```
MIT License

Copyright (c) 2021 Nirav Koley

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

```

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Runtime data
pids
*.pid
*.seed
*.pid.lock

# Directory for instrumented libs generated by jscoverage/JSCover
lib-cov

# Coverage directory used by tools like istanbul
coverage

# nyc test coverage
.nyc_output

# Grunt intermediate storage (https://gruntjs.com/creating-plugins#
storing-task-files)
.grunt

# Bower dependency directory (https://bower.io/)
bower_components

# node-waf configuration
.lock-wscript

# Compiled binary addons (https://nodejs.org/api/addons.html)
build/Release

# Dependency directories
node_modules/
jspm_packages/

# TypeScript v1 declaration files
typings/

# Optional npm cache directory
.npm

# Optional eslint cache
.eslintcache

# Optional REPL history
.node_repl_history

# Output of 'npm pack'
*.tgz

# Yarn Integrity file
.yarn-integrity

```

```
# dotenv environment variables file
.env

# parcel-bundler cache (https://parceljs.org/)
.cache

# next.js build output
.next

# nuxt.js build output
.nuxt

# vuepress build output
.vuepress/dist

# Serverless directories
.serverless

# FuseBox cache
.fusebox/

```

```
1 # Auto detect text files and perform LF normalization
2 * text=auto
3
```

```yaml
# Compose the client and server containers into a single docker instance
version: '3'

services:
  backend:
    env_file:
        "./server/.env"
    build:
      context: ./server
      dockerfile: ./Dockerfile
    ports:
      - "5000:5000"
  frontend:
    build:
      context: ./client
      dockerfile: ./Dockerfile
    ports:
      - "3000:3000"
    links:
      - "localhost:server"
```

```
 1 # Getting Started with Create React App
 2
 3 This project was bootstrapped with [Create React App](https://github.
   com/facebook/create-react-app).
 4
 5 ## Available Scripts
 6
 7 In the project directory, you can run:
 8
 9 ### `npm start`
10
11 Runs the app in the development mode.\
12 Open [http://localhost:3000](http://localhost:3000) to view it in the
   browser.
13
14 The page will reload if you make edits.\
15 You will also see any lint errors in the console.
16
17 ### `npm test`
18
19 Launches the test runner in the interactive watch mode.\
20 See the section about [running tests](https://facebook.github.io/
   create-react-app/docs/running-tests) for more information.
21
22 ### `npm run build`
23
24 Builds the app for production to the `build` folder.\
25 It correctly bundles React in production mode and optimizes the build
   for the best performance.
26
27 The build is minified and the filenames include the hashes.\
28 Your app is ready to be deployed!
29
30 See the section about [deployment](https://facebook.github.io/create-
   react-app/docs/deployment) for more information.
31
32 ### `npm run eject`
33
34 **Note: this is a one-way operation. Once you `eject`, you can’t go
   back!**
35
36 If you aren’t satisfied with the build tool and configuration choices
   , you can `eject` at any time. This command will remove the single
   build dependency from your project.
37
38 Instead, it will copy all the configuration files and the transitive
   dependencies (webpack, Babel, ESLint, etc) right into your project so
   you have full control over them. All of the commands except `eject`
   will still work, but they will point to the copied scripts so you can
   tweak them. At this point you’re on your own.
39
40 You don’t have to ever use `eject`. The curated feature set is
   suitable for small and middle deployments, and you shouldn’t feel
   obligated to use this feature. However we understand that this tool
   wouldn’t be useful if you couldn’t customize it when you are ready for
    it.
41
```

```
## Learn More

You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/getting-started).

To learn React, check out the [React documentation](https://reactjs.org/).

### Code Splitting

This section has moved here: [https://facebook.github.io/create-react-app/docs/code-splitting](https://facebook.github.io/create-react-app/docs/code-splitting)

### Analyzing the Bundle Size

This section has moved here: [https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size](https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size)

### Making a Progressive Web App

This section has moved here: [https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app](https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app)

### Advanced Configuration

This section has moved here: [https://facebook.github.io/create-react-app/docs/advanced-configuration](https://facebook.github.io/create-react-app/docs/advanced-configuration)

### Deployment

This section has moved here: [https://facebook.github.io/create-react-app/docs/deployment](https://facebook.github.io/create-react-app/docs/deployment)

### `npm run build` fails to minify

This section has moved here: [https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify](https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify)
```

```
1 # Docker file for deploying front end
2
3 FROM node:14.17.4
4
5 WORKDIR /app
6
7 COPY package*.json ./
8
9 RUN npm i
10
11 COPY . .
12
13 EXPOSE 3000
14
15 CMD ["npm", "start"]
16
```

```
{
  "name": "react-fe-counsweb",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.14.1",
    "@testing-library/react": "^11.2.7",
    "@testing-library/user-event": "^12.8.3",
    "axios": "^0.21.4",
    "bootstrap": "^4.6.0",
    "dayjs": "^1.10.6",
    "react": "^17.0.2",
    "react-bootstrap": "^1.6.1",
    "react-calendar": "^3.4.0",
    "react-dom": "^17.0.2",
    "react-icons": "^4.2.0",
    "react-router-dom": "^5.2.0",
    "react-scripts": "4.0.3",
    "web-vitals": "^1.1.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

```
1 node_modules
2
```

```jsx
// Client side-routing for serving web pages by rendering components

import React, { useState } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import FormPage from './components/FormPage/FormPage';
import EditBookings from './components/AdminPage/bookings/EditBookings';
import EditCalendar from './components/AdminPage/calendar/EditCalendar';
import EditEmail from './components/AdminPage/emails/EditEmail';
import Support from './components/Support';
import { BrowserRouter, Route, Switch } from 'react-router-dom';
import Sidebar2 from './components/Sidebar2';
import PrivateRoute from './PrivateRoute';

function App() {
    const [admin, setAdmin] = useState(false)
    return (
        <>
            <BrowserRouter>
                <Sidebar2 admin={admin} setAdmin={setAdmin}/>
                <Switch>
                    <Route path='/' exact component={FormPage} />
                    <Route path='/support' exact component={Support} />

                    {/*Private routes are blocked until admin access is granted*/}
                    <PrivateRoute admin={admin} path='/admin/booking' exact component={EditBookings} />
                    <PrivateRoute admin={admin} path='/admin/calendar' exact component={EditCalendar} />
                    <PrivateRoute admin={admin} path='/admin/emails' exact component={EditEmail} />
                </Switch>
            </BrowserRouter>
        </>
    );
}

export default App;

```

```
 1 import React from 'react';
 2 import ReactDOM from 'react-dom';
 3 import './index.css';
 4 import App from './App';
 5
 6 ReactDOM.render(
 7     <React.StrictMode>
 8         <App />
 9     </React.StrictMode>,
10     document.getElementById('root')
11 );
12
```

```css
.formAlign {
    padding: 2em 1em 2em 1em;
    width: 40rem;
    max-width: 100%;
    margin: auto;
}

.calendarAlign {
    padding: 1em 1em 3em 1em;
    width: 38rem;
    max-width: 100%;
    margin: auto;
}

.timeBox {
    text-align: center;
    position: relative;
    width: 38rem;
    max-width: 100%;
    padding: 1rem;
    margin: auto;

    border: 0.3rem solid #ececec;
    border-radius: 8px;
    color: #212529;
}

.submitAlign {
    text-align: center;
    position: relative;
    width: 38rem;
    max-width: 100%;
    padding: 1rem;
    margin: auto;
}

.timeTitle {
    /* text-decoration: underline; */
    font-size: 1.5em;
    font-weight: normal;
    text-decoration-thickness: 0.1rem;
}

.localTime {
    padding-top: 0.5rem;
}

.localTimeTitle {
    text-decoration: underline;
    font-size: 1.3em;
    font-weight: lighter;
    text-decoration-thickness: 0.1rem;
}

.ourTime {
    padding-top: 0.5rem;
}
```

```
 58
 59 .ourTimeTitle {
 60     text-decoration: underline;
 61     font-size: 1.3em;
 62     font-weight: lighter;
 63     text-decoration-thickness: 0.1rem;
 64 }
 65
 66 .alert-success {
 67     color: #0f5132;
 68     background-color: #d1e7dd;
 69     border-color: #badbcc;
 70     position: relative;
 71     padding: 1rem;
 72     margin: 2.5rem;
 73     border: 1px solid transparent;
 74     border-radius: 0.25rem;
 75     font-size: 16px;
 76     font-weight: normal;
 77     text-align: center;
 78     visibility: visible !important;
 79 }
 80
 81 .hidden {
 82     visibility: hidden;
 83 }
 84
 85 .button {
 86     padding: 2em 1em 2.5em 1em;
 87     width: 30rem;
 88     height: 4rem;
 89     max-width: 100%;
 90     margin: auto;
 91 }
 92
 93 .center {
 94     align-items: center;
 95 }
 96
 97 .disableCalendar {
 98     cursor: default !important;
 99 }
100
101 .container {
102     overflow: hidden;
103 }
104
105 .unavailable {
106     color: #000 !important;
107     background-color: rgba(242, 242, 240, 0.45) !important;
108 }
109
110 .past {
111     color: #000 !important;
112     background-color: rgba(232, 232, 225, 0.6) !important;
113 }
114
```

```
.booked:hover {
    color: hsla(3, 100%, 20%, 1) !important;
    background-color: rgba(255, 65, 54, 0.55) !important;
}

.booked {
    color: hsla(3, 100%, 20%, 1) !important;
    background-color: rgba(255, 65, 54, 0.6) !important;
}

.regionBooked {
    color: hsla(350, 65%, 15%) !important;
    background-color: rgba(194, 43, 65, 0.6) !important;
}

.regionBooked:hover {
    color: hsla(350, 65%, 15%) !important;
    background-color: rgba(194, 43, 65, 0.55) !important;
}

.available {
    color: hsla(127, 63%, 15%, 1) !important;
    background-color: rgba(1, 255, 112, 0.7) !important;
}

.available:hover {
    color: hsla(127, 63%, 15%, 1) !important;
    background-color: rgba(1, 255, 112, 0.5) !important;
}

.selected {
    color: #111111 !important;
    background-color: rgba(13, 110, 253 0.85) !important;
}

.selected:hover {
    color: #111111 !important;
    background-color: rgba(13, 110, 253 0.8) !important;
}

.invalidCalendar {
    transition: border-color 0.15s ease-in-out, box-shadow 0.15s ease
-in-out;
    border-radius: 0.25rem;
    border: 1.5px solid #dc3545;
    padding: 0.5em;
}

.invalidText {
    width: 100%;
    margin-top: 0.3rem;
    margin-left: 0;
    margin-right: 0;
    font-size: 0.875em;
    color: #dc3545;
    display: block;
    text-align: center;
```

```
171 }
172
173 .invalidText:hover {
174     cursor: auto;
175 }
176
177 .support {
178     text-align: center;
179     position: relative;
180     width: 38rem;
181     max-width: 100%;
182     padding: 15vh 1rem 1rem 1rem;
183     margin: auto;
184 }
```

```
// Private route component

import {Route, Redirect} from 'react-router-dom'

const PrivateRoute = ({component, path, admin}) => {
    if(admin){
        return(<Route path={path} exact component={component}/>)
    } else {
        return(<Redirect to='/' />)
    }
}

export default PrivateRoute

