CS411 Team Assignment 3 - Prototyping API Calls

Group 20 - Manish Patel, Ji Zhang, Chen Feng, Curtis Mason and Takamitsu Shirono

Deliverable Content

For our project, we decided to employ ReactJS on the front-end, given that a couple of our team members were familiar with it. Additionally, using ReactJS will be beneficial for the other team members and the project objective, since it is relatively easy to learn for the team members who are not familiar with it. It also provides a great deal of flexibility in development and has a huge community providing insights and support which would be helpful when navigating through this project. We will be using TypeScript alongside ReactJS, since we believe that employing a language with static typing allows us to minimize any runtime errors and make it easier for us developers to maintain and parse through other's work.

For the back-end, we decided to use Flask (and naturally Python as our programming language) as it is lightweight, easy to use, and flexible in development. Additionally, most of our team members are already comfortable with Python, which should further simplify the development process. We are also considering employing SQL databases to handle cookies and session management for users who have recently logged in. We could potentially manage other basic information about the users in this database as well.

In regards to the API, we are going to be using the Google Calendar API (GCal API) to obtain users' schedules and create or modify users' events on their calendar. We chose this API as most of the students at BU use Google Calendar instead of Outlook, since our BU student accounts are based in Gmail. Additionally, GCal API has extensive documentation and a large developer community which will assist us in our development process. In order to access users' Google Calendar information through the GCal API, we need to utilize Google's OAuth2 protocol, which is an authentication protocol for users' to authenticate external applications to access their GSuite information (such as the GCal information). We will be requiring every user upon logging on (or once, when they register) to authenticate our application to access their calendar through OAuth2 protocol, and that authentication information will be stored in a secure storage in the backend. We will also be utilizing cookies to prolong this authentication information for the users' convenience.

Another option we considered is a server-side rendering model, through means such as Prisma, Nexus, and ReactJS. We chose our current tech stack over this one because the team is more familiar with ReactJS on the frontend and Python on the backend. Our team has more experience and preference over client-side rendering rather than server-side rendering. Although server-side rendering is an excellent concept and has a lot of advantages, it also has some disadvantages: Server-side rendering seems to be a simple concept; however, its complexity increases as the complexity of the application increases. Rendering a big application on the server-side can be very time consuming and it may increase the loading time due to it being a single bottleneck. Since our project is being incrementally developed, we chose to minimize the build complexity and went with the direct path of least resistance, which in our case is client-side rendering.