Internship Report

Web Development(MERN Stack)

DLithe Consultancy Services Pvt. Ltd.

Internship Report

Trainee/Intern Name: Srinidhi Shetty

Reg. no: U72900KA2019PTC121035

Period: 95 Days

Job Assignment: Music App Using MERN

Organization: DLithe Consultancy Services Pvt. Ltd.

Supervisor's Name: Purushottam

Observations:

- **Efficient API Integration:** The project demonstrates seamless integration with external music APIs, ensuring accurate and real-time song data retrieval without the need for maintaining an internal music database.
- **Modular and Scalable Design:** The separation of concerns across frontend (React), backend (Node.js/Express), and database (MongoDB) enables modular development.
- **User-Centric Interface:** The React-based frontend ensures a responsive and interactive user experience, allowing smooth navigation, real-time music search, and playback functionalities.

Submitted to

Signature of Training Supervisor

Date: 5/05/25

Signature of Co-ordinator

Date: 5/05/25

DLithe™
Be Agile..Be in Demand

Letter of Transmittal

To,

Program Co-ordinator
DLithe Consultancy services
Bengaluru

Dear Sir,

I am writing to submit my report on the web development internship, during which I had the opportunity to work on building a Music Streaming Web Application using the MERN (MongoDB, Express.js, React.js, Node.js) stack. This internship proved to be an enriching and hands-on learning experience that allowed me to apply academic knowledge in a real-world development environment.

Throughout the training, I gained practical exposure to full-stack web development, including frontend UI design, backend API creation, database management, and third-party API integration. The project also involved the use of modern tools like Visual Studio Code, Jira, Git, and Figma for design, collaboration, and version control. I developed a strong understanding of how frontend and backend components interact to deliver a seamless user experience.

The music application included features such as user authentication, music search, media playback, and navigation between pages. It also involved integrating external music APIs to fetch dynamic data. These tasks not only helped enhance my coding skills but also improved my problem-solving, debugging, and design capabilities.

This report provides a detailed overview of the internship program, covering the technologies used, tasks performed, and the outcomes achieved. It also reflects on the relevance and importance of full-stack development skills in today's rapidly evolving tech industry.

I believe the skills acquired during this internship will prove valuable to future projects and professional development.

Sincerely,

Name: Srinidhi Shetty

DLithe™
Be Agile..Be in Demand

Table of Contents

1. INTRODUCTION

As part of my internship at Dlithe Consultancy Services Pvt. Ltd., I was assigned to design and develop a Music Streaming Web Application using the MERN stack (MongoDB, Express.js, React.js, and Node.js).

The core objective of this project was to design and develop a dynamic, full-featured music streaming web application named *Audio Oasis*. The platform allows users to search for songs, play or pause tracks, and explore different sections of the application through a clean and responsive interface. With an emphasis on user experience, features such as favorites, playlists, recently played tracks, and a dedicated now-playing screen were incorporated to closely mimic the functionality of commercial music services.

This project leverages local storage for music files, enabling fast playback and eliminating dependency on third-party APIs or cloud storage services for audio streaming. Despite using local storage, the application architecture is designed in a modular way, which can easily be extended to integrate external music APIs or cloud hosting in the future.

User authentication is implemented to provide a personalized experience, allowing users to maintain their music library, track preferences, and preserve session states. The intuitive navigation and minimalistic design enhance usability across devices, while the music player provides essential controls like play/pause, seek, shuffle, repeat, and volume adjustment.

Built using the MERN stack—MongoDB, Express.js, React.js, and Node.js—the project benefits from a unified JavaScript environment across both frontend and backend. This consistency accelerates development, simplifies debugging, and ensures that data flows

smoothly between components. The modular design also promotes scalability, enabling new features to be integrated with minimal disruption to existing functionality.

Overall, *Audio Oasis* serves as a practical demonstration of full-stack development capabilities, combining aesthetics with robust backend logic to deliver a complete multimedia web application.

2. LITERATURE SURVEY

A thorough review of research papers, development articles, and technical documentation was conducted to understand the current landscape of web-based music streaming platforms and their underlying technologies. The aim was to identify best practices, methodologies, and technologies relevant to building a performant, scalable, and user-centric application. These studies highlighted several key areas that directly influenced the design and development of *Audio Oasis*:

1. **Single Page Applications (SPAs):**
   SPAs enhance user experience by dynamically updating content without requiring full page reloads. React.js, with its virtual DOM and reusable component architecture, emerged as a preferred choice for building SPAs. In *Audio Oasis*, this enabled smooth transitions between views such as Home, Library, and Playlist without reloading the entire interface.

2. **MERN Stack Integration:**
   The combination of MongoDB, Express.js, React.js, and Node.js offers a robust full-stack JavaScript environment. This unified stack reduces development complexity by using a single programming language across the application, thereby improving productivity and maintaining consistency in data handling and logic.

3. **Use of MongoDB for Media Metadata:**
   MongoDB's schema-less, document-oriented structure is ideal for storing dynamic content like user preferences, song metadata, and playlist structures. Its ability to handle nested documents and relationships helped in efficiently managing user-specific content in the application.

4. **Node.js and Express.js for Backend APIs:**
   Node.js supports asynchronous operations and non-blocking I/O, making it highly suitable for applications involving frequent data transfer, such as media streaming. Express.js was used to set up RESTful APIs for handling user requests, media file serving, and interaction with the database.

5. **Local Media Storage vs. External APIs:**
   While APIs from providers like Spotify, Deezer, and Last.fm are commonly used for real-time music data and playback, *Audio Oasis* takes a different approach by storing music files locally. This allows greater control over the content, avoids API rate limitations, and enables offline hosting scenarios.

6. **User Authentication using JSON Web Tokens (JWT):**
   JWT provides a stateless and secure method for managing user authentication and session control. It ensures that authenticated users can access personalized features like favorites, recently played tracks, and custom playlists without compromising security.

7. **HTML5 Audio API for Playback Control:**
   The native HTML5 Audio API was leveraged to build a responsive audio player supporting essential features like play, pause, seek, skip, repeat, and volume control. This API ensures compatibility across major browsers and offers fine-grained control over audio events.

8. **Responsive Design with Flexbox and Grid:**
   To ensure accessibility across devices, *Audio Oasis* was designed using responsive layout techniques. Tailwind CSS, along with CSS Flexbox and Grid, was used to build adaptive components that seamlessly adjust to different screen sizes.

9. **Frontend Routing with React Router:**
   React Router enables efficient client-side routing, allowing the application to manage different views such as Home, Library, Search, Login, and Playlist without reloading the page. This contributes to a faster and smoother user experience.

10. **Design Prototyping with Figma:**
    UI/UX design was initially planned and visualized using Figma. Prototypes were created to outline the overall layout, navigation flow, and visual theme before implementation. This helped maintain design consistency and streamline the development workflow.

3. PROPOSED WORK

The proposed project, *Audio Oasis*, is a full-stack web application designed to provide users with a seamless music streaming experience using local audio storage. The application is developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack, which enables efficient and scalable development entirely in JavaScript. The core idea is to simulate the functionality of popular music platforms like Spotify, but with the simplicity and flexibility of local media management.