```

```
 1 import { useState } from 'react'
 2 import Form from 'react-bootstrap/Form'
 3 import Modal from 'react-bootstrap/Modal'
 4 import Button from 'react-bootstrap/Button'
 5 import axios from 'axios';
 6
 7 const Login = ({show,cancel,setShow,error,setError,setAdmin}) => {
 8     const [loginfo, setLoginfo] = useState({email:'',password:''})
 9
10
11     async function handleSubmit(){
12         await axios.post('http://localhost:5000/login/login', loginfo)
13         .then((res)=>{
14             if(res.data === "FALSE"){
15                 setError(true)
16
17             } else{
18                 setAdmin(true)
19                 localStorage.setItem('adminToken', res.data.adminToken
   ) // Store admin token in local storage
20                 setShow(false)
21             }
22         })
23         .catch(err =>{
24             console.log(err)
25         });
26     }
27
28     return (
29         <Modal show={show} onHide={cancel}>
30             <Modal.Header closeButton>
31                 <Modal.Title>Login</Modal.Title>
32             </Modal.Header>
33             <Modal.Body>
34             <Form>
35                 <Form.Group>
36                     <Form.Label> Email </Form.Label>
37                     <Form.Control type='text' onChange={(e) => {
   setLoginfo({...loginfo,email:e.target.value})}}/>
38                 </Form.Group>
39                 <Form.Group>
40                     <Form.Label> Password </Form.Label>
41                     <Form.Control type='password' onChange={(e) => {
   setLoginfo({...loginfo,password:e.target.value})}}/>
42                 </Form.Group>
43                 {error?
44                 <div className="incorrect">
45                     Incorrect Password/Email
46                 </div>
47                 :<div></div>
48                 }
49             </Form>
50             </Modal.Body>
51             <Modal.Footer>
52                 <Button onClick={()=>{handleSubmit()}}>Submit</Button>
53             </Modal.Footer>
54         </Modal>
```

```
55     )
56 }
57
58 export default Login
59
```

```
 1 import "../index.css"
 2
 3 const Support = () => {
 4     return (
 5       <div className="support">
 6         If you have any questions please call or text:
 7         <h6>+1 713 290 9025</h6>
 8         <br />
 9         or email:
10         <h6>Paula.Cooper@houston.nae.school </h6>
11       </div>
12     )
13 }
14
15 export default Support
16
```

```
// Sidebar for client side routing

import React, { useEffect, useState } from 'react';
import { BsPencilSquare } from 'react-icons/bs';
import { FaCalendarPlus, FaCalendarAlt,  } from 'react-icons/fa'
import { AiFillLock, AiFillUnlock, AiOutlineQuestionCircle } from 'react-icons/ai';
import { MdEmail } from 'react-icons/md'
import { Link, NavLink } from 'react-router-dom';
import './sidebar2.css';
import Navbar from 'react-bootstrap/Navbar'
import {Nav, NavDropdown} from 'react-bootstrap';
import Login from './Login';
import axios from 'axios';

const Sidebar2 = ({admin,setAdmin}) => {
    const [sidebar, setSidebar] = useState(false); // Toggle sidebar visibility
    const toggleSidebar = () => setSidebar(!sidebar); // Seperate function as bugfix

    const [show, setShow] = useState(false);
    const [error, setError] = useState(false);

    useEffect(async () => {
        const token = localStorage.getItem('adminToken')
        if(!!token){
            const verify = await axios.post('http://localhost:5000/login/verify', {adminToken:token})
            if(verify){
                setAdmin(true)
            }
        }
    }, [])

    function isAdmin(){
        if(admin){
            return(
                <NavDropdown title={<span><AiFillUnlock size={22}/> Admin</span>} className="nav-text">
                    <NavDropdown.Item href="#edit">
                        <Nav.Link as={NavLink} to='/admin/booking' className='nopad'>
                            <span><FaCalendarPlus size={20}/> Bookings </span>
                        </Nav.Link>
                    </NavDropdown.Item>

                    <NavDropdown.Item href="#dates">
                        <Nav.Link as={NavLink} to='/admin/calendar' className='nopad'>
                            <span><FaCalendarAlt size={20}/> Calendar </span>
                        </Nav.Link>
                    </NavDropdown.Item>

                    <NavDropdown.Item href="#email">
```

```
                            <Nav.Link as={NavLink} to='/admin/emails' className='nopad'>
                                <span><MdEmail size={20}/> Emails</span>
                            </Nav.Link>
                        </NavDropdown.Item>
                    </NavDropdown>
                )
            } else {
                return(
                    <Nav.Link onClick={()=>{setShow(true);setError(false)}} className="nav-text">
                        <span><AiFillLock size={22}/> Admin</span>
                    </Nav.Link>
                )
            }
        }

        function cancel(){
            setShow(false)
        }

        return (
            <>
            <div className='wCube'></div>
            <Login show={show} cancel={cancel} setShow={setShow} error={error} setError={setError} setAdmin={setAdmin}/>
            <div>
                <Navbar bg="light">
                    <Navbar.Brand className='bars' as={Link} onClick={()=>{toggleSidebar()}}>
                        <div className='burgerpad'>
                            <div className={!sidebar?'burger':'burger-close'}></div>
                        </div>
                    </Navbar.Brand>
                        <Nav className={sidebar?"me-auto sidebar-active":"me-auto sidebar-inactive"}>
                        <Nav.Link as={NavLink} to='/' className="nav-text">
                                <span className='nav-icon'><BsPencilSquare size={22}/> Form</span>
                        </Nav.Link>
                        {isAdmin()}
                        <Nav.Link as={NavLink} to='/support' className="nav-text">
                                <span><AiOutlineQuestionCircle size={22}/> Help</span>
                        </Nav.Link>
                    </Nav>
                </Navbar>
            </div>
            </>
        )
}

export default Sidebar2

