Meeter

Software Engineering
ITSC-3155
Final Project Report

May 6th, 2021

Dhruvi Desai
John Lee
Shane Kemp
Shruti Rabara

Table of Contents

1. Introduction

The objective of this project is to develop an application that can easily set up a meeting schedule between two individuals with the least amount of work required for users to perform. As technologies are becoming mandatory for most of the population in the US, not that many people would carry their calendars around and write down every one of their schedules in their notes. And with the overwhelming amount of different worlds and schedules that individuals are assigned to, there should be an easy tool that can handle the organization.

Google calendar is one of the most used software that enables users to organize their schedules easily and nicely with minimal effort required into making a schedule. Using the Google calendar, a software that can automatically generate the meeting schedule between two entities will come handy in the upcoming future world for personal and business purposes.

1.1 Purpose

Through the project, we are looking to create a software that can optimize the schedule of two entities with minimal amount of work required. This will help users to find a matching free schedule for them to create a new meeting schedule among themselves. More so, using our software, instead of looking through two different schedules, users will be able to find a list of available meeting times with other users by just a click of a button and selecting one of the available time slots. Our software will then send an email notification to the other user asking for confirmation on the selected time. Once confirmed, this event will be created in both of the users' calendars.

We propose to create a software application that can handle two user's schedules through Google calendar, and generate a meeting schedule based on their available times from the received schedule upon both users' agreements.

1.2 Scope

From this project, an optimized potential schedule for two end users to host a meeting would be generated. The benefit of Meeter is to simplify the process of making an appointment for two separate parties. It allows users to match and compare their different schedules to help identify an open slot for both users. Our final product will reduce the amount of work, time, and effort it takes for an individual to make an appointment. The program is designed to reduce the risks associated with which may be generated in cases such as communication difficulties and locational difficulties. Each circumstances will be resolved through the use of the product by a simplified, casually visible user interface of the result. The application organizes both users' schedules with better visualization of both parties through clean, simplified end user output that is easily identifiable in the causal calendar interface.

1.3 Context Diagram and Data Flow Diagram

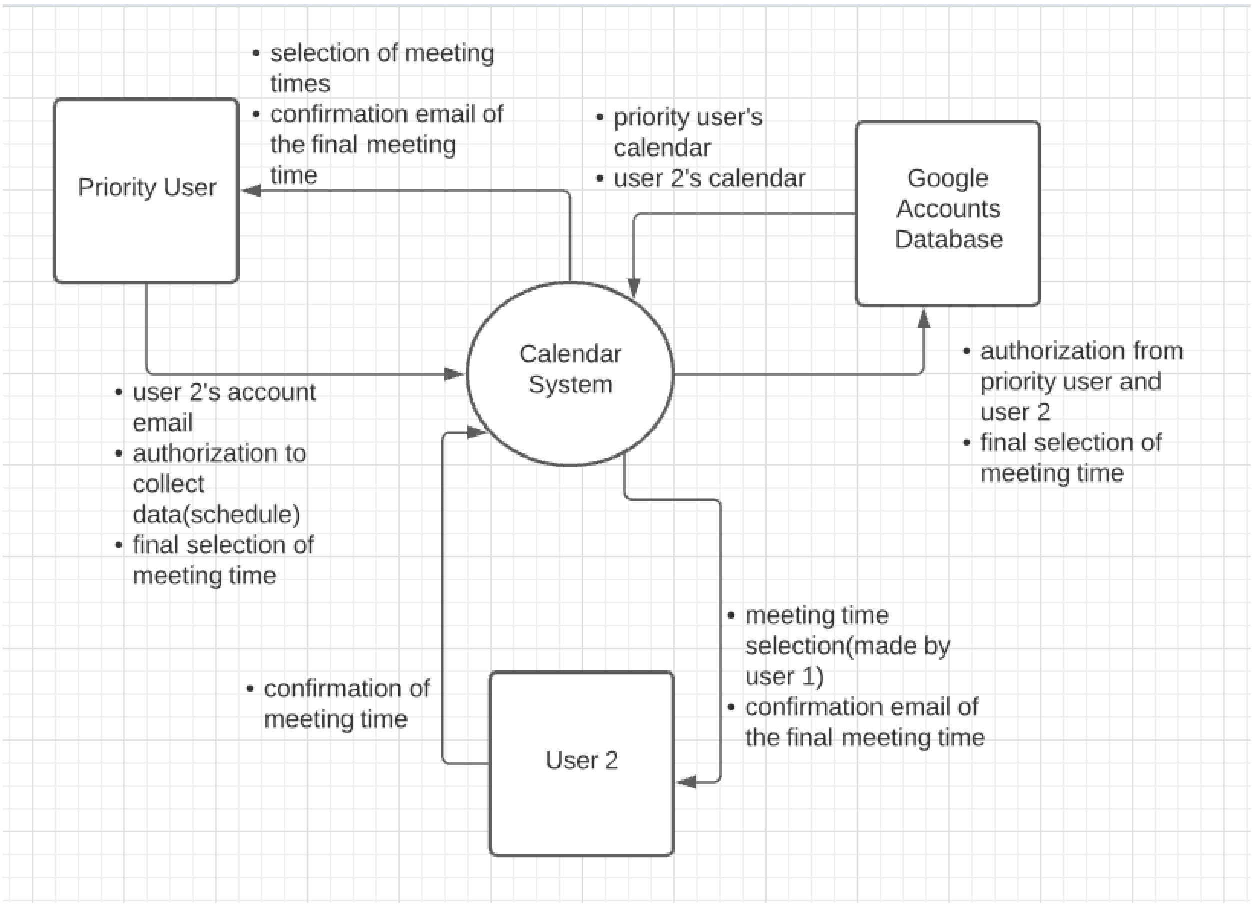

• selection of meeting times
• confirmation email of the final meeting time
Priority User
• priority user's calendar
• user 2's calendar
Google Accounts Database
Calendar System
• authorization from priority user and user 2
• final selection of meeting time
• user 2's account email
• authorization to collect data(schedule)
• final selection of meeting time
• meeting time selection(made by user 1)
• confirmation email of the final meeting time
• confirmation of meeting time
User 2