The work is divided into the following key modules:

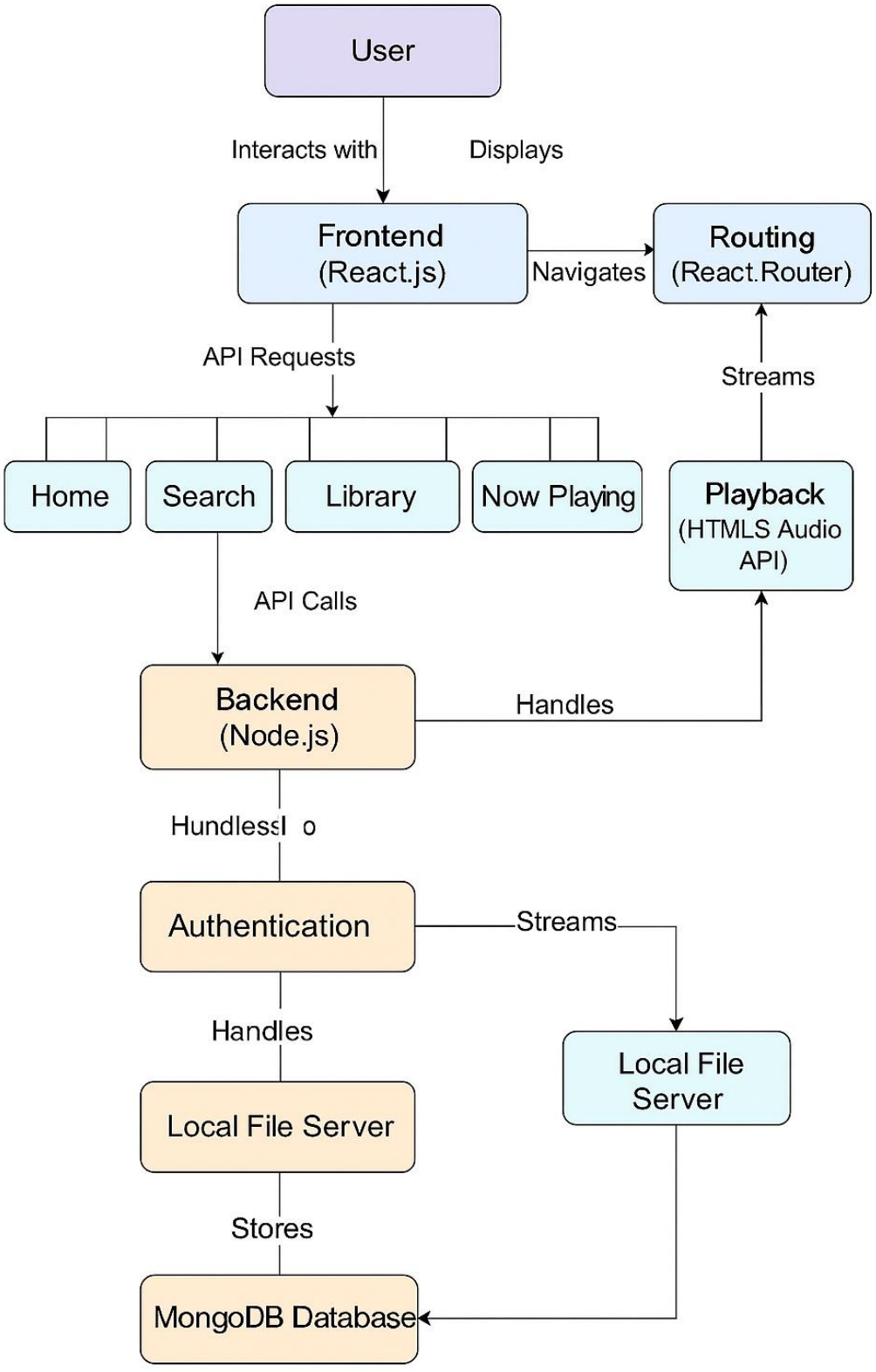
1. **User Interface Development (Frontend – React.js):**
   A responsive and interactive user interface is developed using React.js and Tailwind CSS. The frontend includes views for Home, Library, Search, Login/Register, and Now Playing. The design follows modern UI/UX principles for intuitive navigation and smooth user experience.

2. **User Authentication:**
   Secure login and registration functionality is implemented using JWT-based authentication. Users can register, log in, and maintain personalized sessions to access their favorites, playlists, and recently played tracks.
3. **Media Management and Local Playback:**
   Audio files are stored locally on the server and served to the frontend via secure routes. The HTML5 Audio API is used to control playback on the client side, offering features such as play, pause, seek, skip, and volume control.
4. **Backend API Development (Node.js + Express.js):**
   RESTful APIs are developed using Express.js to handle all data transactions including user management, song metadata handling, and playback history. These APIs serve as the communication bridge between frontend requests and backend operations.
5. **Database Integration (MongoDB):**
   MongoDB is used to store user data, song metadata, playlists, favorites, and playback history. Its flexible schema is ideal for handling nested and dynamic data associated with user interactions.
6. **Routing and Navigation:**
   React Router is used to enable smooth client-side routing between different pages of the application, maintaining the SPA (Single Page Application) behavior for a fast and responsive experience.
7. **Responsive Design and Cross-Device Compatibility:**
   The application is built with a mobile-first approach, ensuring that it performs well across desktops, tablets, and smartphones. Flexbox and Grid layout systems are used to create a flexible design that adapts to various screen sizes.
8. **Playlist and Favorites Functionality:**
   Users can create and manage their playlists, mark songs as favorites, and view recently played songs. This provides a more personalized and engaging experience.
9. **Future Scalability Considerations:**
   Although the current version uses locally stored music, the system is designed in a

modular fashion to allow future integration with cloud storage or third-party music APIs, enhancing scalability and real-time music access.

4. FLOW DIAGRAM – Description

The overall flow of the *Audio Oasis* application can be divided into several major stages that depict the interactions between the user, frontend, backend, and database:


User
Interacts with
Displays
Frontend
(React.js)
Navigates
Routing
(React.Router)
API Requests
Streams
Home
Search
Library
Now Playing
Playback
(HTMLS Audio
API)
API Calls
Backend
(Node.js)
Handles
Hundless
o
Authentication
Streams
Handles
Local File
Server
Local File Server
Stores
MongoDB Database

1 **User Interaction Layer (Frontend – React.js):**

- Users interact with the application through a responsive UI.
- Pages include: Home, Search, Library, Now Playing, Login/Register, and Playlists.
- Actions like login, search, and playback are initiated here.

2 **Frontend Routing:**

- React Router manages navigation between views without full page reload.
- UI state (e.g., playback status, selected track) is handled using React hooks or context.

3 **Backend Communication (Express.js APIs):**

- API calls are made to Node.js backend for authentication, fetching song metadata, user actions (favorites, playlists), and serving audio files.
- Uses RESTful endpoints for structured communication.

4 **Authentication Module:**

- Handles login and registration.
- Verifies user credentials and returns a JWT token for session management.

5 **Audio Management:**

- Frontend uses the HTML5 Audio API to stream music from local files.
- Playback controls (play, pause, skip, repeat) are handled on the client side.