```

```css
.nopad{
    padding: 0 0 0 0 !important;
}

.wCube{
    position: absolute;
    top:0;
    left:0;
    height:64px;
    width:64px;
    background-color: #f8f9fa;
    z-index: 1;
}

.bars{
    z-index: 2;
    margin-right: 0;
}

.sidebar-active{
    left: 0rem;
    position: relative;
    transition: 500ms;
}

.sidebar-inactive{
    left: -20rem;
    position: absolute;
    transition: 500ms;
}

.burger{
    width:40px;
    height: 6px;
    background: rgba(0,0,0,0.8);
    border-radius: 3px;
    box-shadow: 0 2px 2px rgba(255,101,47,.2);
    transition: all .5s ease-in-out;
}

.burger::before,.burger::after{
    content:'';
    position:absolute;
    width:40px;
    height: 6px;
    background: rgba(0,0,0,0.8);
    border-radius: 3px;
    box-shadow: 0 2px 2px rgba(255,101,47,.2);
    transition: all .5s ease-in-out;
}

.burger::before {
    transform: translateY(-12px);
}

.burger::after {
    transform: translateY(12px);
```

```
 58 }
 59
 60 .burger-close::before,.burger-close::after{
 61     content:'';
 62     position:absolute;
 63     width:40px;
 64     height: 6px;
 65     background: rgba(0,0,0,0.8);
 66     border-radius: 3px;
 67     box-shadow: 0 2px 2px rgba(255,101,47,.2);
 68     transition: all .5s ease-in-out;
 69 }
 70
 71 .burger-close{
 72     transform: translateX(-50px);
 73     background: transparent;
 74     box-shadow: none;
 75     width: 40px;
 76     height: 6px;
 77     border-radius: 3px;
 78     transition: all .5s ease-in-out;
 79 }
 80
 81 .burger-close::before{
 82     transform: translateY(-12px);
 83     transform: rotate(45deg) translate(35px, -35px);
 84 }
 85 .burger-close::after{
 86     transform: translateY(12px);
 87     transform: rotate(-45deg) translate(35px, 35px);
 88 }
 89
 90 .burgerpad{
 91     margin:1rem;
 92     margin-left:0.5rem !important;
 93 }
 94
 95 .nav-text{
 96     font-size: 1.3rem;
 97     margin-right: 1rem;
 98     margin-left:0.35rem;
 99 }
100
101 .nav-div{
102     display: flex;
103     justify-content: center;
104 }
105
106 .incorrect{
107     color: #dc3545;
108     font-size: 80%;
109 }
110
```

```
// Main form page, a sort of hub for many components to mount to and render from

import React, { useState, useEffect } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import Button from 'react-bootstrap/Button';
import TextForm from './TextForm';
import TimeDisplay from './TimeDisplay';
import DateCalendar from './DateCalendar';
import axios from 'axios';
import dayjs from 'dayjs';
import isBetween from 'dayjs/plugin/isBetween';
import validateInfo from './validateForm';

dayjs.extend(isBetween);

// Initialize calendar states
let available = [];
let booked = [];
let regionBooked = [];

function FormPage() {
    let calendarData;

    const [checkDate, setCheckDate] = useState(false); // Used to manually re-render after validation
    const [key, setKey] = useState(false); // Used to manually re-render after validation

    const [validated, setValidated] = useState(false); // Used to prevent validation tooltips from appearing before submission
    const [error, setError] = useState({}); // Array of validation errors
    const [success, setSuccess] = useState(false);
    const [dateData, setDateData] = useState(new Date()); // Store appointment date
    const [formData, setFormData] = useState({ // Store booking information
        uniName: '',
        uniRepName: '',
        uniRepJobTitle: '',
        uniRepEmail: '',
        uniRegion: '',
    });


    // Submission handler
    const handleSubmit = (e) => {
        e.preventDefault();

        const bookingDateInfoInstance = {
            aptDate: dateData,
            status: 'Booked',
            booking: {
                uniName: formData.uniName,
                uniRepName: formData.uniRepName,
                uniRepJobTitle: formData.uniRepJobTitle,
```

```
51                  uniRepEmail: formData.uniRepEmail,
52                  uniRegion: formData.uniRegion,
53              },};
54
55          setCheckDate(!checkDate); // Manually re-render before
    submission to trigger validation
56          if (Object.keys(validateInfo(bookingDateInfoInstance)).length
     === 0) {
57              localStorage.setItem('lastDate', new Date())
58              setSuccess(true); // Lock input fields
59              axios.post('http://localhost:5000/booking/create',
    bookingDateInfoInstance);
60
61          } else {
62              setError(validateInfo(bookingDateInfoInstance));
63              setValidated(true); // Trigger validation styles
64          }
65      };
66
67      useEffect(()=>{
68          let lastDate = localStorage.getItem('lastDate')
69          if(dayjs(lastDate).isAfter(dayjs(new Date()).subtract(3, '
    month'))){
70              setSuccess(true);
71          }
72      }, [])
73
74      // eslint-disable-next-line
75      useEffect(async () => {
76          if (formData.uniRegion !== '') { // Prevent render on mount
77              async function getRegionData() { // Async to prevent
    stalling
78                  let response = await axios.get('http://localhost:5000
    /booking/readforform');
79                  return response;
80              }
81
82              // eslint-disable-next-line
83              calendarData = (await getRegionData()).data;
84
85              available = [];
86              booked = [];
87              regionBooked = [];
88
89              // console.log(instance.booking.uniRegion)
90              console.log(calendarData)
91
92              calendarData.forEach((instance, index) => {
93                  if (instance.status === "Booked" && instance.booking.
    uniRegion === formData.uniRegion) {
94                      booked.push(dayjs(instance.aptDate).format('DD/MM
    /YYYY'));
95
96                      // Lower bound of the week
97                      let lowB = dayjs(instance.aptDate)
98                          .add(7, 'day')
99                          .format('DD/MM/YYYY');
```

```
100
101                     // Upper bound of the week
102                     let uppB = dayjs(instance.aptDate)
103                         .subtract(7, 'day')
104                         .format('DD/MM/YYYY');
105
106                     if (index - 1 >= 0) {
107                         // Checks last available day
108                         if (calendarData[index - 1].status === '
    Available' && dayjs(calendarData[index - 1])
109                         .isBetween(lowB,uppB,null,'[]')) {
110                             regionBooked.push(dayjs(calendarData[
    index - 1].aptDate).format('DD/MM/YYYY'));
111                             calendarData[index - 1].status = 'Region
    booked'; // Prevent further formatting
112                         }
113                     }
114
115                     if (index + 1 <= calendarData.length - 1) {
116                         // Checks next available day
117                         if (calendarData[index + 1].status === '
    Available' && dayjs(calendarData[index + 1])
118                         .isBetween(lowB,uppB,null,'[]')) {
119                             regionBooked.push(dayjs(calendarData[
    index + 1].aptDate).format('DD/MM/YYYY'));
120                             calendarData[index + 1].status = 'Region
    booked';
121                         }
122                     }
123                 }
124             });
125
126             calendarData.forEach((instance) => {
127                 if (instance.status === 'Available') {
128                     available.push(dayjs(instance.aptDate).format('DD
    /MM/YYYY'));
129
130                 } else if (instance.status === 'Booked') {
131                     booked.push(dayjs(instance.aptDate).format('DD/MM
    /YYYY'));
132                 }
133             });
134
135             // Deselect current date if it became booked/region
    booked during selection
136             if (booked.includes(dayjs(dateData).format('DD/MM/YYYY'
    )) || regionBooked.includes(dayjs(dateData).format('DD/MM/YYYY'))) {
137                 setDateData(new Date());
138             }
139
140             setKey(!key); // Manually re-render
141         }
142     }, [formData.uniRegion, checkDate]); // Dependencies which, after
     change, will trigger the useEffect
143
144
145     return (
```

```
146         <div className='container'>
147             <div className='row align-items-center'>
148                 <div className='col-6'>
149                     <TextForm
150                         validated={validated}
151                         errors={error}
152                         isReadOnly={success}
153                         formData={formData}
154                         setFormData={setFormData}
155                     />
156                 </div>
157                 <div className='col-6'>
158                     <DateCalendar
159                         validated={validated}
160                         errors={error}
161                         available={available}
162                         booked={booked}
163                         regionBooked={regionBooked}
164                         success={success}
165                         selectedDate={dateData}
166                         setSelectedDate={setDateData}
167                     />
168                 </div>
169             </div>
170             <div className='row align-items-center'>
171                 <div className='col-6'>
172                     <TimeDisplay date={dateData} />
173                 </div>
174 
175                 <div className='col-6'>
176                     <div className='submitAlign'>
177                         <Button
178                             className={success ? 'disabled button' :
    'button'}
179                             as='input'
180                             type='submit'
181                             value='Submit'
182                             onClick={handleSubmit}
183                         />
184                     </div>
185                 </div>
186             </div>
187             <h5
188                 id='successSubmit'
189                 className={success ? 'alert-success' : 'hidden'}>
190                 Thank you for signing up for the BISH virtual
    presentation, you
191                 will receive an email shortly with the details of
    your booking.
192             </h5>
193         </div>
194     );
195 }
196 
197 export default FormPage;
198 
```

```
// All the text fields for the form page

import React from 'react';
import Form from 'react-bootstrap/Form';

const TextForm = ({
    validated,
    errors,
    isReadOnly,
    formData,
    setFormData
}) => {
    return (
        <div className='formAlign'>
            <Form validated={validated}>
                <Form.Group className='mb-3' controlId='form.University'>
                    <Form.Label>University</Form.Label>
                    <Form.Control
                        required
                        readOnly={isReadOnly ? true : false}
                        as='input'
                        type='text'
                        defaultValue={formData.uniName}
                        onChange={(e) => {
                            setFormData({...formData,uniName: e.target.value,});
                            delete errors.uniRepName;
                        }}
                    />
                    <Form.Control.Feedback type='invalid'>
                        {errors.uniName}
                    </Form.Control.Feedback>
                </Form.Group>

                <Form.Group className='mb-3' controlId='form.FullName'>
                    <Form.Label>Full name</Form.Label>
                    <Form.Control
                        required
                        readOnly={isReadOnly ? true : false}
                        as='input'
                        type='text'
                        defaultValue={formData.uniRepName}
                        onChange={(e) => {
                            setFormData({...formData,uniRepName: e.target.value,});
                            delete errors.uniRepName;
                        }}
                    />
                    <Form.Control.Feedback type='invalid'>
                        {errors.uniRepName}
                    </Form.Control.Feedback>
                </Form.Group>

                <Form.Group className='mb-3' controlId='form.JobTitle'>
```

```
                    <Form.Label>Job title</Form.Label>
                    <Form.Control
                        required
                        isValid={false}
                        readOnly={isReadOnly ? true : false}
                        as='input'
                        type='text'
                        defaultValue={formData.uniRepJobTitle}
                        onChange={(e) => {
                            setFormData({...formData,uniRepJobTitle:
e.target.value,});
                            delete errors.uniRepJobTitle;
                        }}
                    />
                    <Form.Control.Feedback type='invalid'>
                        {errors.uniRepJobTitle}
                    </Form.Control.Feedback>
                </Form.Group>

                <Form.Group className='mb-3' controlId='form.
EmailAddress'>
                    <Form.Label>Email address</Form.Label>
                    <Form.Control
                        required
                        readOnly={isReadOnly ? true : false}
                        as='input'
                        type='email'
                        defaultValue={formData.uniRepEmail}
                        onChange={(e) => {
                            setFormData({...formData,uniRepEmail: e.
target.value,});
                            delete errors.uniRepEmail;
                        }}
                    />
                    <Form.Control.Feedback type='invalid'>
                        {errors.uniRepEmail}
                    </Form.Control.Feedback>
                </Form.Group>

                <Form.Group className='mb-3' controlId='form.Region'>
                    <Form.Label>Region</Form.Label>
                    <Form.Control
                        required
                        disabled={isReadOnly ? true : false}
                        as='select'
                        defaultValue={formData.uniRegion}
                        onChange={(e) =>
                            setFormData({...formData,uniRegion: e.
target.value,})
                        }>
                        <option hidden={formData.uniRegion !== '' ?
true : false} selected='selected'></option>
                        <option value='USA'>USA</option>
                        <option value='CANADA'>Canada</option>
                        <option value='EUROPE'>Europe</option>
                        <option value='UNITED KINGDOM'>United Kingdom
</option>
```

```
                            <option value='OTHER'>Other</option>
                        </Form.Control>
                        <Form.Control.Feedback type='invalid'>
                            {errors.uniRegion}
                        </Form.Control.Feedback>
                    </Form.Group>
                </Form>
            </div>
        );
};

export default TextForm;