Figure 1.3.1: Context diagram of Meeter

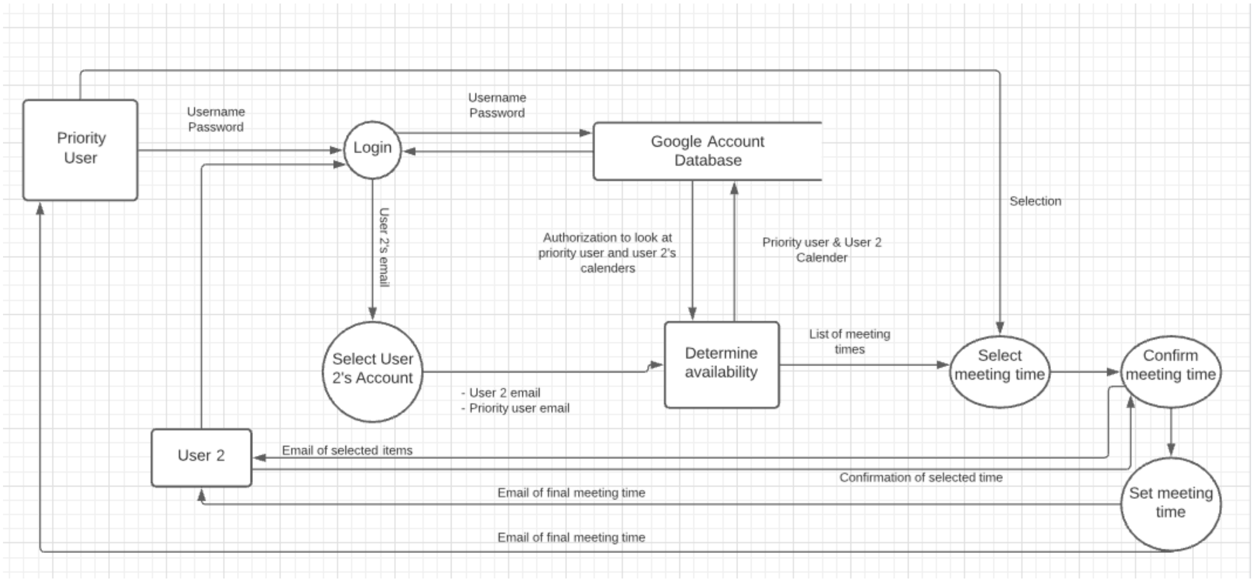


Figure 1.3.2 Data flow diagram of Meeter

1.4 References

https://www.npmjs.com/package/react-google-calendar-api
Title: react-google-calendar-api
Date: April 11, 2021
Publishing organization: npmjs.com

https://developers.google.com/calendar/v3/reference/calendarList
Title: CalendarList
Date: May 4, 2020
Publishing organization: Google

https://githubmemory.com/repo/ericz1803/react-google-calendar
Title: ericz1803/react-google-calendar
Date: May 9, 2020
Publishing organization: Github

1.5 Overview

The program will enable user to input their emails to the given user interface, and the email that was input will be used to search the Google Calendar database through Google Calendar API. Then the program will use given the data from Google database which will be calculated and then output the optimized result of both users' free schedule time range to schedule a meeting. The study of the project is focused on user optimization of a specific, pre-existing application which incorporates integration of API and an effective program that uses the integrated resources in a respectful and efficient manner.

2. General Description

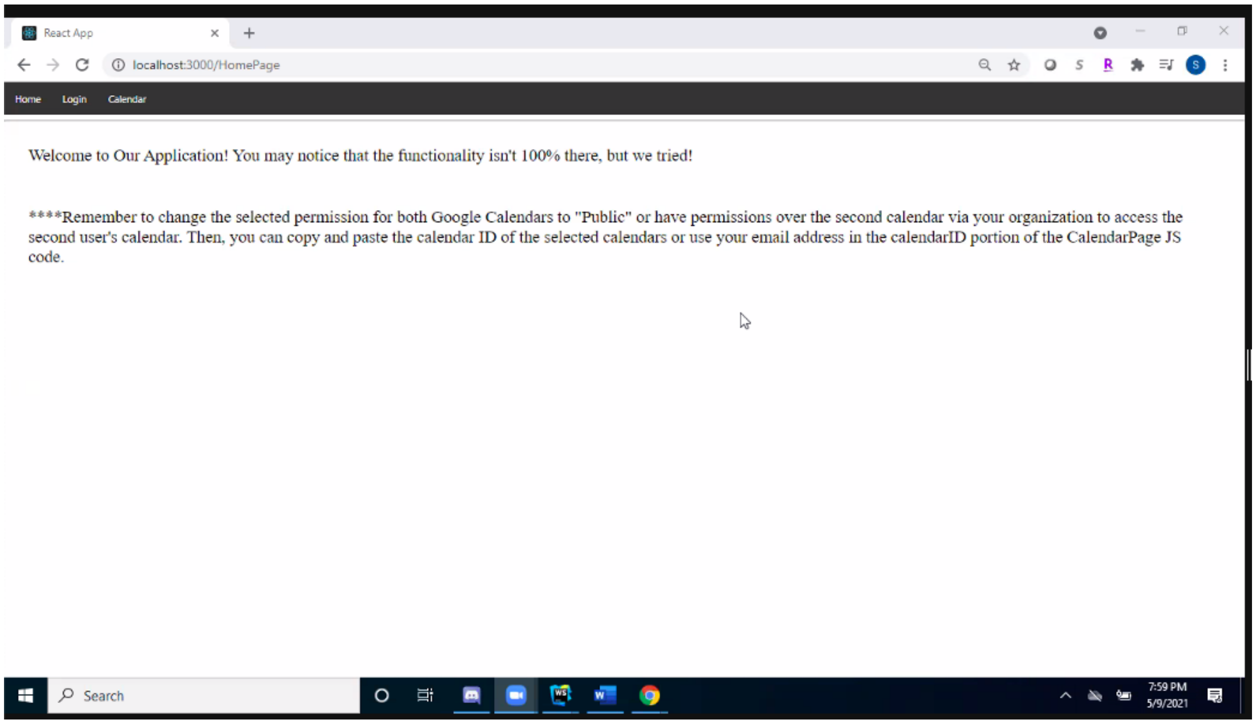
The project focuses on creating a software that can guide the user to give their schedule information listed in the Google Calendar to form up a meeting schedule for two separate users. Two users will give their pre-existing calendar information to the user through Google Calendar API, and then the software will generate the possible resolutions for meeting schedules depending on their time of availability. The project will minimize the users effort and improve their time management skills through the software to generate meeting schedules between two users.

3. Effort

Task	Estimated Time of Research	Actual Time of Research	Estimated Coding Effort (1 - Simple to 5 - Complex)	Actual Coding Effort (1 - Simple to 5 - Complex)
Find an interface to implement our system	3 weeks	1 week	1	1
Find the best way to import React Google Calendar API	3 weeks	1 week	1	1
Utilize the React Google Calendar API to display a calendar with two user's events, simultaneously	3 weeks	2 weeks	2	2
Utilize the React Google Calendar API to access user's calendar events and create a function to find and fill an empty array of available times	3 weeks	3 weeks	5	5
Display list of available times to webpage and take in selected input from user	3 weeks	3 weeks	5	5
Create an event for selected time and add to both user's calendars	3 weeks	3 weeks	5	5
Display a success message to inform the user that their event was successfully created	3 weeks	3 weeks	3	3

4. Programs Developed

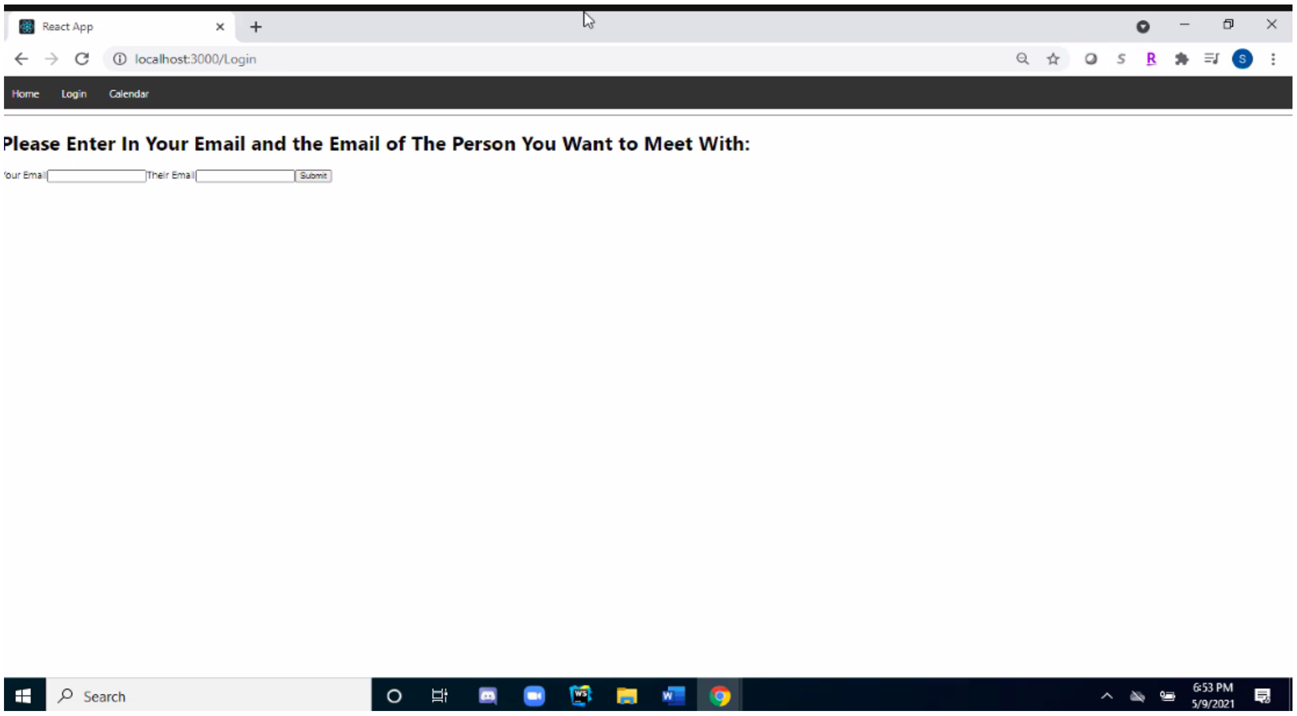
4.1 User Interfaces:



Welcome to Our Application! You may notice that the functionality isn't 100% there, but we tried!

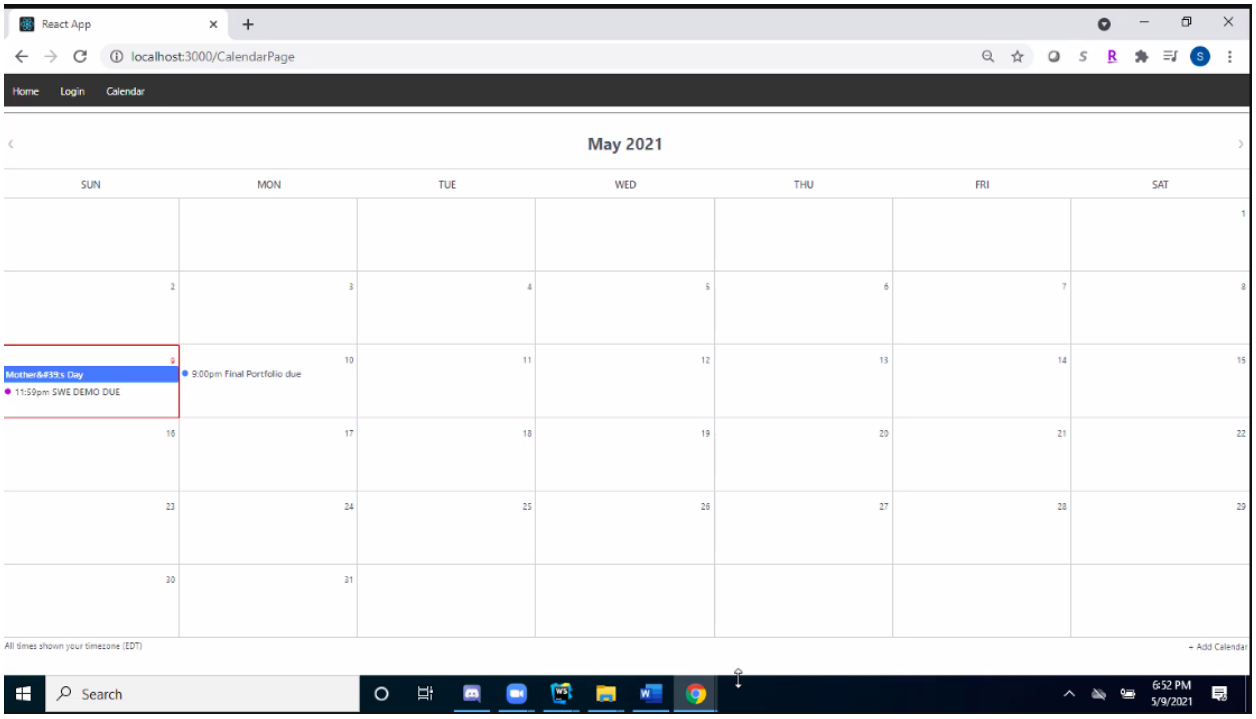
****Remember to change the selected permission for both Google Calendars to "Public" or have permissions over the second calendar via your organization to access the second user's calendar. Then, you can copy and paste the calendar ID of the selected calendars or use your email address in the calendarID portion of the CalendarPage JS code.

Interface Homepage



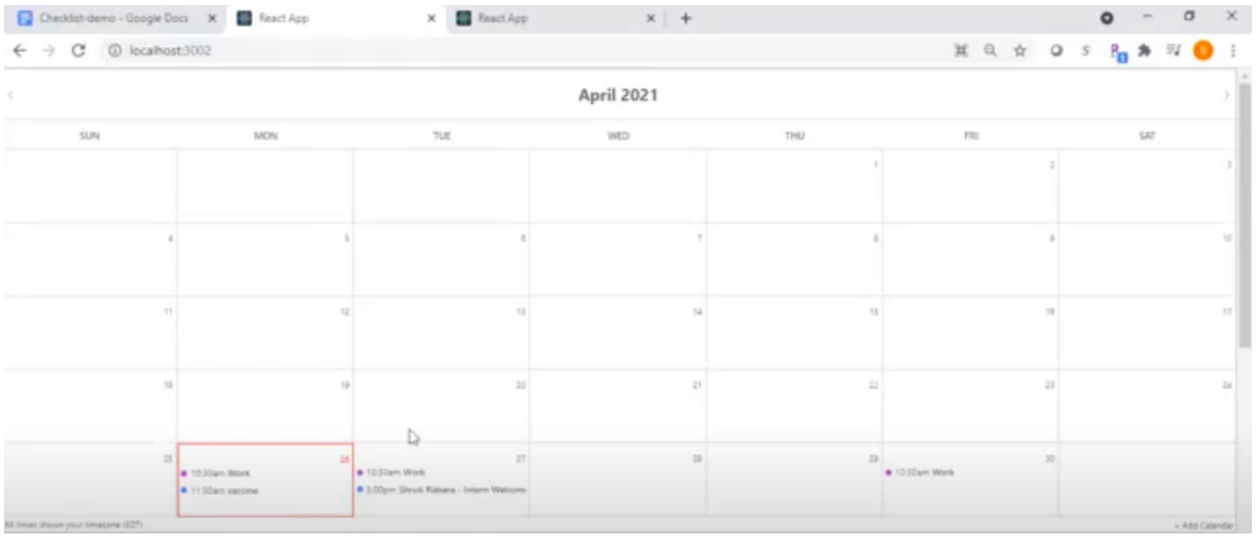
Please Enter In Your Email and the Email of The Person You Want to Meet With:

Interface Login Page

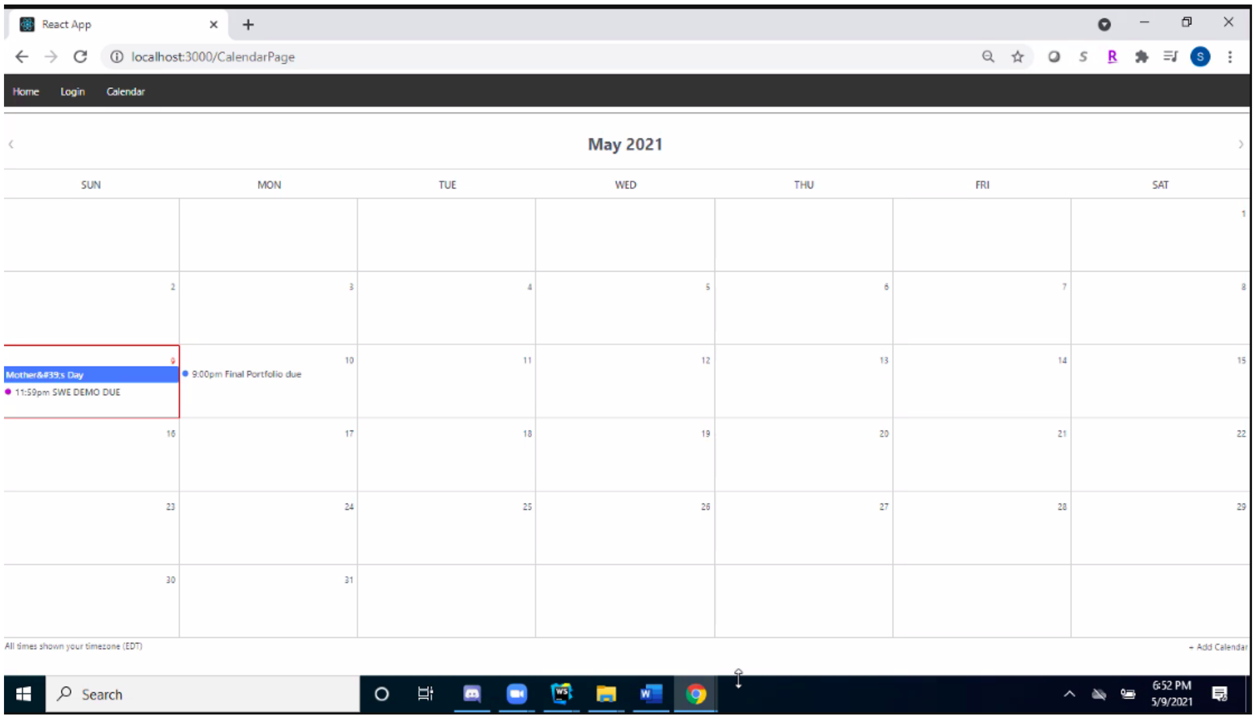


Interface Calendar Page

4.2 Comparison:
In terms of the interface status at our first demo link submission, our interface contained one webpage that was composed of our built in calendar that contained events from two different calendars. When clicked on any event in the calendar, it displayed a message containing more information on the event such as the time and whose calendar it is coming from. Additionally, it also displayed an option to copy the event to the user's other calendars, by navigating the user to the google calendar website for their account. For our final interface submission, we created two different web pages within the website called Login and Home. The home page contains information for the purpose and usability of the calendar. The login page asks the user to input their email addresses for the two calendars that they are trying to match. And lastly, the calendar page displays the calendar with upcoming events for both users. The visible difference between the calendar page from the first demo presentation and the final demo presentation, is the top navigation bar that allows the users to switch back and forth between the three pages within the interface.