6 **MongoDB Database:**

- Stores user credentials, playlists, favorites, and playback history.
- Metadata for music files (e.g., title, artist, duration) is also stored here.

7 **Local File Server (Node.js):**

- Music files are stored locally and served to the frontend via secured routes.
- Node handles file streaming efficiently using built-in modules.

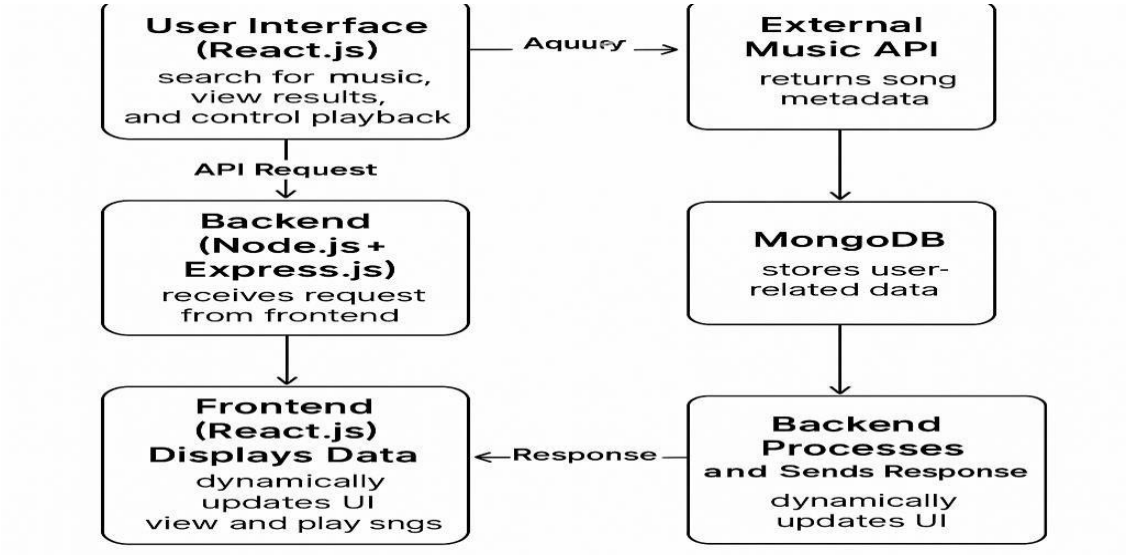
5. USE CASE/ TEST CASE – Audio Oasis

Use Case ID	Use Case	Description	Input	Expected Output	Status
UC01	User Registration	Register a new user with email	Name, Email, Password	Account created, redirected to login page	Pass

		and password.			
UC02	User Login	Authenticate user using credentials.	Email, Password	Redirected to dashboard	Pass
UC03	View Dashboard	Display recent plays and playlists after login.	User logged in	Dashboard components load correctly	Pass
TC04	Search for a Song	Search for songs using keyword.	Song/Artist Name	List of matching songs displayed	Pass
TC05	Play a Song	Play selected song with controls.	Click on play icon	Song starts playing, player visible	Pass
TC06	Pause a Song	Pause the currently playing song.	Click on pause icon	Song pauses at current position	Pass
TC07	Add Song to Favorites	Mark a song as favorite.	Click on heart icon	Song appears in favorites list	Pass
TC08	View Playlist	View user's created playlists.	Navigate to Playlist section	List of playlists shown	Pass
TC09	Responsive Design	Test on various screen sizes (mobile, tablet, desktop).	Resize browser window	Layout adjusts and remains functional	Pass
TC10	Logout	End user session.	Click on logout	Redirected to login page	Pass
TC11	Play Queue & Skip Track	Manage current queue and skip to next song.	Click next button / view queue	Next song plays or queue displayed	Pass
TC12	Volume Control	Adjust playback volume.	Use volume slider	Volume changes accordingly	Pass

TC13	View Song Lyrics	Display lyrics for current track (if available).	Click on "Lyrics" button	Lyrics displayed	Pass

6. IMPLEMENTATION



The development of the Audio Oasis application followed a clear flow between frontend, backend, and data layers using the MERN stack. Below is a breakdown of how each component contributes to the system:

1. **User Interface (React.js):**

   The application begins with a responsive and interactive frontend developed using **React.js**. Users can browse music, perform searches, play or pause audio, and interact with playlists. React's component-based architecture ensures efficient UI rendering and state management using hooks and context API.

2. **API Request from Frontend:**

   When a user performs a search (e.g., for a song or artist), the React app sends an HTTP

request using **Axios** or **Fetch API** to the backend. These requests include search parameters such as keywords, filter options, or song IDs.

3. **Backend Logic (Node.js + Express.js):**
   The backend, developed using **Node.js** and **Express.js**, receives the request and handles it through defined API routes. It validates incoming data, manages session tokens (JWT), and routes the request further to local or external data sources.
4. **Local Audio Management & Media Control:**
   Instead of streaming from a third-party provider, **audio files are stored locally** on the server. The backend uses Express middleware to serve these static files securely. File paths and metadata are managed through MongoDB and served as part of the API responses.
5. **MongoDB (Database Layer):**
   The application uses **MongoDB** for persistent storage of user data such as:
   - Registered user accounts
   - Login credentials (with hashing and JWT tokens)
   - Saved playlists and favorites
   - Playback history and preferences
6. **Backend Sends Data to Frontend:**
   After retrieving relevant media files and user data, the backend compiles a response containing track information, file URLs, and user-specific metadata. This JSON response is sent back to the React app.
7. **Dynamic UI Rendering (React.js):**
   Upon receiving the response, the React frontend dynamically renders the data on appropriate pages. Components like MusicCard, Player, or PlaylistPage update in real-time based on the incoming data. Users can interact with audio controls, navigate between views, and add tracks to favorites.
8. **Playback Handling (HTML5 Audio API):**
   The app uses the **HTML5 Audio API** for controlling playback (play, pause, seek,

volume). This native browser feature enables smooth music streaming directly from the local server.

9. **Authentication System (JWT):**
   JSON Web Tokens (JWT) are used for secure user login sessions. After a successful login, a token is issued to the client and used for verifying access to protected routes and features such as favorites or playlist management.
10. **Responsive and Cross-Platform Design:**
    The UI adapts to different screen sizes using **CSS Flexbox, Grid**, and media queries. This ensures a consistent experience across desktops, tablets, and smartphones.