```

```css
abbr[title] {
    border-bottom: none !important;
    cursor: inherit !important;
    text-decoration: none !important;
}
.react-calendar {
    width: 38rem;
    max-width: 100%;
    background: white;
    /* border: 1px solid #a0a096; */
    font-family: 'system-ui';
    line-height: 1.125em;
}
.react-calendar--doubleView {
    width: 700px;
}
.react-calendar--doubleView .react-calendar__viewContainer {
    display: flex;
    margin: -0.5em;
}
.react-calendar--doubleView .react-calendar__viewContainer > * {
    width: 50%;
    margin: 0.5em;
}
.react-calendar,
.react-calendar *,
.react-calendar *:before,
.react-calendar *:after {
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
.react-calendar button {
    margin: 0;
    border: 0;
    outline: none;
}
.react-calendar button:enabled:hover {
    cursor: pointer;
}
.react-calendar__navigation {
    height: 44px;
    margin-bottom: 1em;
}
.react-calendar__navigation button {
    min-width: 44px;
    background: none;
}
.react-calendar__navigation button:enabled:hover,
.react-calendar__navigation button:enabled:focus {
    background-color: #e6e6e6;
}
/* .react-calendar__navigation button[disabled] {
  /* background-color: #f0f0f0; */
/*} */
.react-calendar__month-view__weekdays {
    text-align: center;
```

```
    text-transform: uppercase;
    font-weight: bold;
    font-size: 0.75em;
}
.react-calendar__month-view__weekdays__weekday {
    padding: 0.5em;
}
.react-calendar__month-view__weekNumbers {
    font-weight: bold;
}
.react-calendar__month-view__weekNumbers .react-calendar__tile {
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 0.75em;
    padding: calc(0.75em / 0.75) calc(0.5em / 0.75);
}
.react-calendar__month-view__days__day--weekend {
    color: #d10000;
}
.react-calendar__month-view__days__day--neighboringMonth {
    color: #757575;
}
.react-calendar__year-view .react-calendar__tile,
.react-calendar__decade-view .react-calendar__tile,
.react-calendar__century-view .react-calendar__tile {
    padding: 2em 0.5em;
}
.react-calendar__tile {
    max-width: 100%;
    text-align: center;
    padding: 0.75em 0.5em;
    background: none;
}
.react-calendar__tile:disabled {
    background-color: #f0f0f0;
}
.react-calendar__tile:enabled:hover,
.react-calendar__tile:enabled:focus {
    background-color: #e6e6e6;
}
.react-calendar__tile--now {
    background: #ffff76;
}
.react-calendar__tile--now:enabled:hover,
.react-calendar__tile--now:enabled:focus {
    background: #ffffa9;
}
.react-calendar__tile--hasActive {
    background: #76baff;
}
.react-calendar__tile--hasActive:enabled:hover,
.react-calendar__tile--hasActive:enabled:focus {
    background: #a9d4ff;
}
.react-calendar__tile--active {
    background: #006edc;
```

```
115     color: white;
116 }
117 .react-calendar__tile--active:enabled:hover,
118 .react-calendar__tile--active:enabled:focus {
119     background: #1087ff;
120 }
121 .react-calendar--selectRange .react-calendar__tile--hover {
122     background-color: #e6e6e6;
123 }
124
```

```
// Simple time display component

import React from 'react';
import dayjs from 'dayjs';
import utc from 'dayjs/plugin/utc';
import isToday from 'dayjs/plugin/isToday';
import advancedFormat from 'dayjs/plugin/advancedFormat';
import timezone from 'dayjs/plugin/timezone';

dayjs.extend(isToday);
dayjs.extend(utc);
dayjs.extend(timezone);
dayjs.extend(advancedFormat);

const TimeDisplay = ({ date }) => {
    let aptDate = dayjs(date).utc().hour(13).minute(30).second(0); // Sets date to predetermined appointment time
    let visible = dayjs(date).format('DD/MM/YYYY') === dayjs(new Date()).format('DD/MM/YYYY'); // Checks if a date has been selected

    return ( // A lot of the code here is just formatting
        <div className='timeBox'>
            <h1 className='timeTitle'>Your appointment is at</h1>
            <div className='localTime'>
                <h1 className='localTimeTitle'>Local time</h1>
                {visible ? dayjs(aptDate).local().format('h:mm A') : dayjs(aptDate).local().format('MMMM D, h:mm A')}
                <br />
                {'GMT' + dayjs(aptDate).local().format('Z (z)')}
            </div>
            <br />

            <div className='ourTime'>
                <h1 className='ourTimeTitle'>Our time</h1>
                {visible ? dayjs(aptDate).tz('America/Chicago').format('h:mm A') : dayjs(aptDate).tz('America/Chicago').format('MMMM D, h:mm A')}
                <br />
                {'GMT' + dayjs(aptDate).tz('America/Chicago').format('Z (z)')}
            </div>
        </div>
    );
};

export default TimeDisplay;

```

```
// Dynamic calendar

import React from 'react';
import Calendar from 'react-calendar';
import './Calendar.css'
// import 'react-calendar/dist/Calendar.css'; <- Old CSS file
import dayjs from 'dayjs';

const DateCalendar = ({
    errors,
    validated,
    available,
    booked,
    regionBooked,
    success,
    selectedDate,
    setSelectedDate,
}) => {
    return (
        <div className='calendarAlign disableCalendar'>
            <div className={validated && !!errors.date ? '
invalidCalendar' : ''}> {/*Control validation formatting*/}
                <Calendar
                    onChange={setSelectedDate}
                    value={selectedDate}
                    maxDate={new Date(dayjs().add(11, 'month').endOf('
month').toDate())}
                    minDate={new Date()}
                    next2Label=''
                    prev2Label=''
                    minDetail='year'
                    showNeighboringMonth='false'
                    tileClassName={({ date, view }) => { {/*
Conditional formatting*/}
                        if (dayjs(date).format('DD/MM/YYYY') === dayjs
(selectedDate).format('DD/MM/YYYY')) {
                            return 'selected';

                        } else if (available.includes(dayjs(date).
format('DD/MM/YYYY'))) {
                            return 'available';

                        } else if (booked.includes(dayjs(date).format(
'DD/MM/YYYY'))) {
                            return 'booked';

                        } else if (regionBooked.includes(dayjs(date).
format('DD/MM/YYYY'))) {
                            return 'regionBooked';

                        } else if (date > new Date()) {
                            return 'unavailable';

                        } else {
                            return 'past';
                        }
                    }}
```

```
51                      tileDisabled={({ date, view }) => { {/*
   Conditional disabling of dates*/}
52                          if (success && dayjs(date).format('DD/MM/YYYY
   ') !== dayjs(selectedDate).format('DD/MM/YYYY')) {
53                              return true;
54
55                          } else {
56                              if (available.includes(dayjs(date).format
   ('DD/MM/YYYY'))) {
57                                  return false;
58
59                              } else {
60                                  return true;
61                              }
62                          }
63                      }}
64                  />
65              </div>
66              <span className='invalidText'>{errors.date}</span>
67          </div>
68      );
69 };
70
71 export default DateCalendar;
72
```

```
// Validation code

import dayjs from 'dayjs';

export default function validateInfo(info) {
    let errors = {};
    const regex =
    /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[
0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[
a-zA-Z]{2,}))$/;

    if (dayjs(info.aptDate).format('DD/MM/YYYY') === dayjs(new Date
()).format('DD/MM/YYYY')) {
        errors.date = 'Date required';
    }
    if (info.booking.uniName === '') {
        errors.uniName = 'University name required';
    }
    if (info.booking.uniRepName === '') {
        errors.uniRepName = 'Name required';
    }
    if (info.booking.uniRepJobTitle === '') {
        errors.uniRepJobTitle = 'Job title required';
    }
    if (info.booking.uniRepEmail === '') {
        errors.uniRepEmail = 'Email required';

    } else if (!regex.test(info.booking.uniRepEmail)) {
        errors.uniRepEmail = 'Invalid email';
    }
    if (info.booking.uniRegion === '') {
        errors.uniRegion = 'Region required';
    }

    return errors;
}