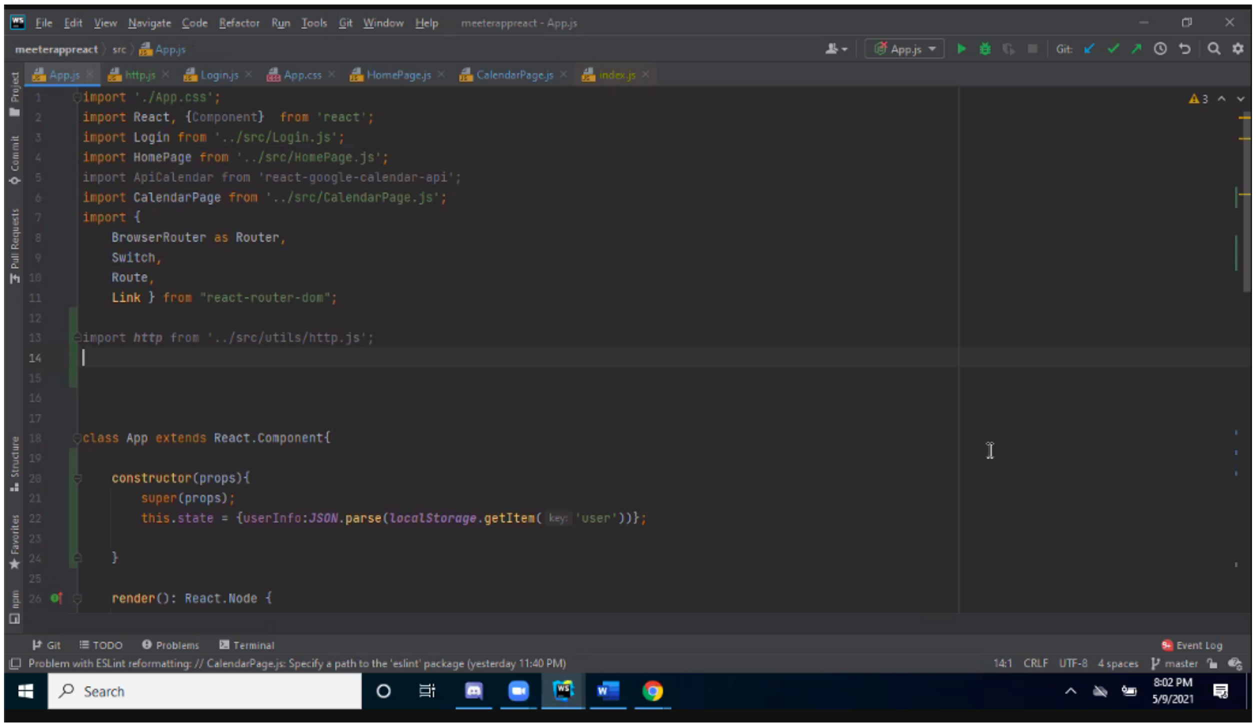


Interface before



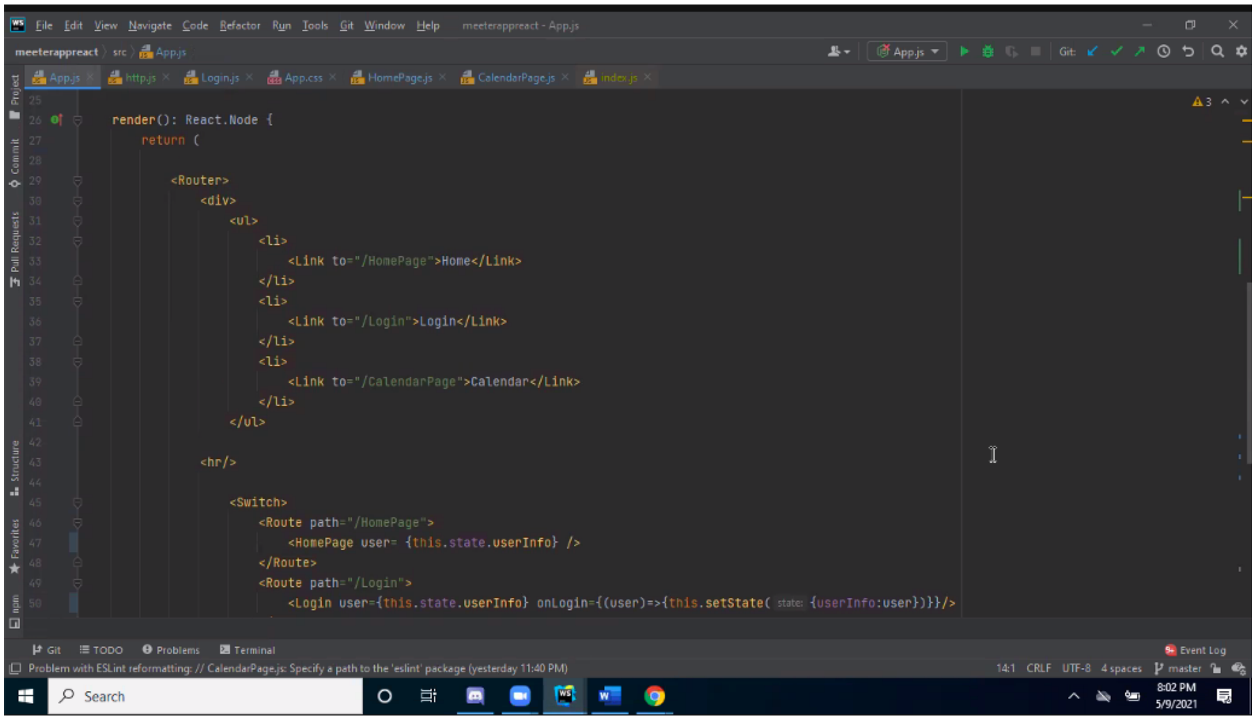
Interface after

In terms of the interface code status between the first demo link submission and the final demo link submission, there is some difference between the two. Since our group was not able to complete all of our intended tasks for this project, there may not be as many changes, however, we did go through many trial and error stages of trying different bits of code. We decided to only show code that works with our interface and leave out our testing code. To create the different individual pages, we added in different switches in our root file, aka App.js, and functions to support those events.



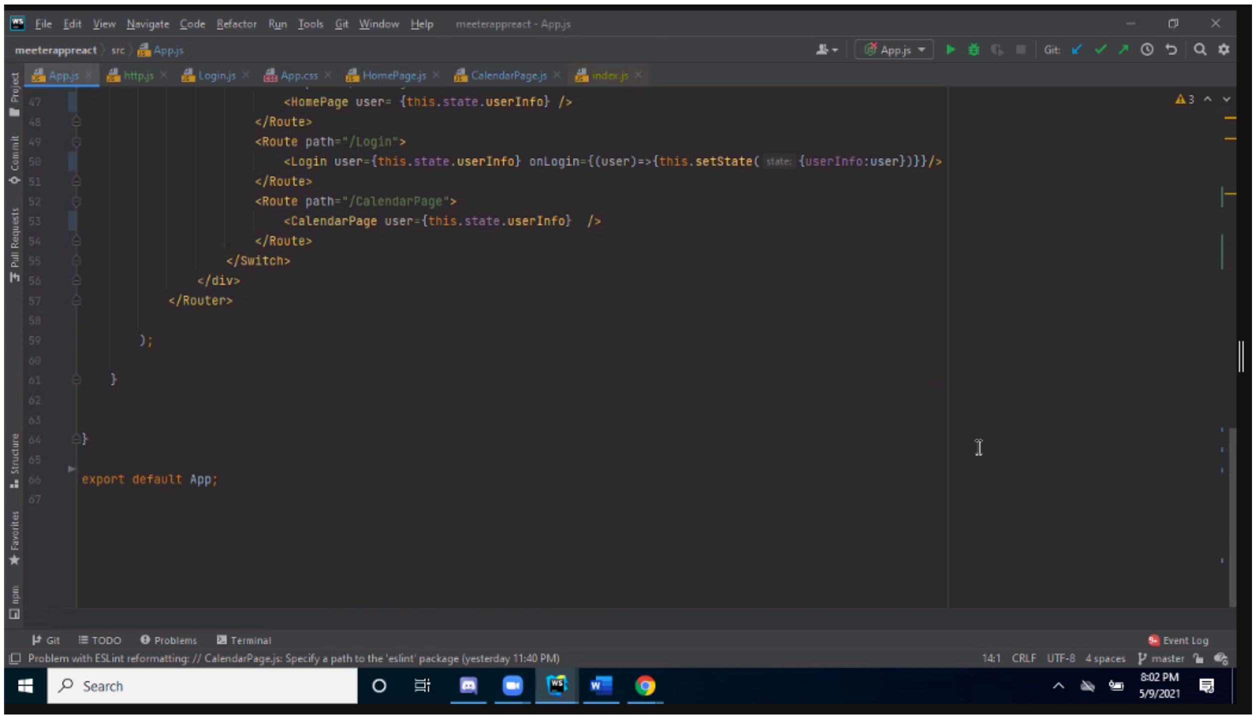
```
import './App.css';
import React, {Component}  from 'react';
import Login from '../src/Login.js';
import HomePage from '../src/HomePage.js';
import ApiCalendar from 'react-google-calendar-api';
import CalendarPage from '../src/CalendarPage.js';
import {
    BrowserRouter as Router,
    Switch,
    Route,
    Link } from "react-router-dom";

import http from '../src/utils/http.js';




class App extends React.Component{

    constructor(props){
        super(props);
        this.state = {userInfo:JSON.parse(localStorage.getItem( key: 'user'))};

    }

    render(): React.Node {
```

Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)



```

    render(): React.Node {
        return (

            <Router>
                <div>
                    <ul>
                        <li>
                            <Link to="/HomePage">Home</Link>
                        </li>
                        <li>
                            <Link to="/Login">Login</Link>
                        </li>
                        <li>
                            <Link to="/CalendarPage">Calendar</Link>
                        </li>
                    </ul>

                <hr/>

                    <Switch>
                        <Route path="/HomePage">
                            <HomePage user= {this.state.userInfo} />
                        </Route>
                        <Route path="/Login">
                            <Login user={this.state.userInfo} onLogin={(user)=>{this.setState( state: {userInfo:user})}}/>
```