7. CODE IMPLEMENTATION: SONG MANGEMENT API(BACKEND)

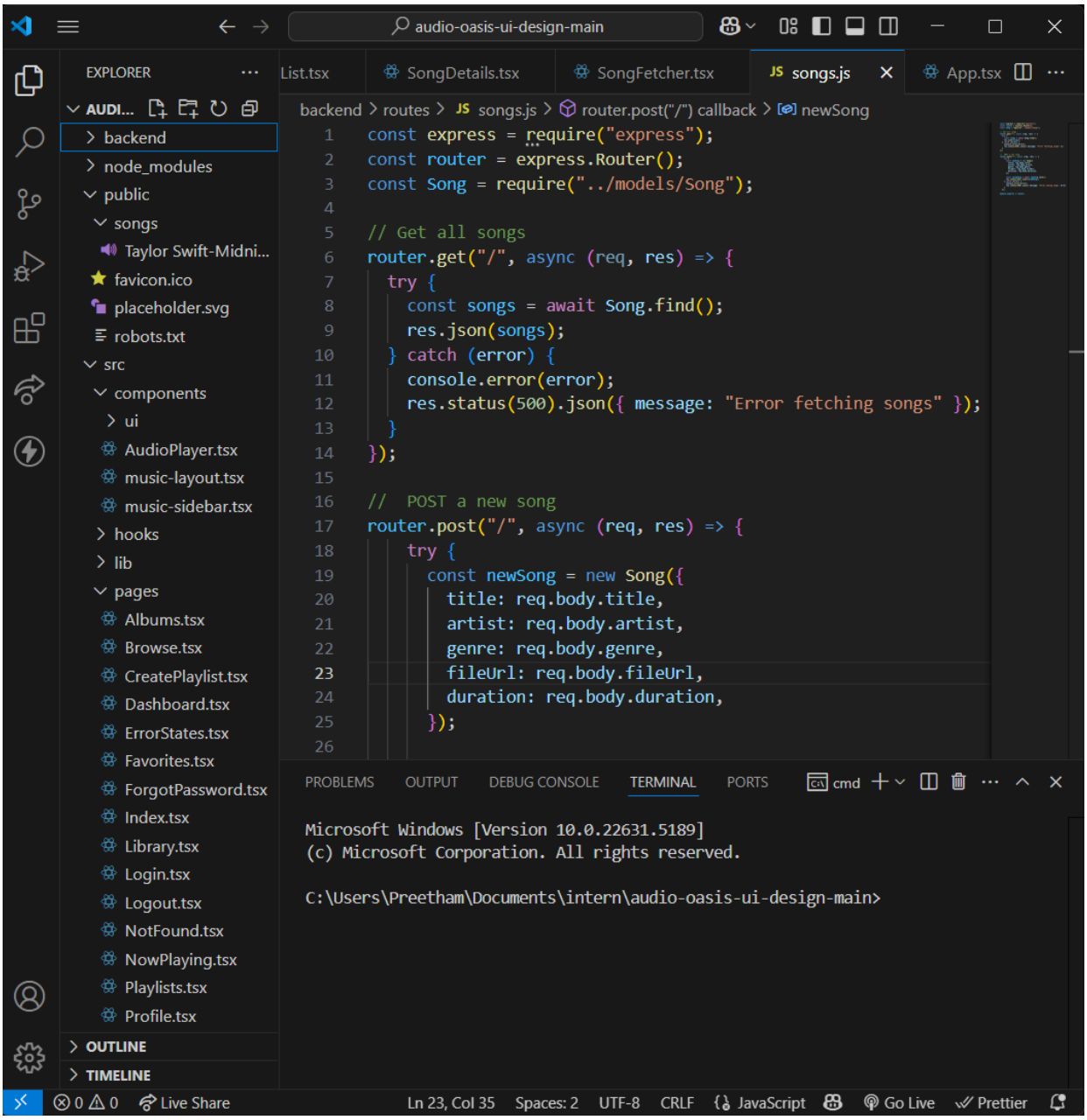
The following code defines a backend route using **Express.js** for managing songs in the *Audio Oasis* application. It connects to a MongoDB database via the Song model and includes functionality to:

- Retrieve all songs from the database
- Add a new song entry with relevant metadata (title, artist, genre, file URL, and duration)

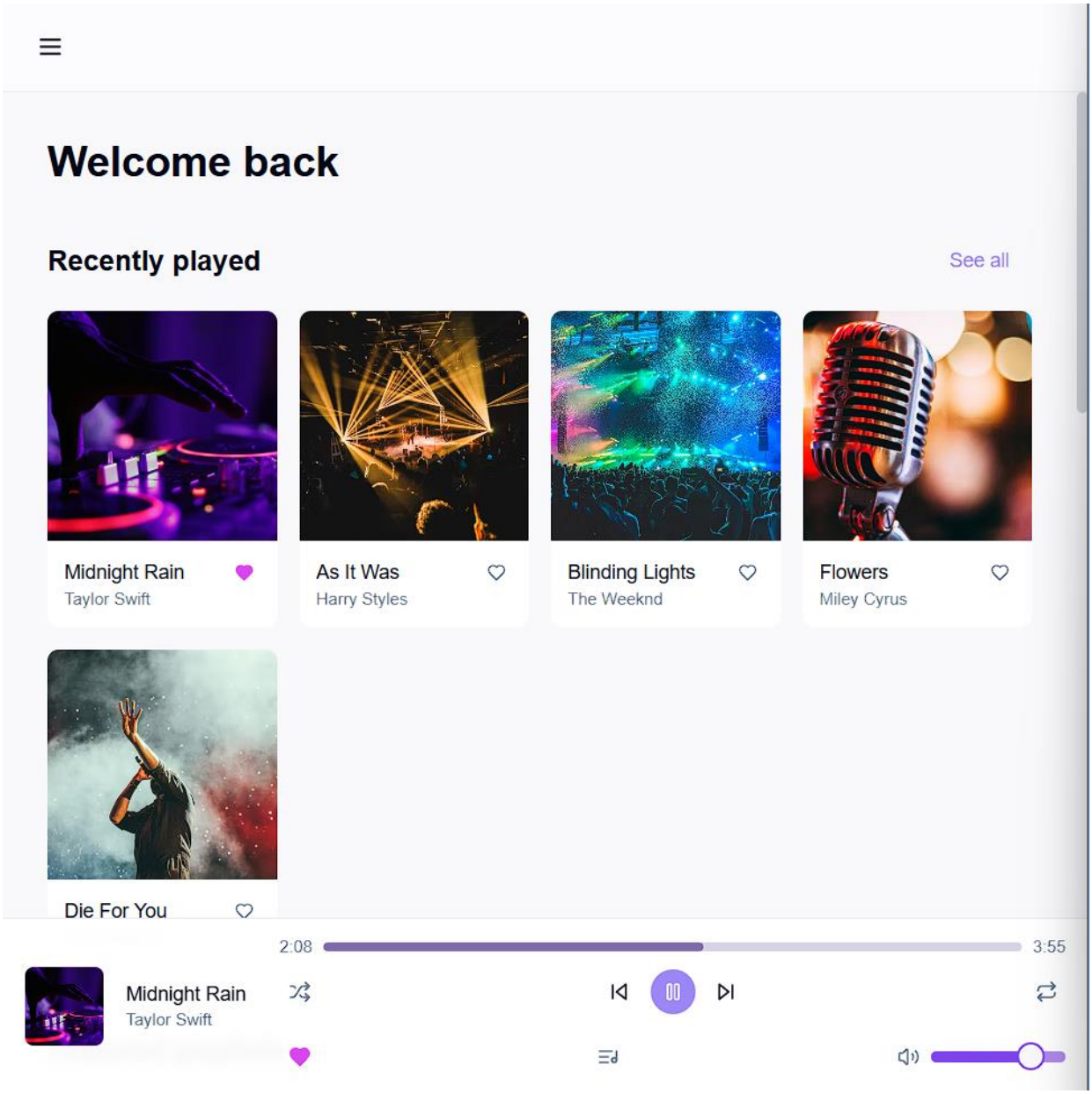
Code:

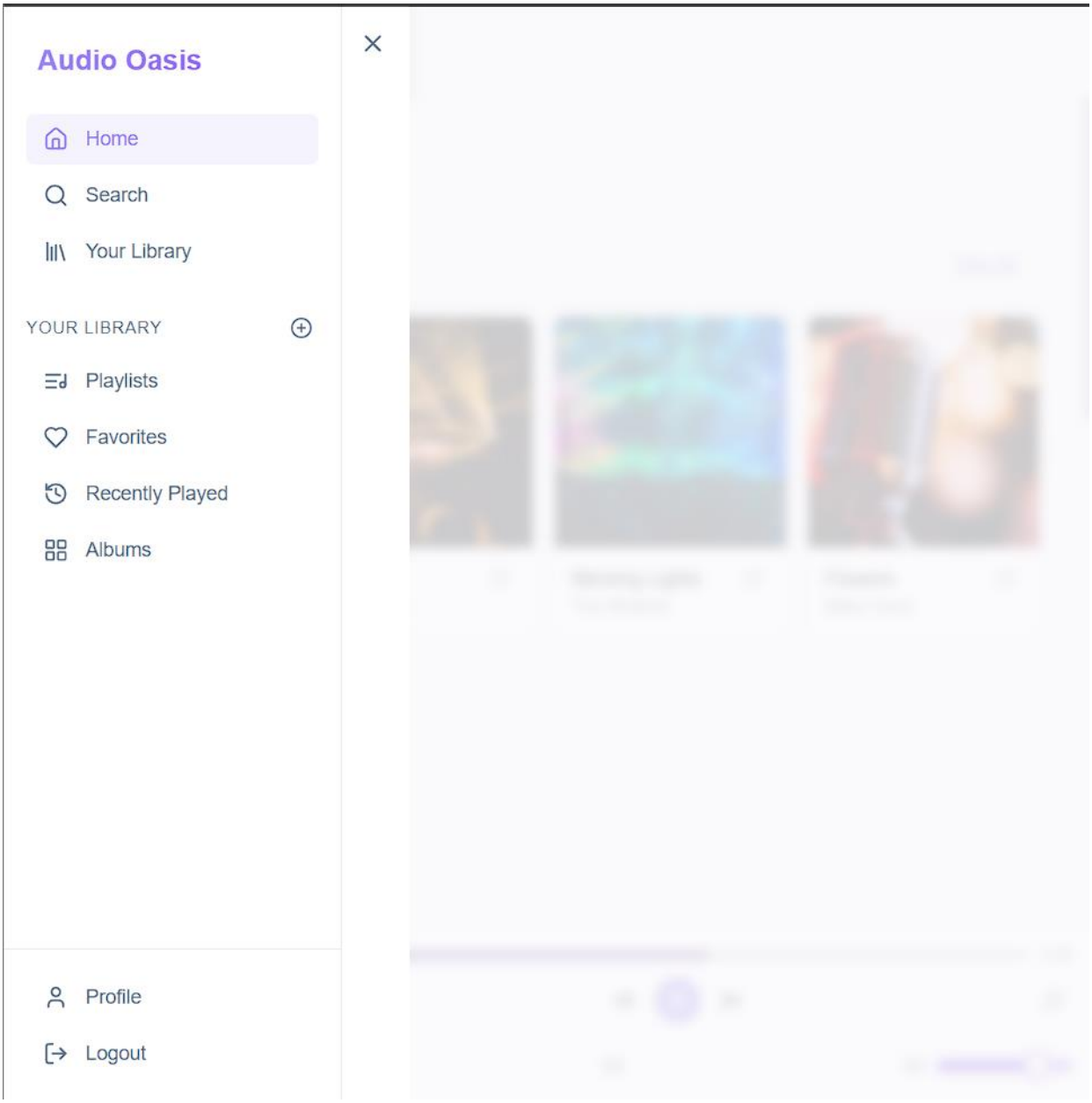
```
const express = require("express");
const router = express.Router();
const Song = require("../models/Song");

// Get all songs
router.get("/", async (req, res) => {
  try {
    const songs = await Song.find();
    res.json(songs);
  } catch (error) {
```

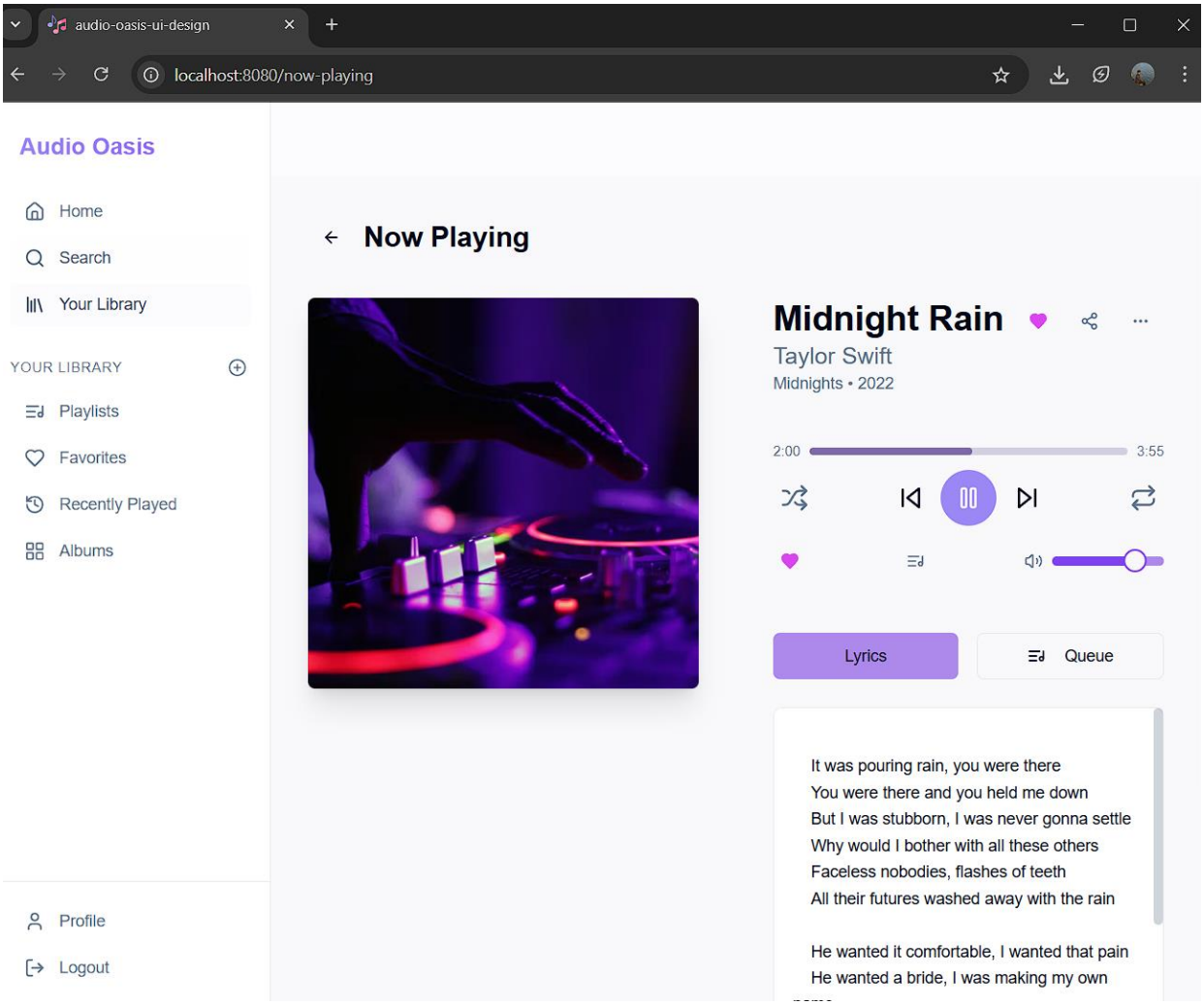
```
    console.error(error);
    res.status(500).json({ message: "Error fetching songs" });
  }
});

// POST a new song
router.post("/", async (req, res) => {
  try {
    const newSong = new Song({
      title: req.body.title,
      artist: req.body.artist,
      genre: req.body.genre,
      fileUrl: req.body.fileUrl,
      duration: req.body.duration,
    });

    const savedSong = await newSong.save();
    res.status(201).json(savedSong);
  } catch (error) {
    console.error(error);
    res.status(400).json({ message: "Error saving song", error });
  }
});

module.exports = router;
```