```

```
.cardUni{
    font-size: 1.5rem;
    font-weight: 300;
}

.cardRegion{
    font-size: 1rem;
    font-weight: 500;
    color: #6c757d;
}

.cardImage{
    width: calc(100% + 2px);
    border-radius: 0;
    margin-left: -1px;
    border-left: 1px solid #DFDFDF;
    border-right: 1px solid #DFDFDF;
}

.cardDate{

}

.cardTitle{

}

.cardEmail{

}

.pad{
    margin-top: 1rem;
    margin-bottom: 1rem;
}

.cardButton{
    width: 100%;
}

.login-container{
    display: flex;
    justify-content: center;
    align-items: center;
}

.ccontainer {
    margin-left: calc(50vw - 20rem);
    margin-top: calc(50vh - 15rem);
}

.cbooked {
    color: hsla(3, 100%, 20%, 1) !important;
    background-color: rgba(255, 65, 54, 0.6) !important;
}

.cbooked:hover {
```

```
    color: hsla(3, 100%, 20%, 1) !important;
    background-color: rgba(255, 65, 54, 0.55) !important;
}

.cavailable {
    color: hsla(127, 63%, 15%, 1) !important;
    background-color: rgba(1, 255, 112, 0.7) !important;
}

.cavailable:hover {
    color: hsla(127, 63%, 15%, 1) !important;
    background-color: rgba(1, 255, 112, 0.5) !important;
}
```

```
// Same algorithm as used in AdminPage/bookings

import {useState} from "react";
import Card from 'react-bootstrap/Card'
import '../admin.css'
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';
import Form from 'react-bootstrap/Form'
import axios from 'axios';
import validateCouns from "./validateCouns";

const CCard = ({instance,setEmails}) => {
    const [show, setShow] = useState(false);
    const [validated, setValidated] = useState(false);
    const [error, setError] = useState({});

    const [counsInfo, setCounsInfo] = useState({
        name:instance.name,
        counsEmail:instance.counsEmail,
        receiveEmail:instance.receiveEmail,
    })

    const [tempCounsInfo, setTempCounsInfo] = useState({
        name:instance.name,
        counsEmail:instance.counsEmail,
        receiveEmail:instance.receiveEmail,
    })

    function cancel(){
        setCounsInfo(tempCounsInfo)
        setShow(false)
    }

    function handleOpen(){
        setTempCounsInfo(counsInfo)
        setShow(true);
        setValidated(false);
    }

    async function getEmail(){
        let res = await  axios.get('http://localhost:5000/couns/read')
        return res
    }

    async function edit(){
        if(!(Object.keys(validateCouns(counsInfo)).length === 0)){
            setError(validateCouns(counsInfo));
            setValidated(true);
        } else{
            const token = localStorage.getItem('adminToken')
            let verify;

            if(!!token) {
                verify = await axios.post('http://localhost:5000/login
  /verify', {adminToken:token});
            } else {
                verify =  false;
```

```
 57              }
 58
 59              if(verify){
 60                  setValidated(false);
 61
 62                  counsInfo.id = instance._id
 63                  counsInfo.adminToken = token
 64
 65                  axios.patch('http://localhost:5000/couns/edit',
    counsInfo);
 66                  setShow(false);
 67
 68              } else{
 69                  alert('Invalid authentication token')
 70              }
 71
 72          }
 73      }
 74
 75      async function remove(){
 76          const token = localStorage.getItem('adminToken')
 77          let verify;
 78
 79          if(!!token) {
 80              verify = await axios.post('http://localhost:5000/login/
    verify', {adminToken:token});
 81          } else {
 82              verify =  false;
 83          }
 84
 85          if(verify){
 86              axios.delete('http://localhost:5000/couns/remove', {data
    : {id:instance._id, adminToken:token}})
 87              .then(()=>{
 88                  // eslint-disable-next-line
 89                  getEmail()
 90                  .then((res)=>{
 91                      setEmails(res.data);
 92                  })
 93              }).then(setShow(false));
 94          } else{
 95              alert('Invalid authentication token')
 96          }
 97
 98      }
 99
100      return (
101          <>
102              <Modal show={show} onHide={cancel} backdrop="static">
103                  <Modal.Header closeButton>
104                      <Modal.Title>{tempCounsInfo.name}</Modal.Title>
105                  </Modal.Header>
106                  <Modal.Body>
107                      <Form>
108                          <Form.Group>
109                              <Form.Label> ID </Form.Label>
110                              <Form.Control
```

```
111                                 type='text'
112                                 value={instance._id}
113                                 readOnly={true}
114                             />
115                         </Form.Group>
116                         <Form.Group>
117                             <Form.Label> Counsellor Name </Form.Label
    >
118                             <Form.Control
119                                 type='text'
120                                 defaultValue={tempCounsInfo.name}
121                                 onChange={(e) => {
122                                     setCounsInfo({...counsInfo,name:
    e.target.value,})
123                                 }}
124                                 isInvalid={validated?!!error.name:
    false}
125                                 isValid={validated?!error.name:false}
126                             />
127                         </Form.Group>
128                         <Form.Group>
129                             <Form.Label> Email </Form.Label>
130                             <Form.Control
131                                 type='text'
132                                 defaultValue={tempCounsInfo.
    counsEmail}
133                                 onChange={(e) => {
134                                     setCounsInfo({...counsInfo,
    counsEmail: e.target.value,})
135                                 }}
136                                 isInvalid={validated?!!error.
    counsEmail:false}
137                                 isValid={validated?!error.counsEmail:
    false}
138                             />
139                         </Form.Group>
140                         <Form.Group>
141                             <Form.Label> Receive Emails? </Form.Label
    >
142                             <Form.Control
143                                 type='checkbox'
144                                 defaultChecked={tempCounsInfo.
    receiveEmail}
145                                 onChange={(e)=> {
146                                     setCounsInfo({...counsInfo,
    receiveEmail: e.target.value,})
147                                 }}
148                                 isInvalid={validated?!!error.
    receiveEmail:false}
149                                 isValid={validated?!error.
    receiveEmail:false}
150                             />
151                         </Form.Group>
152                     </Form>
153                 </Modal.Body>
154                 <Modal.Footer>
155                     <Button variant="danger" onClick={()=>{remove
```

```
155 ()}} style={{'margin-right':'auto'}}>
156                         Delete Booking
157                     </Button>
158                     <Button variant="primary" onClick={()=>{edit
    ()}} >
159                         Save Changes
160                     </Button>
161                 </Modal.Footer>
162             </Modal>
163 
164             <div className='col-6 pad'>
165                 <Card style={{ width: '100%' }}>
166                     <Card.Body>
167                         <Card.Title className='cardUni'>{counsInfo.
    name}</Card.Title>
168                         <Card.Text className='cardEmail'>{counsInfo.
    counsEmail}</Card.Text>
169                         <Card.Text className='cardTitle'>{'Receive
    Emails?: '+counsInfo.receiveEmail}</Card.Text>
170                         <Button variant="primary" className="
    cardButton" onClick={() => {handleOpen()}}>Edit</Button>
171                     </Card.Body>
172                 </Card>
173             </div>
174         </>
175     )
176 }
177 
178 export default CCard
179 
```

```
// Same algorithm as used in AdminPage/bookings

import {useState} from "react";
import Card from 'react-bootstrap/Card'
import '../admin.css'
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';
import Form from 'react-bootstrap/Form'
import axios from 'axios';
import validateCouns from "./validateCouns";

const CCardAdd = ({setEmails}) => {
    const [show, setShow] = useState(false);
    const [validated, setValidated] = useState(false);
    const [error, setError] = useState({});

    const [counsInfo, setCounsInfo] = useState({
        name: "Name",
        counsEmail: "Email",
        receiveEmail:true,
    })

    const [tempCounsInfo, setTempCounsInfo] = useState({
        name: "Name",
        counsEmail: "Email",
        receiveEmail:true,
    })

    function cancel(){
        setCounsInfo(tempCounsInfo)
        setShow(false)
    }

    function handleOpen(){
        setTempCounsInfo(counsInfo)
        setShow(true);
        setValidated(false);
    }

    async function getEmail(){
        let res = await  axios.get('http://localhost:5000/couns/read')
        return res
    }

    async function create(){
        if(!(Object.keys(validateCouns(counsInfo)).length === 0)){
            setError(validateCouns(counsInfo));
            setValidated(true);
        } else{
            const token = localStorage.getItem('adminToken')
            let verify;

            if(!!token) {
                verify = await axios.post('http://localhost:5000/login
  /verify', {adminToken:token});
            } else {
                verify =  false;
```

```
            }

            if(verify){
                setValidated(false);

                counsInfo.adminToken = token

                axios.post('http://localhost:5000/couns/create', counsInfo)
                .then(()=>{
                    getEmail()
                    .then((res)=>{
                        setEmails(res.data)
                    })
                }).catch((err)=>{
                    console.log(err)
                    alert(err)
                })
                setShow(false);
            } else{
                alert('Invalid authentication token')
            }
        }
    }

    return (
        <>
          <Modal show={show} onHide={cancel} backdrop="static">
                <Modal.Header closeButton>
                    <Modal.Title>Name</Modal.Title>
                </Modal.Header>
                <Modal.Body>
                    <Form>
                        <Form.Group>
                            <Form.Label> Counsellor Name </Form.Label>
                            <Form.Control
                                type='text'
                                onChange={(e) => {
                                    setCounsInfo({...counsInfo,name: e.target.value,})
                                }}
                                isInvalid={validated?!!error.name: false}
                                isValid={validated?!error.name:false}
                            />
                        </Form.Group>
                        <Form.Group>
                            <Form.Label> Email </Form.Label>
                            <Form.Control
                                type='text'
                                onChange={(e) => {
                                    setCounsInfo({...counsInfo, counsEmail: e.target.value,})
                                }}
                                isInvalid={validated?!!error.counsEmail:false}
```

```
                                    isValid={validated?!error.counsEmail:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Receive Emails? </Form.Label>
                                <Form.Control
                                    type='checkbox'
                                    defaultChecked={true}
                                    onClick={()=>{setCounsInfo({...counsInfo,receiveEmail:!counsInfo.receiveEmail})}}
                                    isInvalid={validated?!!error.receiveEmail:false}
                                    isValid={validated?!error.receiveEmail:false}
                                />
                            </Form.Group>
                        </Form>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button variant="success" onClick={()=>{create()}}>
                            Add Counsellor
                        </Button>
                    </Modal.Footer>
                </Modal>
                <div className='col-6 pad'>
                    <Card style={{ width: '100%' }}>
                        <Card.Body>
                            <Card.Title className='cardUni'>Name</Card.Title>
                            <Card.Text className='cardTitle'>Email</Card.Text>
                            <Card.Text className='cardEmail'>Receive Emails?: true/false</Card.Text>
                            <Button variant="success" className="cardButton" onClick={() => {handleOpen()}}>Add</Button>
                        </Card.Body>
                    </Card>
                </div>
            </>
        )
}

export default CCardAdd
```

```
// Same algorithm as used in AdminPage/bookings

import { useEffect, useState } from "react"
import axios from 'axios';
import CCard from "./CCard";
import CCardAdd from "./CCardAdd";

const EditEmail = () => {
    const [emails, setEmails] = useState([])

    useEffect(async () => {
        async function getEmail(){
            let res = await  axios.get('http://localhost:5000/couns/read')
            return res
        }

        getEmail().then((res)=>setEmails(res.data))
    }, [])

    return (
        <div className='container'>
            <div className='row pad'>
                {emails.map(instance=>{
                    return(
                      <CCard instance={instance} setEmails={setEmails}/>
                    )
                })}
                <CCardAdd setEmails={setEmails}/>
            </div>
        </div>
    )
}

export default EditEmail

