Project Report on Music Website

Github Link:
https://github.com/soumyo2002/Music-Website

Deployment Link:
https://music-website-fhxd.onrender.com/

Introduction:

The Music Website is a web application developed using HTML, CSS, JavaScript, Node.js, Express.js, and MongoDB. It aims to provide a platform for music lovers to access and listen to their favorite music. The application provides a user-friendly interface that is easy to navigate.

Project Objectives:

The main objectives of the project are as follows:

- To provide a platform where users can listen to music
- To allow users to request new songs
- To make the platform user-friendly and easy to navigate

Features:

The Music Website has several features, including:

Home Page: The Home page of the website displays the latest albums and songs.

Search Functionality: The website provides a search bar where users can enter the name of the song they want to listen to.

Music Player: The website has a built-in music player that allows users to listen to music directly on the website. The player has standard functions such as play, pause, next, previous, and shuffle.

Request a Music Feature:The website provides a feature where users can request a particular song or album that is not currently available on the website. Users can enter the name of the song they want to request, and the website administrator can review the requests and add the requested music to the website. This feature is available on the home page, and users can access it by clicking on the "Request a Song" button.

Download: Users can download their favourite songs
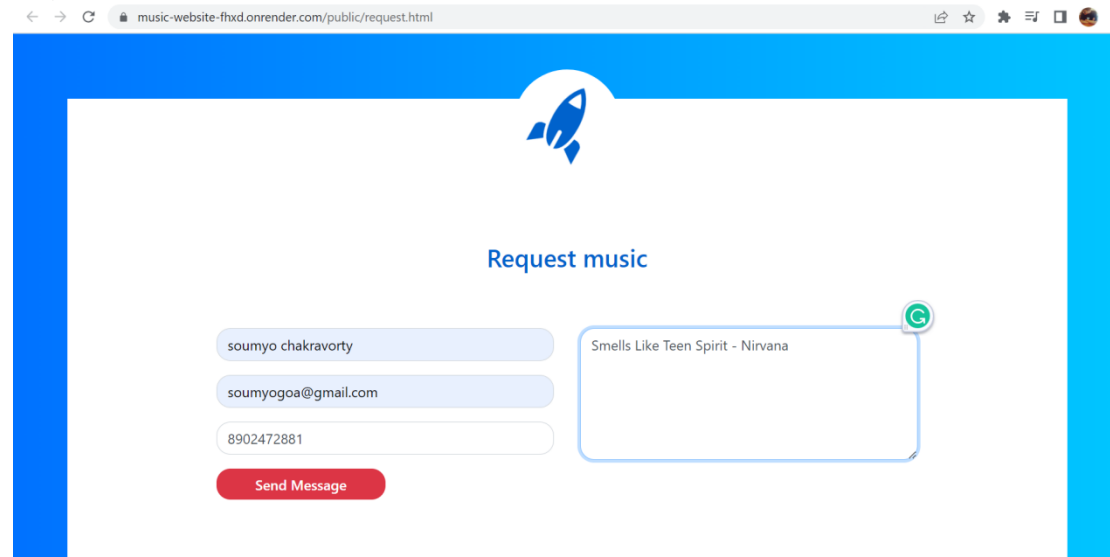
Home screen:
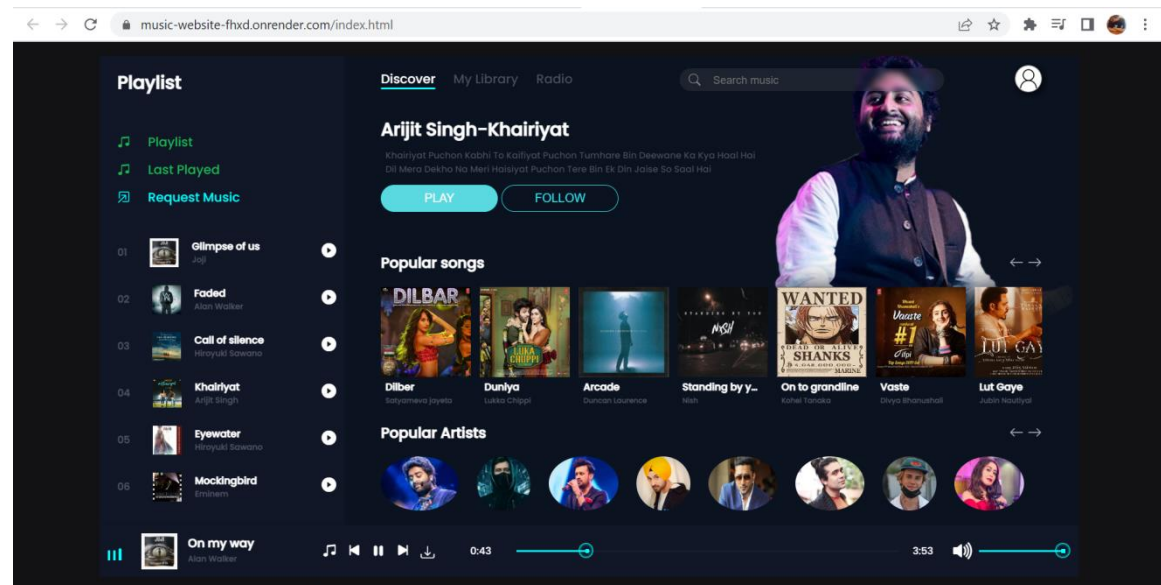
Architecture:
The website is designed using a client-server architecture, where the client (user) interacts with the server (website) through HTTP requests. The server processes the requests and sends the appropriate response back to the client.