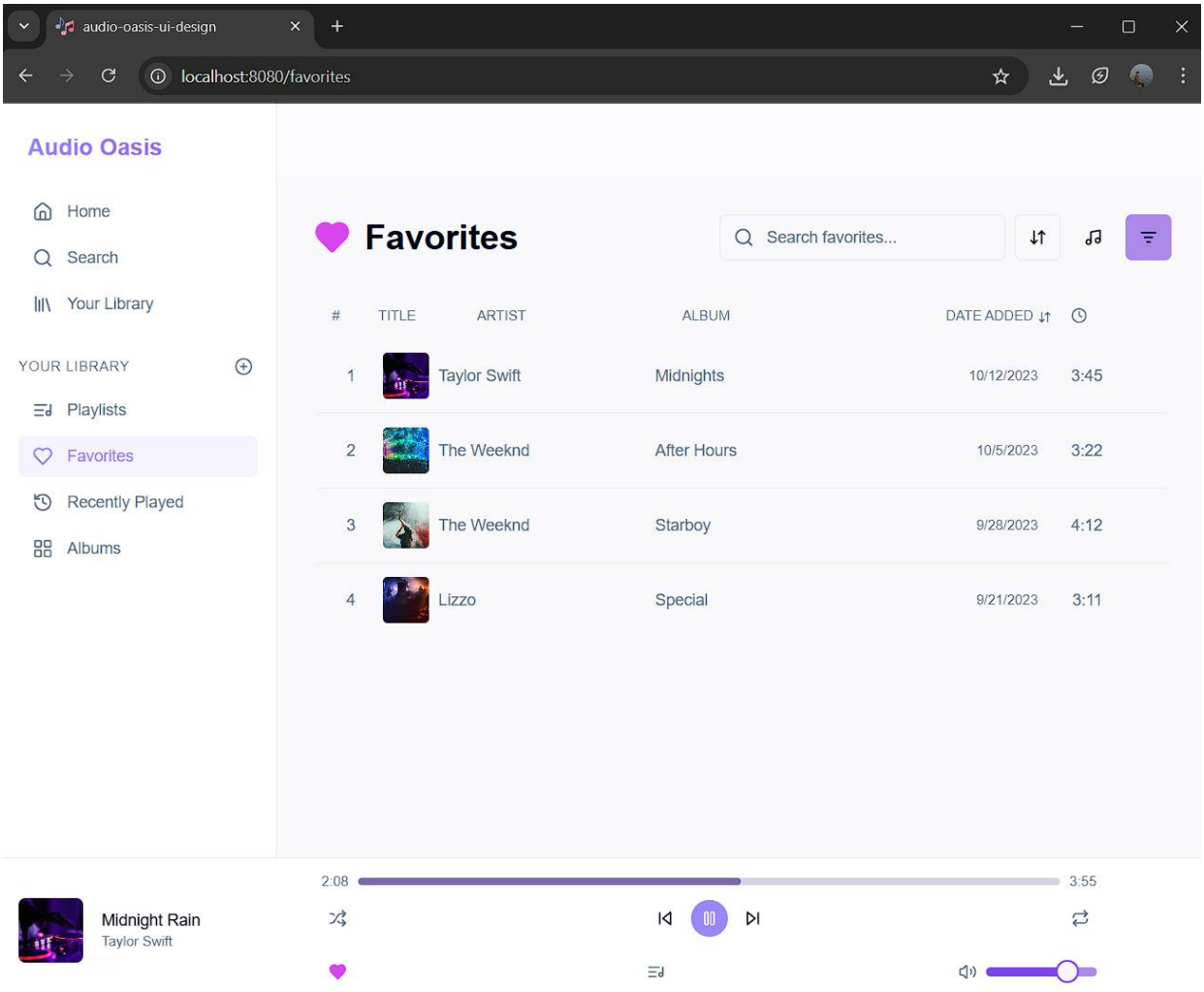


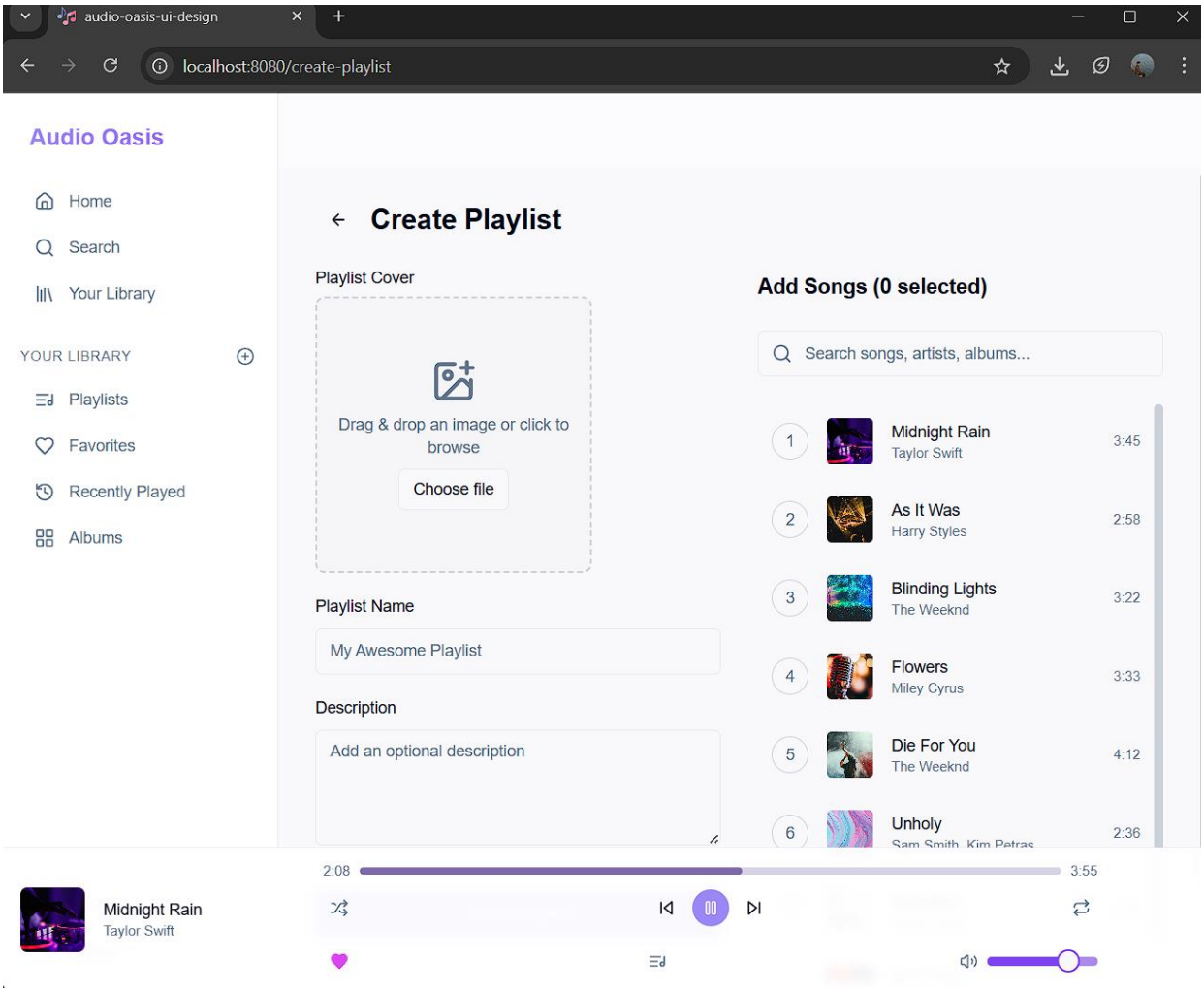
8. PROJECT SCREENSHOTS

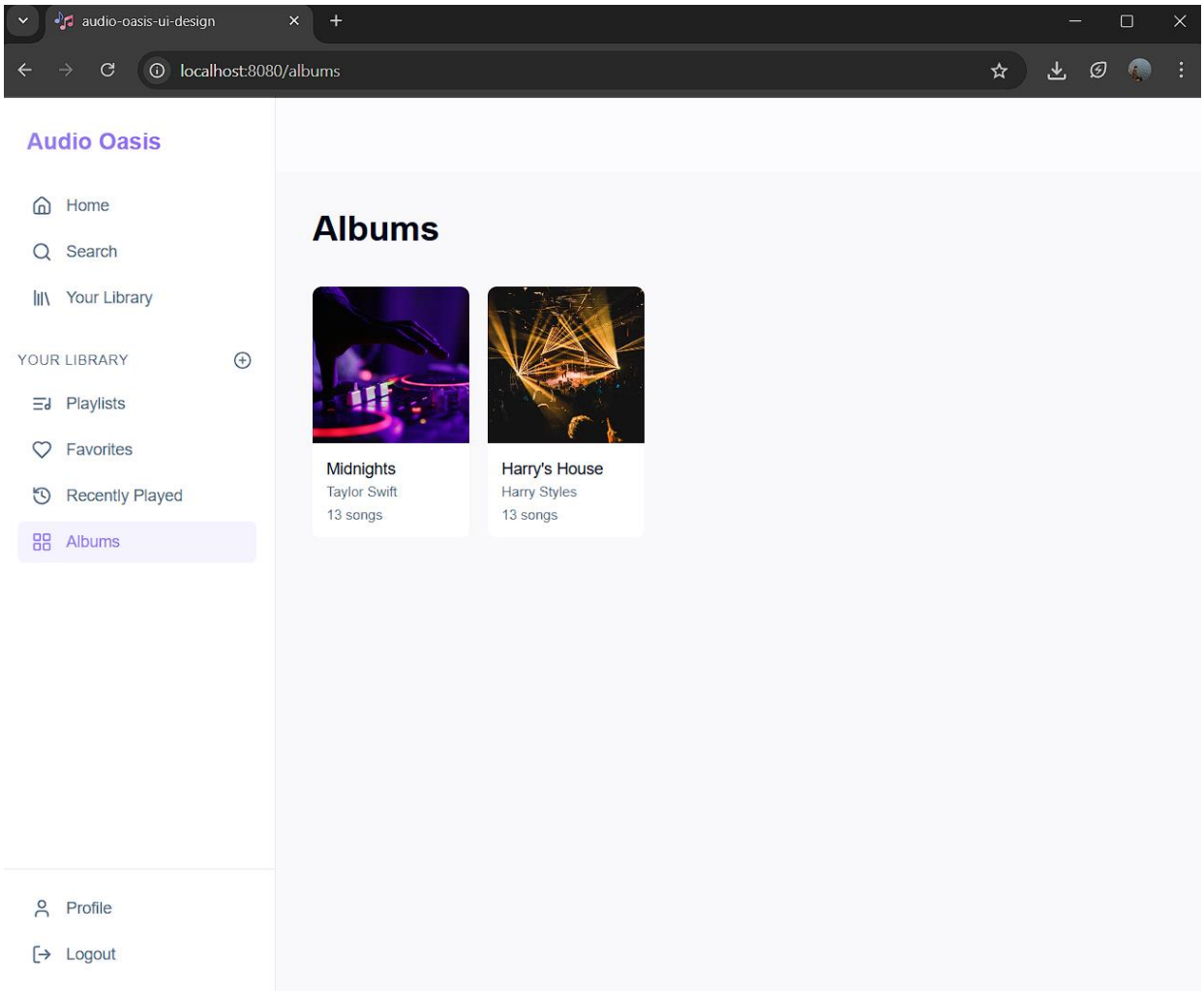

Welcome back
Recently played
See all
Midnight Rain
Taylor Swift
As It Was
Harry Styles
Blinding Lights
The Weeknd
Flowers
Miley Cyrus
Die For You
2:08
3:55
Midnight Rain
Taylor Swift