```

```
// Same algorithm as used in AdminPage/bookings

export default function validateCouns(counsInfo) {
    let errors = {};
    const regex =
    /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[
0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[
a-zA-Z]{2,}))$/;

    if (counsInfo.name === '' || counsInfo.name === 'Name') {
        errors.name = 'University name required';
    }

    if (counsInfo.counsEmail === '' || counsInfo.counsEmail === 'Email
') {
        errors.counsEmail = 'Email required';

    } else if (!regex.test(counsInfo.counsEmail)) {
        errors.counsEmail = 'Invalid email';
    }

    return errors
}
```

```
// Booking card template used to display upcoming bookings
// Also features the modal used to edit bookings

import {useState} from "react";
import Card from 'react-bootstrap/Card'
import image from './undraw_Page_not_found_re_e9o6.png'
import dayjs from "dayjs";
import '../admin.css'
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';
import Form from 'react-bootstrap/Form'
import axios from 'axios';
import validateInfo from "./validateCreate";

const BCard = ({instance,setBooking}) => {
    const [show, setShow] = useState(false);
    const [validated, setValidated] = useState(false);
    const [error, setError] = useState({date:''});

    const [date, setDate] = useState(instance.aptDate);
    const [tempDate, setTempDate] = useState(instance.aptDate);

    const [sendEmail, setSendEmail] = useState(false)

    const [bookingData, setBookingData] = useState({
        uniName: instance.booking.uniName,
        uniRegion: instance.booking.uniRegion,
        uniRepJobTitle: instance.booking.uniRepJobTitle,
        uniRepName: instance.booking.uniRepName,
        uniRepEmail: instance.booking.uniRepEmail,
        logoUrl:instance.booking.logoUrl,
    });
    const [tempBooking, setTempBooking] = useState({
        uniName: instance.booking.uniName,
        uniRegion: instance.booking.uniRegion,
        uniRepJobTitle: instance.booking.uniRepJobTitle,
        uniRepName: instance.booking.uniRepName,
        uniRepEmail: instance.booking.uniRepEmail,
        logoUrl:instance.booking.logoUrl,
    });

    function cancel(){
        setBookingData(tempBooking)
        setDate(tempDate)
        setShow(false);
    }

    function handleOpen(){
        setTempBooking(bookingData);
        setTempDate(date);
        setValidated(false);
        setShow(true);
    }

    async function getDisplayData() { // Async to prevent stalling
        let res = await axios.get('http://localhost:5000/booking/
readfordisplay');
```

```
 57         return res;
 58     }
 59 
 60     async function edit(){
 61         if(!(Object.keys(validateInfo(bookingData,date)).length === 0
    )){
 62             setError(validateInfo(bookingData,date));
 63             setValidated(true);
 64         } else{
 65             const token = localStorage.getItem('adminToken')
 66             let verify;
 67 
 68             if(!!token) {
 69                 verify = await axios.post('http://localhost:5000/
    login/verify', {adminToken:token});
 70             } else {
 71                 verify = false;
 72             }
 73 
 74             if(verify){
 75                 setValidated(false);
 76                 let bookingDateInfoInstance = {
 77                     id:instance._id,
 78                     aptDate: date,
 79                     status: 'Booked',
 80                     booking: {
 81                         uniName: bookingData.uniName,
 82                         uniRepName: bookingData.uniRepName,
 83                         uniRepJobTitle: bookingData.uniRepJobTitle,
 84                         uniRepEmail: bookingData.uniRepEmail,
 85                         uniRegion: bookingData.uniRegion,
 86                         logoUrl: bookingData.logoUrl,
 87                     },
 88                     allowMail: sendEmail,
 89                     adminToken: token,
 90                 };
 91 
 92                 axios.patch('http://localhost:5000/booking/
    editbooking', bookingDateInfoInstance);
 93                 setShow(false);
 94             } else{
 95                 alert('Invalid authentication token')
 96             }
 97         }
 98     }
 99 
100 
101     async function remove(){
102         const token = localStorage.getItem('adminToken')
103         let verify;
104 
105         if(!!token) {
106             verify = await axios.post('http://localhost:5000/login/
    verify', {adminToken:token});
107         } else {
108             verify = false;
109         }
```

```
110
111         if(verify){
112             axios.delete('http://localhost:5000/booking/remove', {
    data: {id:instance._id,adminToken:token}})
113             .then(()=>{
114                 // eslint-disable-next-line
115                 getDisplayData()
116                 .then((res)=>{
117                     setBooking(res.data);
118                 })
119             }).then(setShow(false));
120         } else{
121             alert('Invalid authentication token')
122         }
123     }
124
125     return (
126         <>
127             <Modal show={show} onHide={cancel} backdrop="static">
128                 <Modal.Header closeButton>
129                     <Modal.Title>{tempBooking.uniName}</Modal.Title>
130                 </Modal.Header>
131                 <Modal.Body>
132                     <Form >
133                         <Form.Group>
134                             <Form.Label> ID </Form.Label>
135                             <Form.Control
136                                 type='text'
137                                 value={instance._id}
138                                 readOnly={true}
139                             />
140                         </Form.Group>
141                         <Form.Group>
142                             <Form.Label> University Name </Form.Label
    >
143                             <Form.Control
144                                 type='text'
145                                 defaultValue={tempBooking.uniName}
146                                 onChange={(e) => {
147                                     setBookingData({...bookingData,
    uniName: e.target.value,})
148                                 }}
149                                 isInvalid={validated?!!error.uniName:
    false}
150                                 isValid={validated?!error.uniName:
    false}
151                             />
152                         </Form.Group>
153                         <Form.Group>
154                             <Form.Label> University Region </Form.
    Label>
155                             <Form.Control
156                                 type='text'
157                                 defaultValue={tempBooking.uniRegion}
158                                 onChange={(e) => {
159                                     setBookingData({...bookingData,
    uniRegion: e.target.value,})
```

```
160                                 }}
161                                 isInvalid={validated?!!error.
    uniRegion:false}
162                                 isValid={validated?!error.uniRegion:
    false}
163                             />
164                         </Form.Group>
165                         <Form.Group>
166                             <Form.Label> Date </Form.Label>
167                             <Form.Control
168                                 type='text'
169                                 defaultValue={dayjs(tempDate).format(
    'M/D/YYYY')}
170                                 onChange={(e)=> {
171                                     setDate(new Date(dayjs(e.target.
    value)))
172 
173                                 }}
174                                 isInvalid={validated?!!error.date:
    false}
175                                 isValid={validated?!error.date:false}
176                             />
177                         </Form.Group>
178                         <Form.Group>
179                             <Form.Label> Rep Title </Form.Label>
180                             <Form.Control
181                                 type='text'
182                                 defaultValue={tempBooking.
    uniRepJobTitle}
183                                 onChange={(e) => {
184                                     setBookingData({...bookingData,
    uniRepJobTitle: e.target.value,})
185                                 }}
186                                 isInvalid={validated?!!error.
    uniRepJobTitle:false}
187                                 isValid={validated?!error.
    uniRepJobTitle:false}
188                             />
189                         </Form.Group>
190                         <Form.Group>
191                             <Form.Label> Rep Name </Form.Label>
192                             <Form.Control
193                                 type='text'
194                                 defaultValue={tempBooking.uniRepName}
195                                 onChange={(e) => {
196                                     setBookingData({...bookingData,
    uniRepName: e.target.value,})
197                                 }}
198                                 isInvalid={validated?!!error.
    uniRepName:false}
199                                 isValid={validated?!error.uniRepName:
    false}
200                             />
201                         </Form.Group>
202                         <Form.Group>
203                             <Form.Label> Rep Email </Form.Label>
204                             <Form.Control
```

```
                                    type='text'
                                    defaultValue={tempBooking.uniRepEmail
}
                                    onChange={(e) => {
                                        setBookingData({...bookingData,
uniRepEmail: e.target.value,})
                                    }}
                                    isInvalid={validated?!!error.
uniRepEmail:false}
                                    isValid={validated?!error.uniRepEmail
:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Image URL </Form.Label>
                                <Form.Control
                                    type='text'
                                    defaultValue={tempBooking.logoUrl}
                                    onChange={(e) => {
                                        setBookingData({...bookingData,
logoUrl: e.target.value,})
                                    }}
                                    isValid={validated?true:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Send Email? </Form.Label>
                                <Form.Control
                                    type='checkbox'
                                    defaultChecked={true}
                                    onChange={(e) => {
                                        setSendEmail(e.target.value)
                                    }}
                                    isValid={validated?true:false}
                                />
                            </Form.Group>
                        </Form>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button variant="danger" onClick={()=>{remove
()}} style={{'margin-right':'auto'}}>
                            Delete Booking
                        </Button>
                        <Button variant="primary" onClick={()=>{edit
()}} >
                            Save Changes
                        </Button>
                    </Modal.Footer>
                </Modal>

                <div className='col-4 pad'>
                    <Card style={{ width: '100%' }}>
                        <Card.Body>
                            <Card.Title className='cardUni'>{bookingData.
uniName}</Card.Title>
                            <Card.Subtitle className='cardRegion'>{
bookingData.uniRegion}</Card.Subtitle>
```

```
253                     </Card.Body>
254                     <Card.Img variant="top" src={!!bookingData.
    logoUrl?bookingData.logoUrl:image} className='cardImage'/>
255                     <Card.Body>
256                         <Card.Text className='cardDate'>{dayjs(date).
    format('dddd, MMMM D')}</Card.Text>
257                         <Card.Text className='cardTitle'>{bookingData
    .uniRepJobTitle + ': ' + bookingData.uniRepName}</Card.Text>
258                         <Card.Text className='cardEmail'>{bookingData
    .uniRepEmail}</Card.Text>
259                         <Button variant="primary" className="
    cardButton" onClick={() => {handleOpen()}}>Edit</Button>
260                     </Card.Body>
261                 </Card>
262             </div>
263         </>
264     )
265 }
266
267 export default BCard
268
269
```

```
// Add booking card template used to display upcoming bookings
// Uses a modal that displays a form similiar to the form page

import {useState} from "react";
import Card from 'react-bootstrap/Card'
import image from './undraw_No_data_re_kwbl.png'
import dayjs from "dayjs";
import '../admin.css'
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';
import Form from 'react-bootstrap/Form'
import axios from 'axios';
import validateInfo from "./validateCreate";

const BCardAdd = ({setBooking}) => {
    const [show, setShow] = useState(false);
    const [validated, setValidated] = useState(false);
    const [error, setError] = useState(false)

    const [date, setDate] = useState("Month/Day/Year"); // Store appointment date
    const [tempDate, setTempDate] = useState("Month/Day/Year"); // Store for cancel

    const [bookingData, setBookingData] = useState({ // Store booking information
        uniName: "University Name",
        uniRegion: "University Region",
        uniRepJobTitle: "Rep Title",
        uniRepName: "Rep Name",
        uniRepEmail: "Rep Email",
        logoUrl: "",
    });
    const [tempBooking, setTempBooking] = useState({ // Store for cancel
        uniName: "University Name",
        uniRegion: "University Region",
        uniRepJobTitle: "Rep Title",
        uniRepName: "Rep Name",
        uniRepEmail: "Rep Email",
        logoUrl: "",
    });

    function cancel(){
        setBookingData(tempBooking)
        setDate(tempDate)
        setShow(false);
    }

    function handleOpen(){
        setTempBooking(bookingData);
        setTempDate(date);
        setValidated(false);
        setShow(true);
    }

    async function getDisplayData() { // Async to prevent stalling
```

```
54          let response = await axios.get('http://localhost:5000/booking
    /readfordisplay');
55          return response;
56      }
57
58      async function create(){
59          if(!(Object.keys(validateInfo(bookingData,date)).length === 0
    )){
60              setError(validateInfo(bookingData,date));
61              setValidated(true);
62          } else{
63              const token = localStorage.getItem('adminToken')
64              let verify;
65
66              if(!!token) {
67                  verify = await axios.post('http://localhost:5000/
    login/verify', {adminToken:token});
68              } else {
69                  verify =  false;
70              }
71
72              if(verify){
73                  setValidated(false);
74
75                  let bookingDateInfoInstance = {
76                      aptDate: date,
77                      status: 'Booked',
78                      booking: {
79                          uniName: bookingData.uniName,
80                          uniRepName: bookingData.uniRepName,
81                          uniRepJobTitle: bookingData.uniRepJobTitle,
82                          uniRepEmail: bookingData.uniRepEmail,
83                          uniRegion: bookingData.uniRegion,
84                          logoUrl: bookingData.logoUrl,
85                      },
86                  };
87
88                  axios.post('http://localhost:5000/booking/create',
    bookingDateInfoInstance)
89                  .then(()=>{
90                      // eslint-disable-next-line
91                      getDisplayData()
92                      .then((res)=>{
93                          setBooking(res.data);
94                      })
95                  }).then(setShow(false));
96              } else{
97                  alert('Invalid authentication token')
98              }
99          }
100     }
101
102     return (
103         <>
104             <Modal show={show} onHide={cancel} backdrop="static">
105                 <Modal.Header closeButton>
106                     <Modal.Title>University Name</Modal.Title>
```

```
107                 </Modal.Header>
108                 <Modal.Body>
109                     <Form>
110                         <Form.Group>
111                             <Form.Label> University Name </Form.Label
    >
112                             <Form.Control
113                                 type='text'
114                                 onChange={(e) => {
115                                     setBookingData({...bookingData,
    uniName: e.target.value,})
116                                 }}
117                                 isInvalid={validated?!!error.uniName:
    false}
118                                 isValid={validated?!error.uniName:
    false}
119                             />
120                         </Form.Group>
121                         <Form.Group>
122                             <Form.Label> University Region </Form.
    Label>
123                             <Form.Control
124                                 type='text'
125                                 onChange={(e) => {
126                                     setBookingData({...bookingData,
    uniRegion: e.target.value,})
127                                 }}
128                                 isInvalid={validated?!!error.
    uniRegion:false}
129                                 isValid={validated?!error.uniRegion:
    false}
130                             />
131                         </Form.Group>
132                         <Form.Group>
133                             <Form.Label> Date </Form.Label>
134                             <Form.Control
135                                 type='text'
136                                 onChange={(e)=> {
137                                     setDate(new Date(dayjs(e.target.
    value)))
138
139                                 }}
140                                 isInvalid={validated?!!error.date:
    false}
141                                 isValid={validated?!error.date:false}
142                             />
143                         </Form.Group>
144                         <Form.Group>
145                             <Form.Label> Rep Title </Form.Label>
146                             <Form.Control
147                                 type='text'
148                                 onChange={(e) => {
149                                     setBookingData({...bookingData,
    uniRepJobTitle: e.target.value,})
150                                 }}
151                                 isInvalid={validated?!!error.
    uniRepJobTitle:false}
```

```
                                    isValid={validated?!error.
    uniRepJobTitle:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Rep Name </Form.Label>
                                <Form.Control
                                    type='text'
                                    onChange={(e) => {
                                        setBookingData({...bookingData,
    uniRepName: e.target.value,})
                                    }}
                                    isInvalid={validated?!!error.
    uniRepName:false}
                                    isValid={validated?!error.uniRepName:
    false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Rep Email </Form.Label>
                                <Form.Control
                                    type='text'
                                    onChange={(e) => {
                                        setBookingData({...bookingData,
    uniRepEmail: e.target.value,})
                                    }}
                                    isInvalid={validated?!!error.
    uniRepEmail:false}
                                    isValid={validated?!error.uniRepEmail
    :false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Image URL </Form.Label>
                                <Form.Control
                                    type='text'
                                    onChange={(e) => {
                                        setBookingData({...bookingData,
    logoUrl: e.target.value,})
                                    }}
                                    isValid={validated?true:false}
                                />
                            </Form.Group>
                        </Form>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button variant="success" onClick={()=>{create
    ()}} >
                            Create Booking
                        </Button>
                    </Modal.Footer>
                </Modal>
                <div className='col-4 pad'>
                    <Card style={{ width: '100%' }}>
                        <Card.Body>
                            <Card.Title className='cardUni'>University
    Name</Card.Title>
```

```
                        <Card.Subtitle className='cardRegion'>University Region</Card.Subtitle>
                    </Card.Body>
                    <Card.Img variant="top" src={image} className='cardImage'/>
                    <Card.Body>
                        <Card.Text className='cardDate'>Date</Card.Text>
                        <Card.Text className='cardTitle'>Rep Title: Rep Name</Card.Text>
                        <Card.Text className='cardEmail'>Rep Email</Card.Text>
                        <Button variant="success" className="cardButton" onClick={() => {handleOpen()}}>Add</Button>
                    </Card.Body>
                </Card>
            </div>
        </>
    )
}

export default BCardAdd


