**Abstract**

This report presents a case study on the architecture and implementation of a web application called "MusicFest." The MusicFest web application is designed to provide information about a music festival, including lineup details, stages, FAQs, and contact information. The report provides an overview of the web application's structure, the technologies used, and the implementation details. Additionally, the report discusses the challenges faced during the development process and proposes future enhancements for the application.

**1. Introduction**

The MusicFest web application aims to provide music enthusiasts with a centralized platform to access information about the music festival. The application consists of several components, including the homepage, lineup page, stages page, FAQs page, and contact page. These components work together to deliver a seamless user experience.

**2. Web Application Structure**

The MusicFest web application follows a typical web application structure, consisting of the following components:

**2.1.** Public Folder

The public folder contains subdirectories for CSS, JavaScript, and images. The CSS files define the styles and layout of the web pages, while the JavaScript files handle client-side interactions. The images folder stores visual assets such as the MusicFest logo.

**2.2.** Views

The views directory contains the EJS (Embedded JavaScript) templates for each page of the web application. These templates define the structure of the HTML pages and allow dynamic rendering of data.

**2.3.** Routes

The routes directory contains separate JavaScript files for each page of the web application. These files define the routes and request handlers for each page, allowing the application to respond to client requests.

**2.4.** Models

The models directory contains JavaScript files that encapsulate the data access and manipulation logic. These files interact with the underlying database and provide data to the routes and views.

**2.5.** App.js

The app.js file serves as the entry point of the application. It sets up the Express.js server, configures middleware, and establishes the routing and rendering logic.

**3. Technologies Used**

The MusicFest web application is built using the following technologies:

**3.1**. Node.js

Node.js is a JavaScript runtime environment that allows server-side execution of JavaScript code. It provides a non-blocking, event-driven architecture that enables efficient handling of concurrent requests.

**3.2.** Express.js

Express.js is a popular web application framework for Node.js. It simplifies the development of web applications by providing a robust set of features and middleware for handling routing, request processing, and response generation.

**3.3.** MySQL

MySQL is a widely used relational database management system. It is employed in the MusicFest application to store and retrieve data related to the lineup, stages, and other information.

**3.4.** EJS (Embedded JavaScript)

EJS is a templating language that allows the dynamic generation of HTML pages. It is used in the MusicFest application to render data retrieved from the database and present it to the user.

**4. Implementation Details**

**4.1.** Homepage (index.ejs)

The homepage of the MusicFest application provides an overview of the festival, including highlights of the lineup and general information about the event. The index.ejs template dynamically renders the lineup highlights using the lineup data obtained from the database.

**4.2.** Lineup Page (lineup.ejs)

The lineup page displays the complete lineup of artists performing at the music festival. The lineup.ejs template retrieves the lineup data from the database and renders it in a visually appealing manner, including artist names, descriptions, performance dates, and stage names. The images of the artists are also displayed using the artwork\_url attribute.

**4.3.** Stages Page (stages.ejs)

The stages page provides information about the different stages at the music festival. The stages.ejs template fetches the stage data from the database and presents it in a tabular format. Each stage is listed with its name, description, and location. The table rows alternate between different light colors for improved readability.

**4.4.** FAQs Page (faq.ejs)

The FAQs page addresses frequently asked questions about the music festival. The faq.ejs template renders the FAQ data obtained from the server using the faq array. Each FAQ item is presented with its title and corresponding answer.

**4.5.** Contact Page (contact.ejs)

The contact page allows users to get in touch with the event organizers. It provides a form for users to submit their inquiries or messages. The form data is sent to the server for processing.

**5. Challenges and Future Enhancements**

**5.1.** Database Scaling

As the MusicFest application grows, the database may experience increased traffic and data size. To handle scalability challenges, it is recommended to employ database scaling techniques such as sharding or replication to distribute the workload and ensure high availability.

**5.2.** User Authentication

Currently, the MusicFest application does not include user authentication and authorization features. Implementing user registration, login, and access control would enhance the application's security and enable personalized experiences for users.

**5.3.** Performance Optimization

To improve the application's performance, various optimizations can be implemented, such as caching frequently accessed data, minimizing database queries, and optimizing client-side scripts and assets.

**6. Conclusion**

The MusicFest web application provides a comprehensive platform for music festival enthusiasts to access lineup details, stage information, FAQs, and contact information. The application's architecture, built using Node.js, Express.js, MySQL, and EJS, enables seamless rendering and retrieval of data from the backend. Despite the challenges faced during development, the application successfully delivers an attractive and professional user experience.

References

1. Node.js. Retrieved from<https://nodejs.org/>
2. Express.js. Retrieved from<https://expressjs.com/>
3. MySQL. Retrieved from<https://www.mysql.com/>
4. EJS (Embedded JavaScript). Retrieved from<https://ejs.co/>

Here’s the structure of the file  
  
**/my-event-website**

**│**

**├── /public**

**│ ├── /css**

**│ │ ├── main.css**

**│ │**

**│ ├── /js**

**│ │ ├── main.js**

**│ │**

**│ ├── /images**

**│ ├── logo.png**

**│**

**├── /views**

**│ ├── index.ejs**

**│ ├── lineup.ejs**

**│ ├── stages.ejs**

**│ ├── faq.ejs**

**│ ├── contact.ejs**

**│**

**├── /routes**

**│ ├── index.js**

**│ ├── lineup.js**

**│ ├── stages.js**

**│ ├── faq.js**

**│ ├── contact.js**

**│**

├── /models

│ ├── lineup.js

│ ├── stages.js

│

├── /node\_modules

│

├── app.js

├── package.json

└── README.md