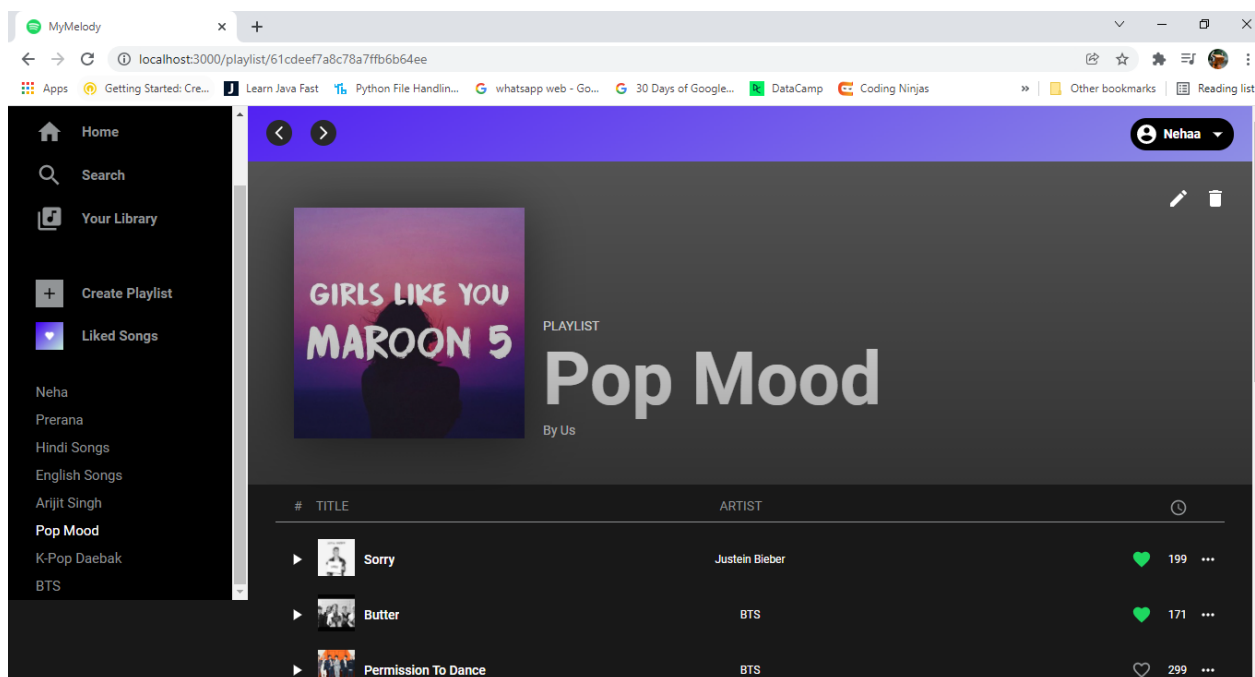
• Your Library Screen



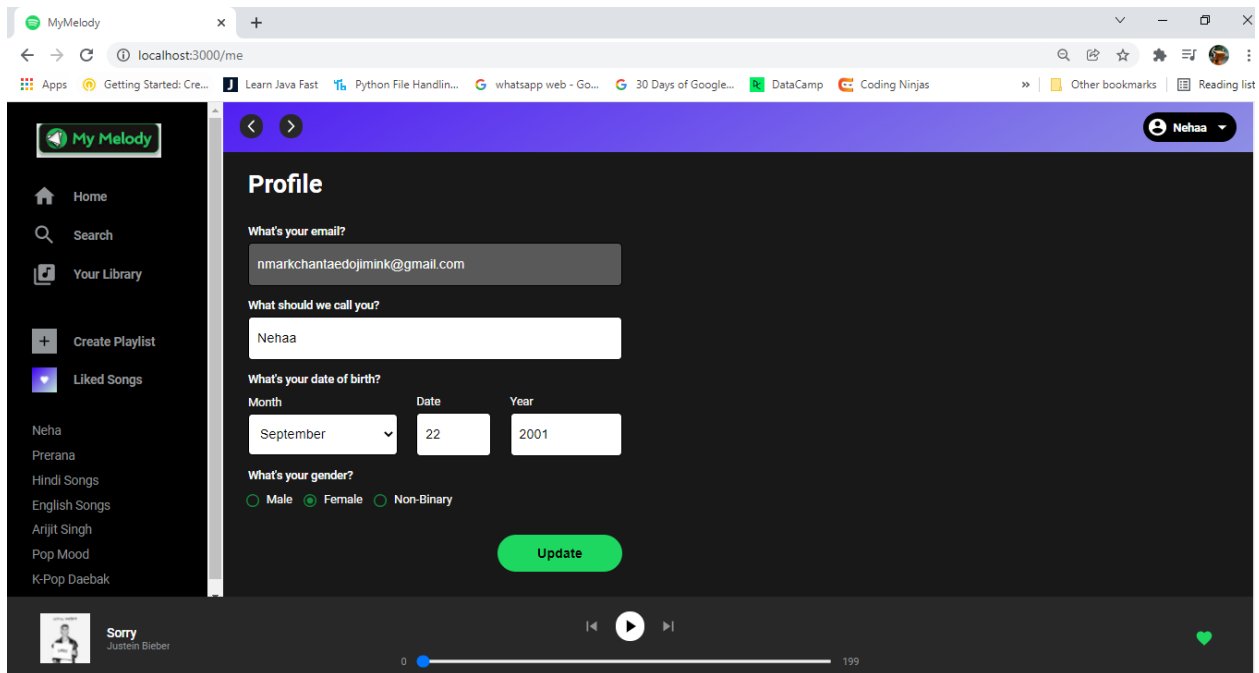
• Liked Songs Screen



• Playlist Screen



● Profile Screen



Future Activity

- Improving the user interface so that it is more user-friendly. Adding as many features as possible and improving the functionality of the existing features of the website.
- Creating a system which will have the equivalent functionality and usability to the systems available in the market.

Conclusion

- Hence we found solutions to implement the suggestions provided and studied the technology that is to be used to implement the solution.
- Created models, views and controllers to implement the MVC architecture.
- In the previous version of the project, we were able to complete the majority of the system that we wished to have from the beginning. Our original goals for this project include creating a music streaming site, having the ability to login and

register with validation, create album and artist pages, and displaying a list of each on their respective pages, be able to repeat and shuffle songs, and have ability to create and edit playlists.

- We are trying to add suggested and new features in the existing website developed by us.
- We want to have all the suggested and optimized features available before we add this project to my portfolio.
- We wish to submit a good version of the website by implementing the suggested, new and unique features as optimally as possible.

References

<https://www.w3schools.com/>

<https://www.mongodb.com/languages/mern-stack-tutorial>

<https://expressjs.com/>