```

```
// Iterate through bookings and display booking cards and the add booking card

import {useEffect, useState} from "react";
import axios from "axios";
import '../admin.css'
import BCard from "./BCard";
import BCardAdd from "./BCardAdd";

const EditBookings = () => {
    const [booking, setBooking] = useState([])

    // eslint-disable-next-line
    useEffect(async () => {
        async function getDisplayData() { // Async to prevent stalling

            let res = await axios.get('http://localhost:5000/booking/readfordisplay');
            return res;
        }

        getDisplayData().then((res)=>setBooking(res.data))
    }, [])

    return (
        <>
        <div className='container'>
            <div className='row pad'>
                {booking.map(instance=>{
                    return(
                      <BCard instance={instance} setBooking={setBooking}/>
                    )
                })}
                <BCardAdd setBooking={setBooking}/>
            </div>
        </div>
        </>
    );
};

export default EditBookings;

```

```
// Validation

export default function validateInfo(bookingData,date) {
    console.log(bookingData)
    let errors = {};
    const regex =
    /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[
0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[
a-zA-Z]{2,}))$/;

    if (isNaN(Date.parse(date))) {
        errors.date = 'Invalid Date';
    }
    if (bookingData.uniName === '' || bookingData.uniName === '
University Name') {
        errors.uniName = 'University name required';
    }
    if (bookingData.uniRepName === '' || bookingData.uniRepName === '
Rep Name') {
        errors.uniRepName = 'Name required';
    }
    if (bookingData.uniRepJobTitle === '' || bookingData.
uniRepJobTitle === 'Rep Title') {
        errors.uniRepJobTitle = 'Job title required';
    }
    if (bookingData.uniRepEmail === '' || bookingData.uniRepEmail ===
'Rep Email') {
        errors.uniRepEmail = 'Email required';

    } else if (!regex.test(bookingData.uniRepEmail)) {
        errors.uniRepEmail = 'Invalid email';
    }
    if (bookingData.uniRegion === '' || bookingData.uniRegion === '
University Region') {
        errors.uniRegion = 'Region required';
    }

    console.log(errors)
    return errors;
}

