Overview of my MERN Stack App "Back Track"

Back Track is group calendar application which uses a MERN Stack (MongoDB, Express, React, Node.js) to create a user interface for a shared google calendar. Additional functionality includes features to encourage competition such as points and leaderboards for users who schedule the most hours in a particular topic. The MERN stack is a set of open source components which provide an end-to-end framework which utilizes JavaScript for every component. MongoDB is an open source document database which stores JSON documents in collections with dynamic schemas. It is used by the back end of the application to store data. Express is a web application framework that runs the back end of the application. In this app Express connects the front end to the database. React is a JavaScript library for building user interfaces. The front end is broken into components which can hold its own state and can pass its state down to child components. Child components can also pass changes back to the parent through the use of callback functions. Node.js is a JavaScript runtime environment that runs the back end using express. It also includes a number of modules that can be installed using the node package manager tool.

Creating a Topic

This blog will go through essential features of the app describing the design and setup. This will encompass the entire setup of a MERN application but it is not a how to with every line of code explained in the order it is written. This application is broken up into three major pages: a make topic page, a make user page and a home page. Rather than fetching the other pages from the server this app utilizes React-Router. There are three packages that fall within react-router. There is a core package, a package for react native and this application uses the react-router-dom package. React-router-dom contains the document object model bindings for the react router. The React Router API is based on three components: Router (the route that keeps the UI in sync with the URL), Link (Renders a navigation link) and Route (renders a UI component dependency on the URL). The Router used in this application is BrowserRouter the tags can be seen surrounding the App component in the src/index.js below.

A router component can only have one child element so it is important that the App component be the child component of BrowserRouter. Within the App component a switch statement with the different pages components. When the location changes because of a navigation action, the child component App is rerendered. The App component can be seen below.

The Links in the home page change the route and rerender the App component. The user begins at the home page and can make a new topic, or make a new user

There are three key changes that would need to be made to a stock create-react-app to setup these three pages. First the App tag in the \_\_ needs to be wrapped in the react-router-dom Browser router tags as seen below. Second the App component needs to separate each page and its respective component in a switch case. Third some instruction to change the route needs to be given in this case it is a button with a link to the page.

The Make Topic page contains a table with the existing topics, a modal used to fill out information for new topics, a loading icon and a footer that contains the add student button that triggers the modal. These components interact with a state that contains a topics array as well as a isFetching and isLoading Boolean.