FLOW OF BACKEND CODE:

1. The server is running node app.js command in the terminal by Render platform
2. The express module is imported and assigned to the app constant.
3. The body-parser middleware is used to parse incoming request bodies in a middleware before the handlers, and is assigned to app.use(bodyparser.json()) and app.use(bodyparser.urlencoded({extended:true})).
4. The mongoose module is imported and connected to the MongoDB Atlas cluster using mongoose.connect(), with the credentials and database details provided in the connection string.
5. A callback function is set to be called when the connection to the MongoDB Atlas cluster is established, which prints "Connected to Database" to the console.
6. An error handler function is set to be called when there is an error connecting to the MongoDB Atlas cluster, which prints "Error in Connecting to Database" to the console.
7. The app.use() function is used to serve static files located in the /css directory, making them publicly available.
8. The app.get() function is used to handle GET requests to the root directory of the server. It sets the response headers to allow cross-origin resource sharing (CORS) and redirects the request to the index.html file using res.redirect().

9. The app.get() function is used to handle GET requests to the request.html file, sending the file to the client using res.sendFile().
10. The app.post() function is used to handle POST requests to the request.html file. It retrieves the data from the request body and stores it in a data object. It then inserts the data into the users collection in the MongoDB Atlas cluster using db.collection().insertOne(). If an error occurs during insertion, it is thrown. If successful, "Record Inserted Successfully" is printed to the console.
11. The function then redirects the client to the request_success.html file using res.redirect().
12. The app.listen() function is used to listen for incoming connections on the specified port number. When the server starts listening on the port, "Listening on port {port}" is printed to the console.
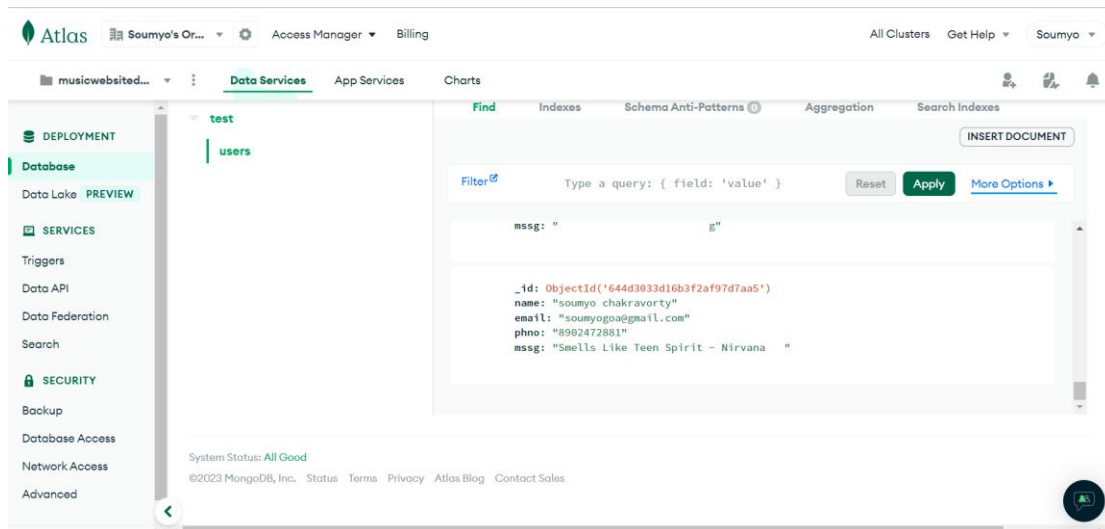
Request for music site:

Request success page:
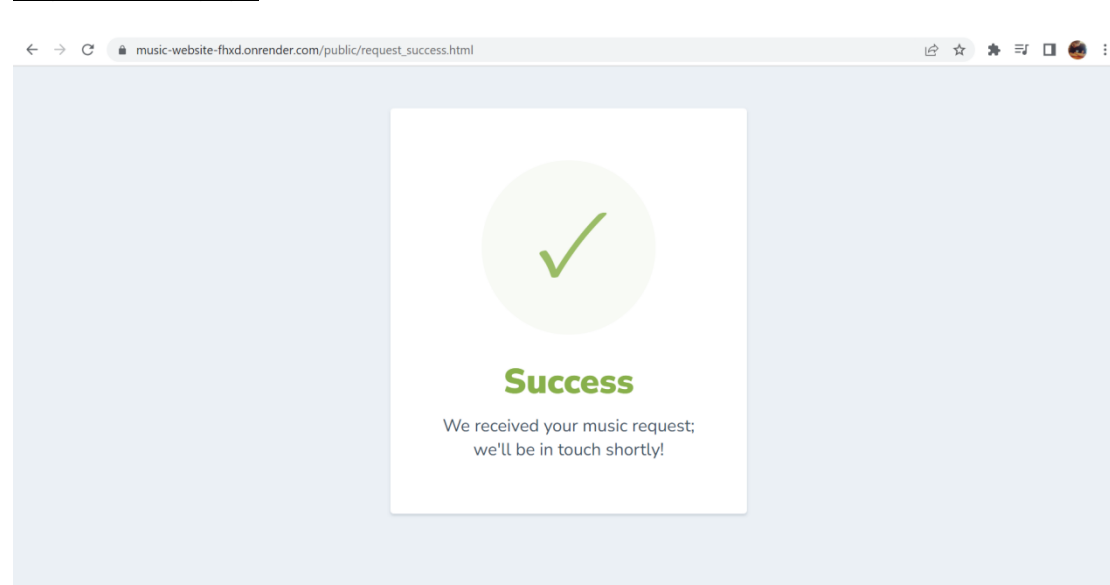
MongoDB Databse:

Technologies Used:

The Music Website was developed using the following technologies:

HTML: HTML: HTML was used to create the structure of the website. It provides the backbone for the website and defines the layout and content of each page.

CSS: CSS was used to style the website and make it visually appealing. It provides the colors, fonts, and overall look and feel of the website.

JavaScript: JavaScript was used to add functionality to the website, including the music player, search functionality, and request feature. It provides the interactivity and dynamic behavior of the website.

Node.js: Node.js is a JavaScript runtime environment used to build the backend of the application. It provides a server-side environment that allows for efficient and scalable backend operations.

Express.js: Express.js is a Node.js web application framework used to create RESTful APIs for the application. It provides a simple and flexible way to create web applications and APIs.

MongoDB: MongoDB is a NoSQL database used to store and manage data for the application. It provides a scalable and efficient way to store and manage data.

Limitations
The Music-Website has the following limitations:

- Limited Song Collection: The website currently has a limited collection of songs, which may not satisfy all users' music preferences.
- No User Accounts: The website does not have a user account system, which limits the functionality that can be provided to users.
- No Ratings or Feedback: The website does not have a rating or feedback system, which limits user engagement and feedback.

In future updates,Work will be done to overcome these limitations along with migrating the website to cloud

Deployment:

The Music Website was deployed using Render, a cloud platform that provides hosting and deployment services. The website can be accessed at https://music-website-fhxd.onrender.com/index.html

Conclusion:

The Music Website provides a platform for music lovers to access and listen to their favorite music. Its user-friendly interface, search functionality, and music player make it easy to navigate and enjoy music. The use of Node.js, Express.js, and MongoDB allows for scalable and efficient backend operations. The addition of the request feature allows users to suggest and request new music, making the website more user-driven. Overall, the Music Website is a great application for music enthusiasts.