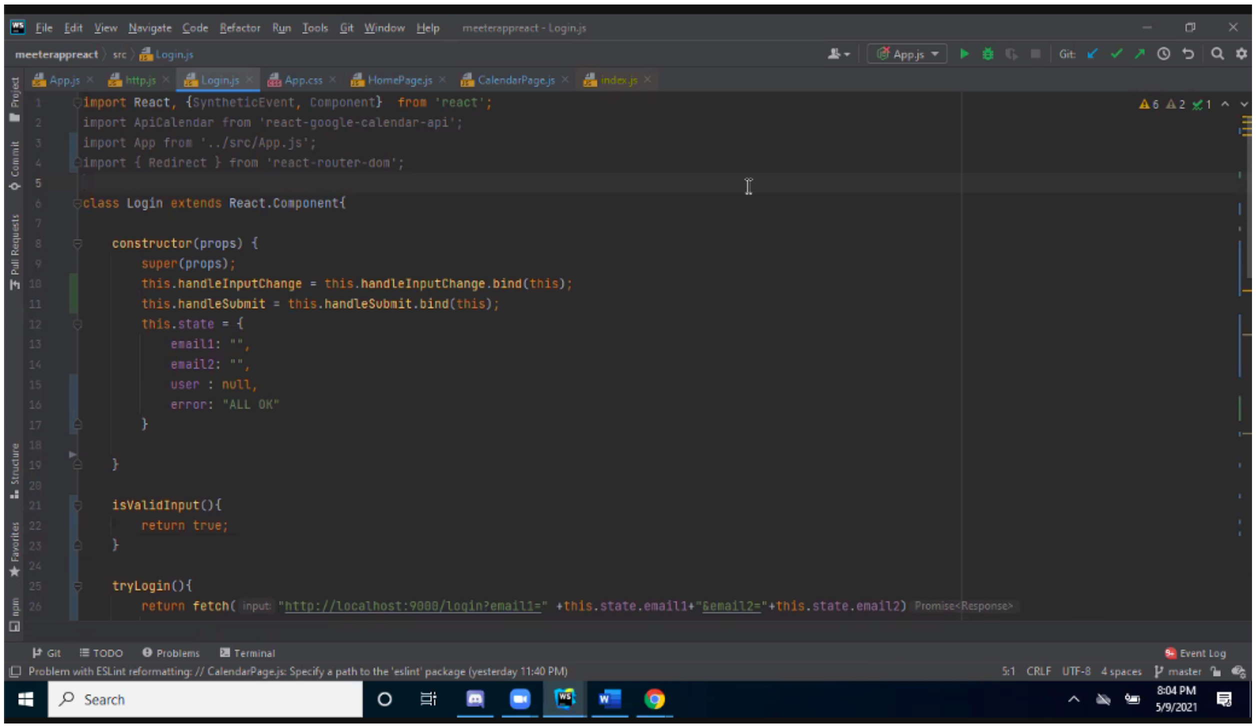
Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)



```
                <HomePage user= {this.state.userInfo} />
              </Route>
              <Route path="/Login">
                <Login user={this.state.userInfo} onLogin={(user)=>{this.setState( {userInfo:user})}}/>
              </Route>
              <Route path="/CalendarPage">
                <CalendarPage user={this.state.userInfo}  />
              </Route>
            </Switch>
          </div>
        </Router>

      );

  }


}

export default App;
```

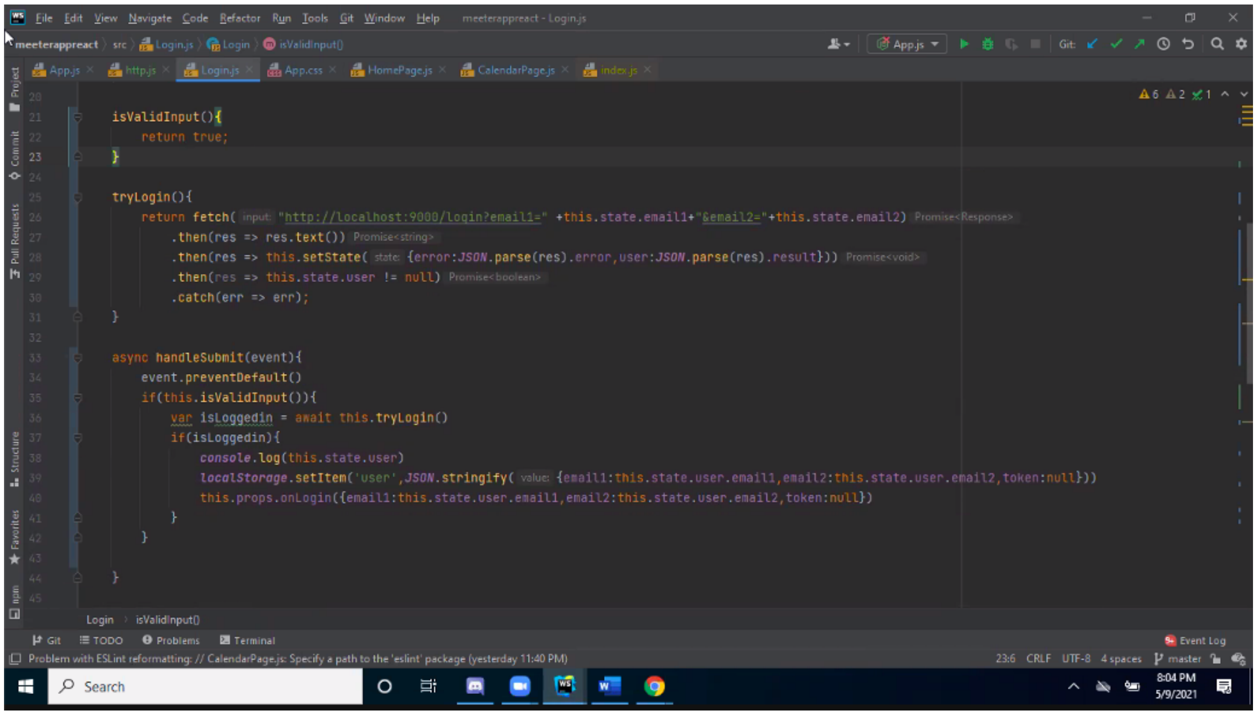
```
Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)
```

Next we created a login page where we included inputs for the user's email address(calendarIDs) and added functions to support them.



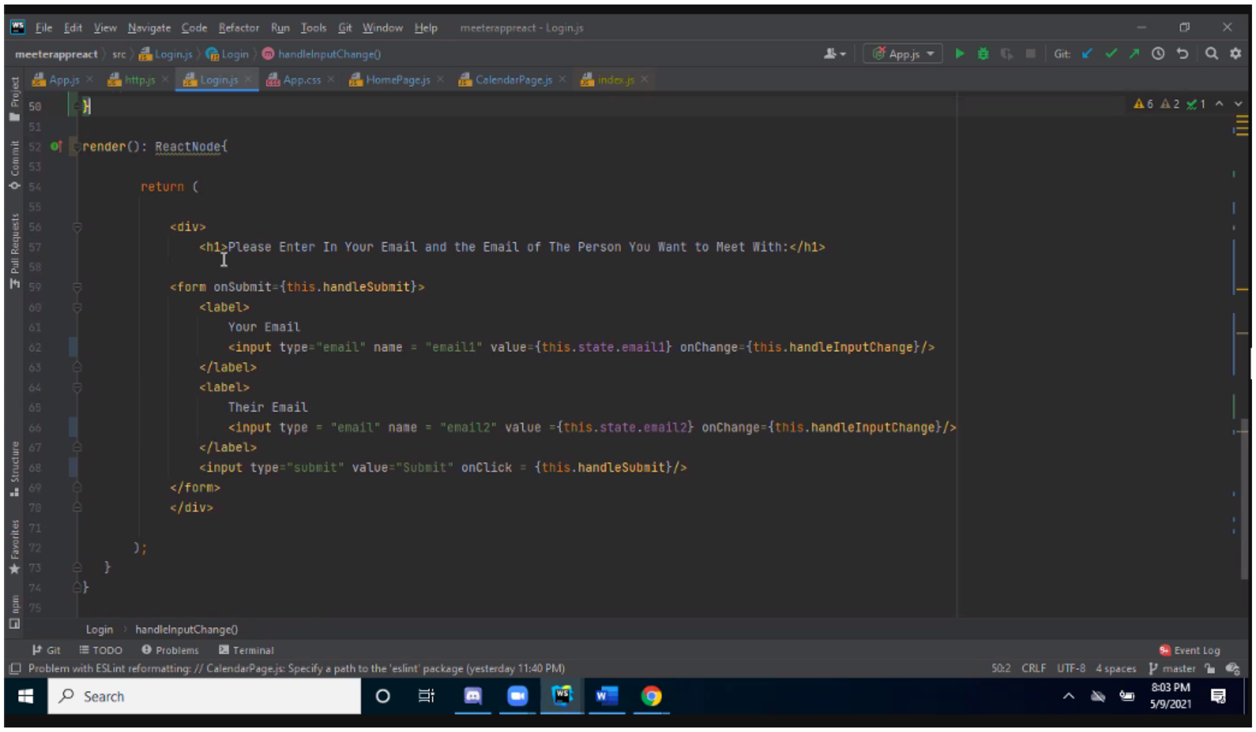
```
import React, {SyntheticEvent, Component}  from 'react';
import ApiCalendar from 'react-google-calendar-api';
import App from '../src/App.js';
import { Redirect } from 'react-router-dom';

class Login extends React.Component{

    constructor(props) {
        super(props);
        this.handleInputChange = this.handleInputChange.bind(this);
        this.handleSubmit = this.handleSubmit.bind(this);
        this.state = {
            email1: "",
            email2: "",
            user : null,
            error: "ALL OK"
        }

    }

    isValidInput(){
        return true;
    }

    tryLogin(){
        return fetch( "http://localhost:9000/login?email1=" +this.state.email1+"&email2="+this.state.email2)
```

```
Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)
```



```
    isValidInput(){
        return true;
    }

    tryLogin(){
        return fetch( input: "http://localhost:9000/login?email1=" +this.state.email1+"&email2="+this.state.email2) Promise<Response>
            .then(res => res.text()) Promise<string>
            .then(res => this.setState( state: {error:JSON.parse(res).error,user:JSON.parse(res).result})) Promise<void>
            .then(res => this.state.user != null) Promise<boolean>
            .catch(err => err);
    }

    async handleSubmit(event){
        event.preventDefault()
        if(this.isValidInput()){
            var isLoggedin = await this.tryLogin()
            if(isLoggedin){
                console.log(this.state.user)
                localStorage.setItem('user',JSON.stringify( value: {email1:this.state.user.email1,email2:this.state.user.email2,token:null}))
                this.props.onLogin({email1:this.state.user.email1,email2:this.state.user.email2,token:null})
            }
        }

    }
```

