Alec Bielanos
IGME430 Project III – EZChord II
December 2017

What Does My Site Do?

EZChord is a tab searching service which scans existing libraries of guitar tablatures (tabs) and presents them in a service which is easy to interact with. Users may create accounts and save their favorite tabs for quicker access later (and may also delete them from the list if they choose). Additionally, the user may change the background and text colors of the service to their liking.

Profitability

The service could potentially profit from multiple different sources. First, and most obvious, ad space can be added in the margins or at the footer, generating revenue per user visiting the page or per click. Next, the user may be limited to a certain number of "favorite" tabs until they paid for the service (i.e. – Free to use with paid option). Other premium features, such as the ability to see how to play the chords listed in the service or audio that plays alongside the tab are possible future adds (see future work) that could be considered premium features.

React Elements

In EZChordII, react is being used to show all of the page content with the exception of the navigation bar and the form to change colors on the settings page (color pickers did not work inside of React elements). Examples of React elements are the tab search box, favorites window, login/signup/change password forms, tab results window, etc.

MVC Design

EZChordII uses the MVC design pattern (Express Library) in the backend for separation of concern. Each request is routed to one of several controllers (Account, Tabs, Searching/Scraping) which will make appropriate changes to models, or change the view as needed. Models include favorited tabs and user accounts that are mapped in schemas to a MongoDB database, which stores basic account information (username/password) as well as the colors of the API (text color/background color). In addition, I am using Handlebars as the view engine to render each page as a baseline for where to insert the react content. In the templates, I have set up basic section elements with IDs that I am able to point at when making ReactDOM.render calls from the client side scripts.

External Dependencies

This project uses the Ultimate Guitar Scraper API:
https://www.npmjs.com/package/ultimate-guitar-scraper

Future Work

In the future, I plan to add my ideas for 'premium' features to the service (clicking on chords to retrieve chord information, playing audio alongside the tab via YouTube or Soundcloud) regardless of whether or not I implement monetization. I would also like to take a look at brushing up on the CSS and using a preprocessor like SASS to implement monochromatic color schemes to the service, so that when the user changes color it doesn't make all of the elements the exact same color.

Above and Beyond

In this project, I was able to include more than just the required functionality and client/server communication. The app includes several different ajax requests that are handled smoothly, handles errors appropriately using the correct HTTP status codes, and dynamically updates the content on the page smoothly. Almost all of the functional elements are in React and are stateless. The code on both the client and server side is tight, formatted and commented. Overall, it's a complete, clean app that I can continue to build on with ease.