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	Signature of Co-ordinator
Date: 5/05/25	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

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 build a dynamic, user-friendly web application that allows users to search for songs, play/pause music, and navigate across different pages of the platform. It also involved implementing essential features such as user authentication, API integration for music data, and responsive design. The MERN stack was chosen due to its popularity in modern web development and its ability to support end-to-end JavaScript development—making the development process more streamlined and efficient.

2. LITERATURE SURVEY

All the published papers and research articles cover the following points. These survey aim to identify the current state of the field, key methodologies, challenges, and potential areas for improvement. They typically includes:

1. **Single Page Applications(SPA)**:SPAs improve performance by loading content dynamically without full page reloads. React.js is widely used for building SPAs due to its virtual DOM and component-based structure.

2. **MERN Stack Integration**:The MERN stack offers a full JavaScript-based development environment. It allows developers to manage both frontend and backend using a single language, streamlining communication and reducing context switching.

3. **Use of MongoDB for MediaData**:MongoDB's flexibility with JSON-like documents makes it an ideal choice for storing user data and media-related metadata in music applications.

4. **Node.js and Express.js for Backend APIs**: Node.js provides an asynchronous, event-driven architecture suited for I/O-heavy applications like media streaming. Express.js simplifies routing and API creation for backend services.

5. **API Integration for Music Search**:APIs such as Deezer, Spotify, and Last.fm offer music metadata and preview URLs. Integration of such APIs enhances app functionality by allowing real-time search and playback.

6. **User Authentication using JWT**:JSON Web Tokens provide a secure, stateless method of user authentication, suitable for scalable applications with login functionality.

7. **Media Control Using HTML5 Audio API:**This API supports native control of audio playback in browsers, enabling features like play, pause, seek, and volume control—essential for music applications.

8. **Responsive Design Principles**:Responsive design ensures usability across desktops, tablets, and smartphones. CSS Flexbox and Grid are commonly used for building adaptive layouts.

9. **Frontend Routing with React Router**:React Router is used to manage multiple views and pages within a React application, such as Home, Search, Login, and Playlist screens.

10. **Design Prototyping with Figma**:Figma allows teams to prototype, iterate, and collaborate on UI/UX design. It aids in visualizing component layout before actual development begins.

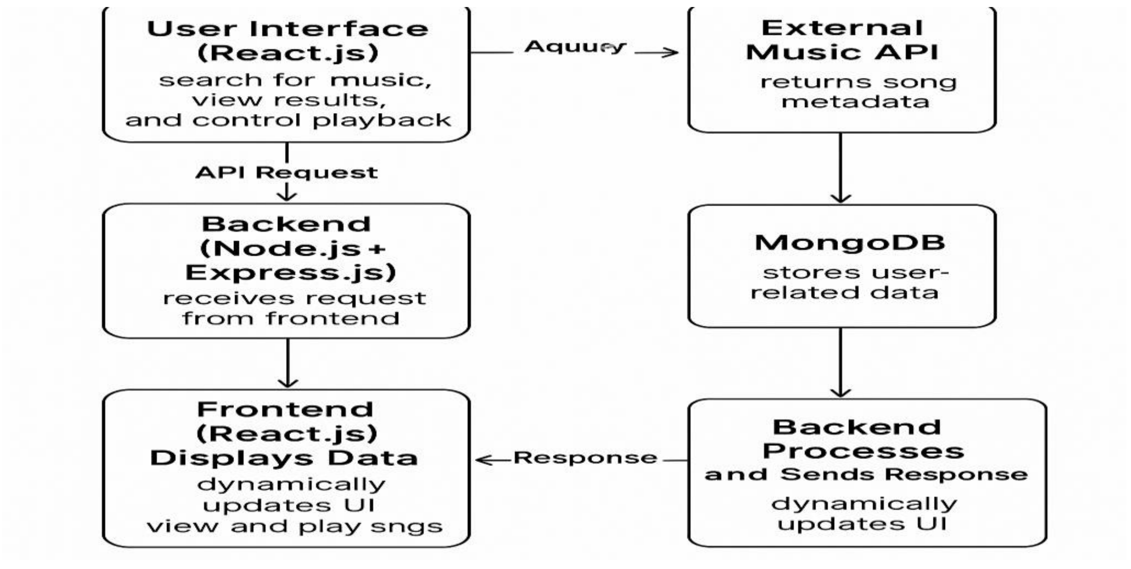
3. PROPOSED WORK

The proposed project involves the design and development of a music streaming web application using the MERN (MongoDB, Express.js, React.js, and Node.js) stack. The primary goal of the project is to provide a user-friendly platform that allows users to search, stream, and manage music through an interactive and responsive web interface. The frontend is developed using React.js, ensuring dynamic content rendering and seamless navigation between components. The backend is powered by Node.js and Express.js, which handle routing, authentication, and server-side operations. MongoDB is used as the database to store user information, playlists, and interaction history in a flexible NoSQL format.

Data Collection

Data for the music streaming application is primarily gathered from third-party APIs that offer access to extensive music catalogs. These APIs are utilized to fetch real-time data such as song titles, artist names, album information, track previews, and album art.

4. IMPLEMENTATION


User Interface (React.js)
search for music, view results, and control playback
Aquuery
External Music API
returns song metadata
API Request
Backend (Node.js + Express.js)
receives request from frontend
MongoDB
stores user-related data
Frontend (React.js) Displays Data
dynamically updates UI
view and play sngs
Response
Backend Processes and Sends Response
dynamically updates UI

1. **User Interface (React.js):**The application begins with the frontend built using **React.js**, where users can search for music, view results, and control music playback. React allows for a dynamic, responsive user experience and manages UI components efficiently.

2. **API Request** :When a user searches for a song, React uses **Axios** (or `fetch`) to send an HTTP request to the backend. This request typically contains search parameters such as artist name, song title, or album.
3. **Backend (Node.js + Express.js):**The backend, powered by **Node.js and Express.js**, receives the request from the frontend. It acts as the core processing layer, validating inputs and preparing requests to external services.
4. **External Music API :**Instead of maintaining its own large music database, the backend sends a query to a **third-party music API** such as Deezer or Last.fm. These services return song metadata like titles, artists, durations, and preview URLs.
5. **MongoDB**:While real-time music data is fetched from APIs, **MongoDB** is used to store persistent user-related information like saved playlists, liked songs, or recent searches. The backend interacts with MongoDB to read/write this data as needed.
6. **Backend Processes and Sends Response:**Once the backend collects music data from the external API and optionally fetches user-specific data from MongoDB, it compiles the response and sends it back to the React frontend.
7. **Frontend (React.js) Displays Data:**The frontend receives the response and dynamically updates the UI. Users can now view search results, click on songs to play them, or navigate between pages such as home, playlist, and search.

5. CONCLUSION

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.

6. 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