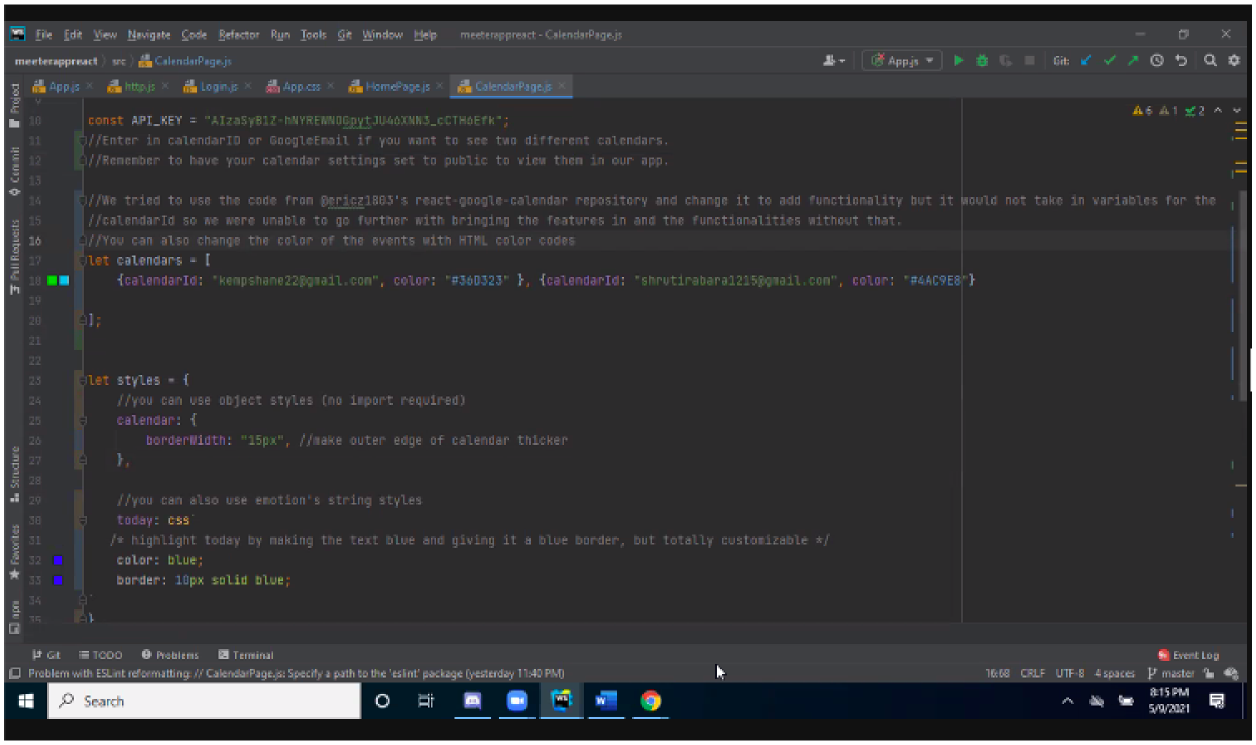
Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)



```
}

render(): ReactNode{

        return (

            <div>
                <h1>Please Enter In Your Email and the Email of The Person You Want to Meet With:</h1>

            <form onSubmit={this.handleSubmit}>
                <label>
                    Your Email
                    <input type="email" name = "email1" value={this.state.email1} onChange={this.handleInputChange}/>
                </label>
                <label>
                    Their Email
                    <input type = "email" name = "email2" value ={this.state.email2} onChange={this.handleInputChange}/>
                </label>
                <input type="submit" value="Submit" onClick = {this.handleSubmit}/>
            </form>
            </div>

        );
    }
}
```

Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)

Since most of our trial and error code was in the Calendar page, we did not have much changes, nonetheless, we added comments in the code explaining exactly what obstacles we faced.



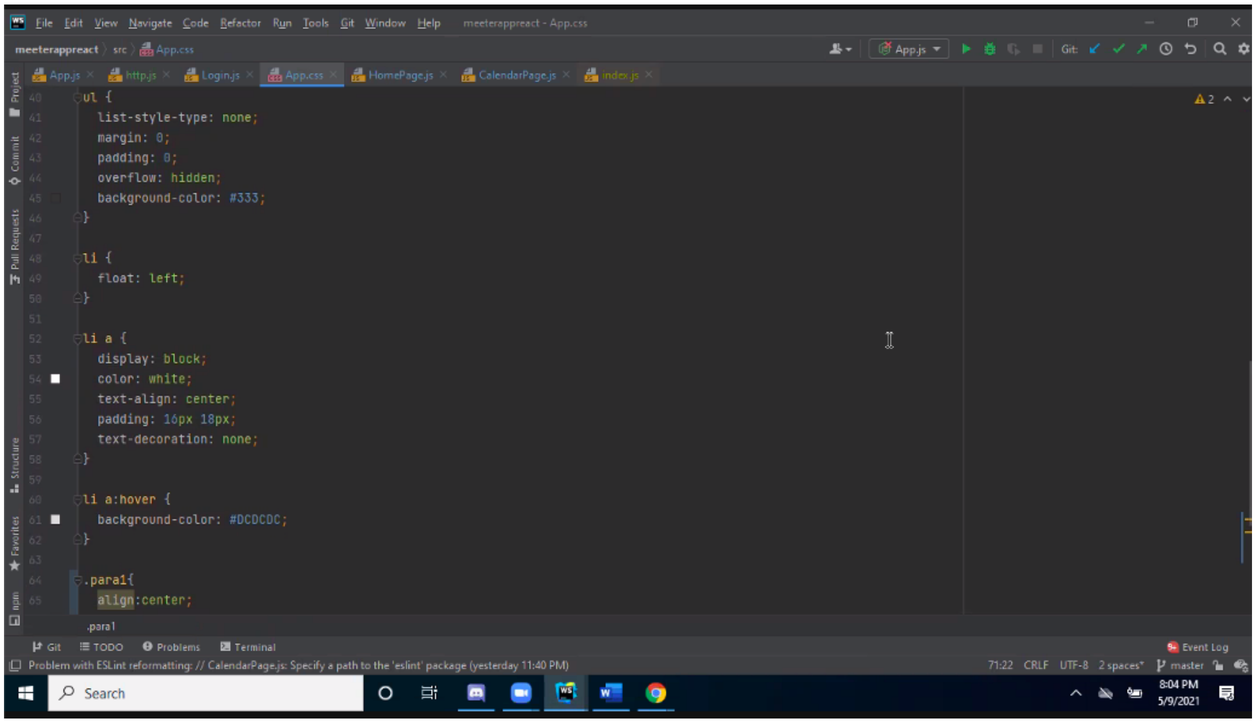
```
const API_KEY = "AIzaSyBlZ-hNYREWN0GpytJU40XNN3_cCTH6Efk";
//Enter in calendarID or GoogleEmail if you want to see two different calendars.
//Remember to have your calendar settings set to public to view them in our app.

//We tried to use the code from @ericz1803's react-google-calendar repository and change it to add functionality but it would not take in variables for the
//calendarId so we were unable to go further with bringing the features in and the functionalities without that.
//You can also change the color of the events with HTML color codes
let calendars = [
    {calendarId: "kempshane22@gmail.com", color: "#360323" }, {calendarId: "shrutirabara1215@gmail.com", color: "#4AC9E8"}

];


let styles = {
    //you can use object styles (no import required)
    calendar: {
        borderWidth: "15px", //make outer edge of calendar thicker
    },

    //you can also use emotion's string styles
    today: css`
   /* highlight today by making the text blue and giving it a blue border, but totally customizable */
    color: blue;
    border: 10px solid blue;
  `
}
```

Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)

File Edit View Navigate Code Refactor Run Tools Git Window Help
meeterappreact - CalendarPage.js
meeterappreact src CalendarPage.js
App.js http.js Login.js App.css HomePage.js CalendarPage.js
Git TODO Problems Terminal
Event Log
16:68 CRLF UTF-8 4 spaces master
Search
8:15 PM 5/9/2021

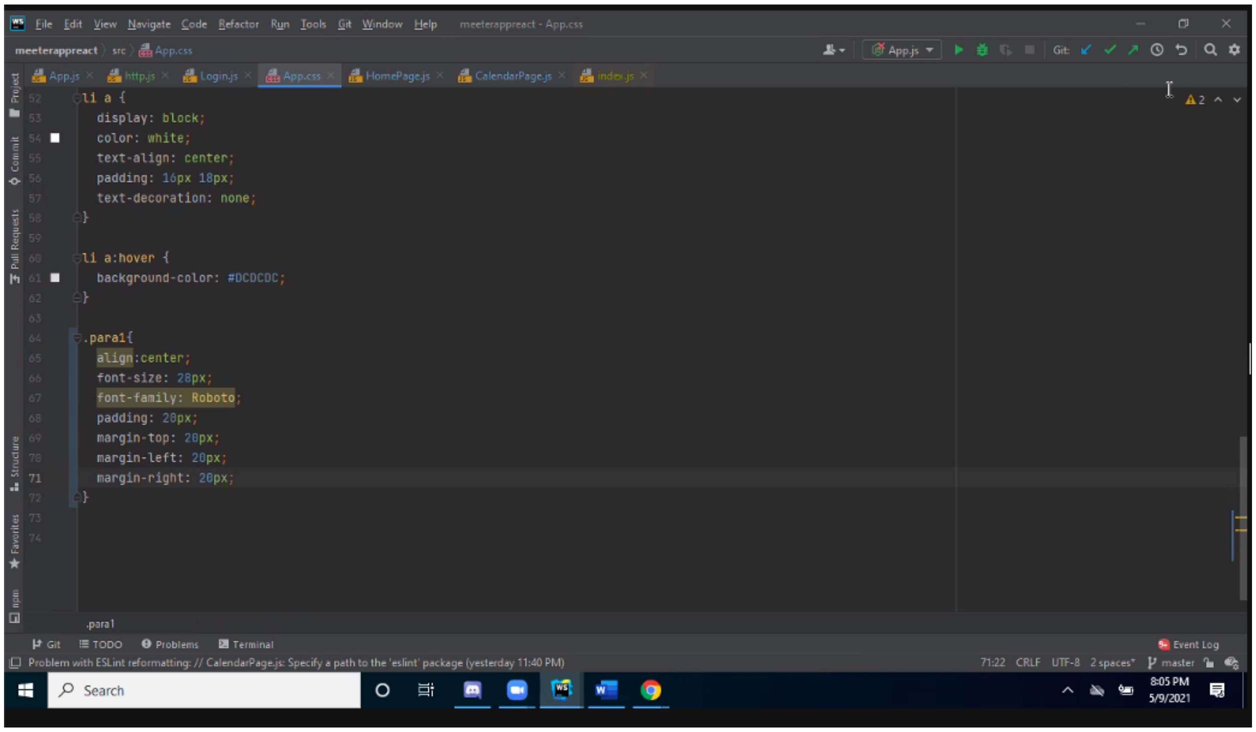
Lastly, we made minor modifications to our CSS to adjust the layout of the navigation bar.

Meeter


File Edit View Navigate Code Refactor Run Tools Git Window Help meeterappreact - App.css
meeterappreact src App.css
App.js
App.js http.js Login.js App.css HomePage.js CalendarPage.js index.js
.para1
Git TODO Problems Terminal Event Log
71:22 CRLF UTF-8 2 spaces* master
Search
8:04 PM
5/9/2021

```css
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px 18px;
  text-decoration: none;
}

li a:hover {
  background-color: #DCDCDC;
}

.para1{
  align:center;
```

Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)


File Edit View Navigate Code Refactor Run Tools Git Window Help meeterappreact - App.css
meeterappreact src App.css
App.js
App.js http.js Login.js App.css HomePage.js CalendarPage.js index.js
.para1
Git TODO Problems Terminal Event Log
71:22 CRLF UTF-8 2 spaces* master
Search
8:05 PM
5/9/2021

```css
li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px 18px;
  text-decoration: none;
}

li a:hover {
  background-color: #DCDCDC;
}

.para1{
  align:center;
  font-size: 28px;
  font-family: Roboto;
  padding: 20px;
  margin-top: 20px;
  margin-left: 20px;
  margin-right: 20px;
}
```

Problem with ESLint reformatting: // CalendarPage.js: Specify a path to the 'eslint' package (yesterday 11:40 PM)

5. Discussions and Conclusions

There were many challenges we encountered during the building of this project. We were unable to get the Google Calendar API to work in a regular JavaScript file with HTML5 and CSS so we switched to ReactJS, hoping that it would rectify the issues. Unfortunately, that came with its own learning curve and a new set of problems. The APIs for ReactJS did not have the same functionality integrated and unfortunately would only let us hard code a string into the calendar ID because of how the parameters of the code were set up in the node modules. Even with trying to change the code parameters many times and working for countless hours on trying to get the calendar to accept the user's input, we were unsuccessful in getting the input to the calendar ID so the calendars could be compared. Additionally, we spend the rest of the time understanding how to utilize the API function that we imported into our js application, however, due to the time constraint, we were unable to correctly utilize the functions to successfully complete some of our intended tasks.

We ultimately decided that due to time constraints, we would have to keep what we have and try to fix small things that didn't interrupt the functionality that we did have of the code.