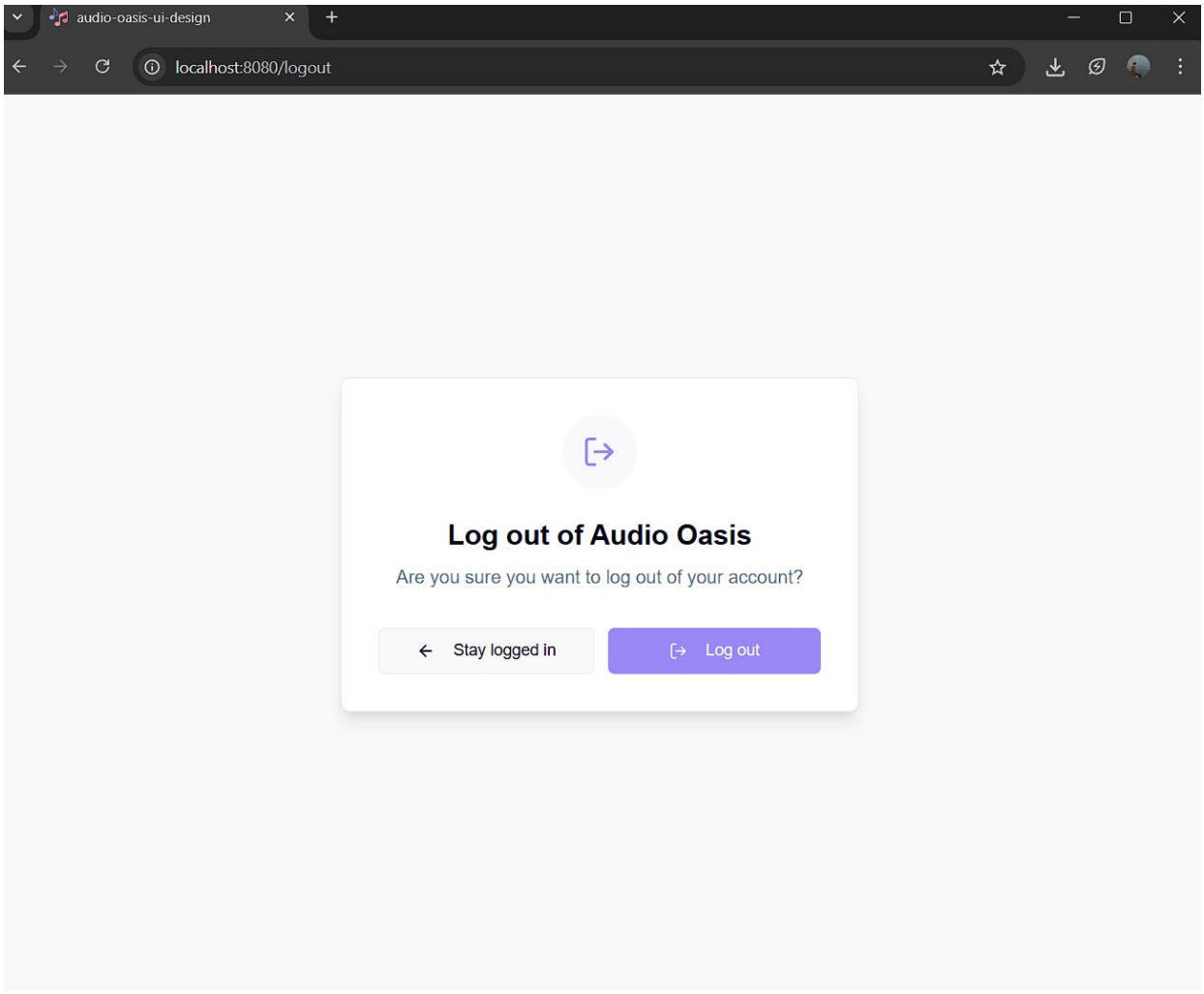

Audio Oasis
Home
Search
Your Library
YOUR LIBRARY
Playlists
Favorites
Recently Played
Albums
Profile
Logout


audio-oasis-ui-design
localhost:8080/now-playing
Audio Oasis
Home
Search
Your Library
YOUR LIBRARY
Playlists
Favorites
Recently Played
Albums
Profile
Logout
Now Playing
Midnight Rain
Taylor Swift
Midnights • 2022
2:00
3:55
Lyrics
Queue
It was pouring rain, you were there
You were there and you held me down
But I was stubborn, I was never gonna settle
Why would I bother with all these others
Faceless nobodies, flashes of teeth
All their futures washed away with the rain
He wanted it comfortable, I wanted that pain
He wanted a bride, I was making my own


audio-oasis-ui-design
localhost:8080/favorites
Audio Oasis
Home
Search
Your Library
YOUR LIBRARY
Playlists
Favorites
Recently Played
Albums
Favorites
Search favorites...
TITLE ARTIST ALBUM DATE ADDED
1 Taylor Swift Midnights 10/12/2023 3:45
2 The Weeknd After Hours 10/5/2023 3:22
3 The Weeknd Starboy 9/28/2023 4:12
4 Lizzo Special 9/21/2023 3:11
2:08
3:55
Midnight Rain
Taylor Swift


audio-oasis-ui-design
localhost:8080/create-playlist
Audio Oasis
Home
Search
Your Library
YOUR LIBRARY
Playlists
Favorites
Recently Played
Albums
Create Playlist
Playlist Cover
Drag & drop an image or click to browse
Choose file
Playlist Name
My Awesome Playlist
Description
Add an optional description
Add Songs (0 selected)
Search songs, artists, albums...
1 Midnight Rain Taylor Swift 3:45
2 As It Was Harry Styles 2:58
3 Blinding Lights The Weeknd 3:22
4 Flowers Miley Cyrus 3:33
5 Die For You The Weeknd 4:12
6 Unholy 2:36
Midnight Rain
Taylor Swift
2:08
3:55


audio-oasis-ui-design
localhost:8080/albums
Audio Oasis
Home
Search
Your Library
YOUR LIBRARY
Playlists
Favorites
Recently Played
Albums
Profile
Logout
Albums
Midnights
Taylor Swift
13 songs
Harry's House
Harry Styles
13 songs



9. CONCLUSION

The development of *Audio Oasis* marked a significant milestone in my journey as a full-stack web developer. By building a music streaming application from the ground up using the **MERN stack**, I gained hands-on experience in designing scalable web architectures, managing real-time interactions, and creating responsive user interfaces.

Throughout this project, I implemented essential features such as **user authentication**, **playlist management**, and **local audio playback**, all while ensuring a smooth and intuitive user experience. Leveraging technologies like **React.js**, **Node.js**, **Express.js**, and **MongoDB**, I was able to create a seamless bridge between the frontend and backend, and effectively manage user data and media content.

Working with **locally stored music files** provided deeper insight into file handling, server configuration, and custom streaming logic—offering more control than relying on external APIs. Moreover, the use of tools like **HTML5 Audio API**, **JWT**, and **React Router** enriched the application's functionality and scalability.

This internship not only enhanced my technical proficiency but also improved my problem-solving, debugging, and system design skills. The project has equipped me with the confidence and practical knowledge needed to contribute meaningfully to real-world software development environments.

The development of the music streaming web application using the MERN stack provided a comprehensive, real-world learning experience in full-stack web development. Throughout the internship and project, I gained practical exposure to React.js for building responsive user interfaces, Node.js and Express.js for handling backend logic, and MongoDB for managing application data. Integrating external music APIs enhanced the dynamic nature of the application by enabling real-time music search and playback functionality.

This project not only helped in understanding the workflow of web applications but also honed essential technical skills such as component-based architecture, asynchronous data handling, RESTful API integration, and full-stack deployment strategies. The process of prototyping, designing with tools like Figma, and handling tasks in a collaborative environment using Jira further added to the professional development aspect of the internship.

10. REFERENCES

- W3Schools. *Node.js Tutorial*. https://www.w3schools.com/nodejs/
- MongoDB Inc. *MongoDB Documentation*. https://www.mongodb.com/docs/
- Express.js Official Docs. *Express Web Framework*. https://expressjs.com/
- Kushagra Null. *Music Recommendation System using MERN*. Kaggle Notebook, https://www.kaggle.com/code/kushagranull/music-recommendation