Kendall Comeaux

Playlist Creation Project

December 3, 2021

Table of Contents

Introduction

Playlist Creation Project, henceforth referred to as Slaps0rNot provides a platform for amateur musicians, DJ's and musical enthusiasts to share, rate and create playlists. Each user makes an account and then can produce playlists for others to rate and share.

The implementation of the project was designed as a three-tier web application completely separating the client-side presentation and server-side logic with the data system on the same server at the server-side logic. The presentation layer is composed of static content delivered to the user as a front-end interface interpreted by web browsers and client-side scripting to facilitate a more dynamic user experience. User interaction with the website in the browser triggers calls to the server-side logic, which manipulates the data system and returns requested data to the presentation layer where it is dynamically interpreted and presented to the user inside the browser. The data system provides structured storage and fast retrieval of user-generated content that should be persistent.

Related Works

slapsOrNot shares common traits with other music sharing services such as Spotify and Pandora. However unlike those slapsOrNot will allow users the to rate and comment on the playlists of other users. This feature will separate us from other services by allowing our users more communication with each other and the ability to improve their music routines based on the feedback of others.

System Architectural Design

The system architecture follows the general three-tier web application design described in the introduction. The application was deployed across two Amazon EC2 instances, one containing the presentation layer and the other containing both the logic and data layers. The first instance runs Nginx hosting a react web application. The second instance runs a node.js server and MySQL for the relational database storing slapsOrNot data. The server was hosted on the ip address of the react ec2 instance. The Web Server was connected to the node server by allowing request from the ip address of the react application. The node server uses sequelize to connect to the MySQL database. The front-end HTML, CSS, and JS was developed using the react library. The interactable portion of the frontend was done using react with browser router and use-state along with axios the send requests to the node server. The server-side functionality will be described in more depth in the next section but in general it interfaces with the database to return data in json form. It is the job of the client-side front-end to make

the returned JSON data presentable. In the developed application, react structures the data as styled HTML renders it for the client. However, any kind of application (e.g. a mobile app or a site with a different UI) could potentially be configured to work with these servlets because the JSON data is completely neutral to the presentation format. Therefore, the back-end of slapsOrNot could be an API for multiple front-end applications to use and making the presentation layer completely independent. MySQL was chosen as the data system because of the general benefits of a Relational Database Management System for persistent storage and the ease of integration with the express framework using sequelize. The design and development process resulted in an entity-relationship model of the data needed for the slapsOrNot application as show in the accompanying PDF file.

Relational table: users

The users table stores the data of user accounts. The primary key is the user's email address, which is used as a foreign key in other tables. Note that for security passwords should not be stored as plain text, but they were for the purpose of rapid development on this application.

Relational table: playlists

The playlists table stores the data for playlists created by users. The primary key is an automatically incremented integer. The email attribute is a foreign key that refers to email in the users table. The name attribute refers to the name of the playlist. And the songs attribute is a JSON array of the songs in the playlist.

Relational table: reviews

The reviews table stores the data for reviews created by users. The primary key is the email address of the user writing the review (reviewer) coupled with the name of the playlist the review is about to ensure that one user can write a maximum of one review for each playlist.

Detailed Description of Components

The following is a description of each functional component of the slapsOrNot application as a function in the express server with a description of how it effects the front end. The server responds with get, post and delete requests. The express server configures the Access-Control-AllowOrigin HTTP header to enable CORS with the slapsOrNot domain that the axios request comes from. If a request doesn't specify a required parameter for the server, then the server will return without a response. sequelize is used for

all communication with the database and prepared statements are used when handling user submitted data.

Account Registration

The account registration servlet receives an email address, name and password for registration of a new account. The signup page queries the database to check if there are any accounts already associated with the email address, and if not then insert a row into the database with the email, name, and password.

Account Login

When the user types an email, name and password and clicks the “Sign in” button. Axios will send a request to the server. The server will check the database and get the users information. If the information is correct, the server will redirect the user to the home screen and pass their information using useState. The session information will allow the user to access the functionality on the web page.

Session Information

All pages past login require a valid state to retrieve the server’s view of the session. The react app passes the user’s session information between pages until the user logs out. This information is used by the client-side for presentation and verification purposes such as showing the user their playlists and allowing them to submit reviews.

Playlist Creation

On the playlist creating page. The user can query the database. The server receives the name of the user and what he wants to call the playlist and then adds it the database. The client will then be able the view their playlist on the page. The client uses the session information to determine what playlist belong to them.

Song Creation

The song creation method receives a playlist and a song to add to it. The current email address attribute in the session is used as the owner of the playlist. These values are inserted into the database where the songs are stored as a JSON array.

Song deletion

The song deletion function deletes songs from the playlist. The current email address attribute in the session is used as the owner of the playlist. The server extracts the song array from the database before updating it. Checking both the name and email ensure the correct playlist is modified.

Playlist Deletion

The playlist deletion function receives json data from the axios delete method. The current email address attribute is used as the playlist owner. The server uses a sequelize method to delete the playlist under a certain email address and name. Checking the owner and the list ensure a user can only delete their own playlists.

Home Screen

The home screen queries the database to receive the users playlists. The name of the user is also stored and displayed along with the playlists. the screen displays the name of the user and their playlists.

Review Creation

The review creation server receives the playlist the review is about from the axios call. The email address and name are used to determine the reviewer. The rating is then added to the database and the count is updated to reflect the new rating. The users comment if the added one will displayed on the playlist's reviews page.

Review Deletion

The review deletion server receives the playlist the review is about from the axios call. The current email address and playlist name attribute are used to confirm the user. The server uses a prepared statement to execute an update to the database deleting the review from the playlist. In order to delete a review you must be logged in the with the email address of he reviews creater.

Account Logout

The logout feature invalidates the current session of the requester which will prevent that user from accessing or creating private information in the application. The client redirects to the original front page and clears the state so the users account can no longer be accessed

Conclusion

The final implementation of the slapsOrNot demo delivers a fully functional three-tier web application hosted at http://18.222.105.117/ that meets the goals of providing an easy-to use space for amateur musicians and musical enthusiasts to share their ideas. The service is relatively free of bugs with few observable errors under normal usage and runs responsively and efficiently with a standard Internet speed. Because the front-end uses the screen-responsive react framework, the website already functions as a mobile web application on smaller devices. The main area's for future improvement are security and Interface design, where communications should be encrypted and passwords stored securely. The UI should be redesigned to appear as one would expect of an app marketed to artists and musicians.