```

```css
abbr[title] {
  border-bottom: none !important;
  cursor: inherit !important;
  text-decoration: none !important;
}
.react-calendar {
  width: 40rem;
  max-width: 100%;
  background: white;
  /* border: 1px solid #a0a096; */
  font-family: "system-ui";
  line-height: 1.125em;
}
.react-calendar--doubleView {
  width: 700px;
}
.react-calendar--doubleView .react-calendar__viewContainer {
  display: flex;
  margin: -0.5em;
}
.react-calendar--doubleView .react-calendar__viewContainer > * {
  width: 50%;
  margin: 0.5em;
}
.react-calendar,
.react-calendar *,
.react-calendar *:before,
.react-calendar *:after {
  -moz-box-sizing: border-box;
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}
.react-calendar button {
  margin: 0;
  border: 0;
  outline: none;
}
.react-calendar button:enabled:hover {
  cursor: pointer;
}
.react-calendar__navigation {
  height: 44px;
  margin-bottom: 1em;
}
.react-calendar__navigation button {
  min-width: 44px;
  background: none;
}
.react-calendar__navigation button:enabled:hover,
.react-calendar__navigation button:enabled:focus {
  background-color: #e6e6e6;
}
/* .react-calendar__navigation button[disabled] {
  /* background-color: #f0f0f0; */
/*} */
.react-calendar__month-view__weekdays {
  text-align: center;
```

```
  text-transform: uppercase;
  font-weight: bold;
  font-size: 0.75em;

}
.react-calendar__month-view__weekdays__weekday {
  padding: 0.5em;
}
.react-calendar__month-view__weekNumbers {
  font-weight: bold;
}
.react-calendar__month-view__weekNumbers .react-calendar__tile {
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 0.75em;
  padding: calc(0.75em / 0.75) calc(0.5em / 0.75);
}
.react-calendar__month-view__days__day--weekend {
  color: #d10000;
}
/* .react-calendar__month-view__days__day--neighboringMonth {
  color: #757575;
} */
.react-calendar__year-view .react-calendar__tile,
.react-calendar__decade-view .react-calendar__tile,
.react-calendar__century-view .react-calendar__tile {
  padding: 2em 0.5em;
}
.react-calendar__tile {
  max-width: 100%;
  text-align: center;
  padding: 0.75em 0.5em;
  background: none;
}
.react-calendar__tile:disabled {
  background-color: #f0f0f0;
}
.react-calendar__tile:enabled:hover,
.react-calendar__tile:enabled:focus {
  background-color: #fff;
}
.react-calendar__tile--now {
  background: #ffff76;
}
.react-calendar__tile--now:enabled:hover,
.react-calendar__tile--now:enabled:focus {
  background: #ffffa9;
}
.react-calendar__tile--hasActive {
  background: #76baff;
}
.react-calendar__tile--hasActive:enabled:hover,
.react-calendar__tile--hasActive:enabled:focus {
  background: #a9d4ff;
}
.react-calendar__tile--active {
```

```
115   background: #006edc;
116   color: white;
117 }
118 .react-calendar__tile--active:enabled:hover,
119 .react-calendar__tile--active:enabled:focus {
120   background: #1087ff;
121 }
122 .react-calendar--selectRange .react-calendar__tile--hover {
123   background-color: #e6e6e6;
124 }
125
```

```
// Calendar to add or remove available days

import Calendar from 'react-calendar';
import Button from 'react-bootstrap/Button'
import './Calendar.css'
import { useState } from 'react';
import { useEffect } from 'react';
import dayjs from 'dayjs';
import axios from 'axios'

const EditCalendar = () => {
    const [dates, setDates] = useState({available:[],booked:[]})
    const [tempDates, setTempDates] = useState({})
    const [changeLog, setChangeLog] = useState({})

    async function resetCalendar(){
        async function getRegionData() { // Async to prevent stalling
            let response = await axios.get('http://localhost:5000/booking/readforform');
            return response;
        }

        // eslint-disable-next-line
        let calendarData = (await getRegionData()).data;

        let available = [];
        let booked = [];

        calendarData.forEach((instance) => { // Simple calendar formatting
            if (instance.status === 'Available') {
                available.push(dayjs(instance.aptDate).format('DD/MM/YYYY'));

            } else if (instance.status === 'Booked') {
                booked.push(dayjs(instance.aptDate).format('DD/MM/YYYY'));
            }
        });

        return {available:available,booked:booked}
    }


    useEffect(async ()=>{
        let res = await resetCalendar()
        setTempDates(res.available);
        setDates(res);
    }, [])

    async function handleSubmit(){
        const token = localStorage.getItem('adminToken')
        let verify;

        if(!!token) {
            verify = await axios.post('http://localhost:5000/login/verify', {adminToken:token});
```

```
 53         } else {
 54             verify =  false;
 55         }
 56 
 57         if(verify){
 58             let submitData = {
 59                 changeLog: changeLog,
 60                 adminToken: token,
 61             }
 62 
 63             axios.post('http://localhost:5000/booking/editcalendar',
    submitData)
 64             setChangeLog({})
 65         } else {
 66             alert('Invalid authentication token')
 67         }
 68     }
 69 
 70     return (
 71         <div className='ccontainer'>
 72             <Calendar
 73                 onChange={(e)=>{
 74                     let changeDate = dayjs(e).format('DD/MM/YYYY')
 75 
 76                     if(dates.available.includes(changeDate)){ //
    Remove
 77 
 78                         let Arr = dates.available // Get the
    available array from dates
 79                         let newArray = Arr.filter(day=>day!==
    changeDate) // Remove the selected date from newArray
 80 
 81                         setDates({...dates,available:newArray}) //
    Set dates to the new filtered array
 82 
 83                         // If the original array already had the
    selected date as unavailalbe then remove the changes from the
    changelog
 84                         // This prevents accidentally trying to push
    duplicate changes
 85                         if(tempDates.includes(changeDate)){
 86                             setChangeLog({...changeLog, [e]:false})
 87                         } else {
 88                             let newChanges = changeLog
 89                             delete newChanges[e]
 90                             setChangeLog(newChanges)
 91                         }
 92 
 93                     } else { // Add
 94                         setDates({...dates,available:[...dates.
    available,changeDate]}) // Add to dates
 95 
 96                         if(tempDates.includes(changeDate)){
 97                             let newChanges = changeLog
 98                             delete newChanges[e]
 99                             setChangeLog(newChanges)
100                         } else {
```

```
                                setChangeLog({...changeLog, [e]:true})
                            }
                        }
                }}

                tileClassName={({ date, view }) => {
                    if (dates.available.includes(dayjs(date).format('DD/MM/YYYY'))) {
                        return 'cavailable';

                    } else if (dates.booked.includes(dayjs(date).format('DD/MM/YYYY'))) {
                        return 'cbooked';
                }}}

                tileDisabled={({ date, view }) => {
                    if (date>new Date()) {
                        return false;
                    } else {
                        return true;
                    }
                }}

                value=''
                showNeighboringMonth='false'
            />
            <Button
            disabled={Object.keys(changeLog).length === 0}
            onClick={(()=>{handleSubmit()})}>Save changes</Button>
            <Button
            disabled={Object.keys(changeLog).length === 0}
            onClick={async ()=>{
                let res = await resetCalendar();
                setTempDates(res.available);
                setDates(res);
            }

            }>Cancel</Button>
        </div>
    )
}

export default EditCalendar

```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">

    </div>
  </body>
</html>

```

```
1 # https://www.robotstxt.org/robotstxt.html
2 User-agent: *
3 Disallow:
4
```

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
  }
```

```
PORT=5000
ATLAS_URI=mongodb+srv://userNirav:userNirav8821@cluster0.k3lyw.mongodb.net/counsellorWebsite?retryWrites=true&w=majority
MAIL_USERNAME=bishvirtualsignup@gmail.com
OAUTH_CLIENTID=549431409213-a77901fhj77v8bjmgt6df6jbikm9tak0.apps.googleusercontent.com
OAUTH_CLIENT_SECRET=byzmdnWwZnSyyXoONTFgaTIh
OAUTH_REFRESH_TOKEN=1//04g53NsW8lntqCgYIARAAGAQSNwF-L9IrGGn4g1X15SXPjjOHYp5g-nnaFChWVID01mPrO2tA344qmb2V6yBNHVeJ2K7YVmjHFpw
OAUTH_ACCESS_TOKEN=ya29.A0ARrdaM-WyL_ZdBFiZuzwzp9BLRCRGKAE9SImtKVgdQMJ10vjAphQJkdn_LvMaZ73XdVdIcQ7zWGsZKBz3fqMptmpXNGEv4oqnx8BIsdsAuCzWRJojcom9_-bnULd-D1XTO4j324S-1FLnopUXRXPEyFZENPr
JWT_SECRET=376*QWe@4i5&@7!dp3!&Jg7As7N9p$Bji2^^
ADMIN_PASSWORD=123A
IMAGE_API_KEY=5ef594258facaf6d70f45f2abb7f91f07ef1e90b03d3b98e1843e345106fc6c4
```

```
// Setting up server

import express from 'express';
import mongoose from 'mongoose';
import cors from 'cors';
import dotenv from 'dotenv';

dotenv.config();

const app = express();

// Some middleware
app.use(express.urlencoded({ extended: true }));
app.use(express.json({ extended: true }));
app.use(cors());

// Routes
import bookingRouter from './API/routes/booking.js';
import counsRouter from './API/routes/couns.js';
import loginRouter from './API/routes/login.js'

app.use('/booking', bookingRouter);
app.use('/couns', counsRouter);
app.use('/login', loginRouter)
app.use('*', (req, res) => res.status(404).json({ error: 'Page not found' }));

// Connect to database
const url = process.env.ATLAS_URI;
mongoose.connect(url, { useNewUrlParser: true, useUnifiedTopology: true, 'useCreateIndex': true});
const connection = mongoose.connection;
connection.once('open', () => {console.log('Connected to MongoDB');});

// Start listening
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port: ${PORT}`));

mongoose.set('useFindAndModify', false);

```

```
# Docker file for server

FROM node:14.17.4

WORKDIR /app

COPY package*.json ./

RUN npm i

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

```

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.24.0",
    "cors": "^2.8.5",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "googleapis": "^84.0.0",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^5.13.5",
    "nodemailer": "^6.6.3",
    "nodemon": "^2.0.12"
  }
}

```

```
1 node_modules
2
```

```
// schematic for a booking record

import mongoose from 'mongoose';
mongoose.pluralize(null); // Prevent database from pluralizing

const schema = mongoose.Schema({
    aptDate: { type: Date, required: true, unique: true },
    status: { type: String, required: true },
    booking: {
        uniName: String,
        uniRepName: String,
        uniRepJobTitle: String,
        uniRepEmail: String,
        uniRegion: String,
        logoUrl: String,
    },
});

const bookingDateInfo = mongoose.model('bookingDateInfo', schema);

export default bookingDateInfo;

```

```
// schematic for a counselor record

import mongoose from 'mongoose';
mongoose.pluralize(null); // Prevent database from pluralizing

const schema = mongoose.Schema({
    name: String,
    counsEmail: {type:String, required:true, unique: true},
    receiveEmail: {type:Boolean ,required:true}
});

const counsellorEmail = mongoose.model('counsellorEmail', schema);

export default counsellorEmail;

```

```
// Counselor route, featuring all endpoints

import express from 'express';
import counsellorEmail from '../models/counsellorEmail.model.js';
import jwt from 'jsonwebtoken'
import dotenv from 'dotenv';

dotenv.config();

const router = express.Router();

// Read
router.get('/read', (req, res) => {
    counsellorEmail.find()
        .then((counsellorEmail) => res.json(counsellorEmail))
        .catch((err) => res.status(400).json('Error: ' + err));
});

router.patch('/edit', (req, res) => {
    let token = req.body.adminToken

    if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
        counsellorEmail.findById(req.body.id)
        .then((counsellorEmail) => {
            counsellorEmail.name = req.body.name;
            counsellorEmail.counsEmail = req.body.counsEmail;
            counsellorEmail.receiveEmail = req.body.receiveEmail;


            counsellorEmail.save()
                .then(() => res.json('Couns updated'))
                .catch((err) => res.status(400).json('Error1: ' + err
));
        })
        .catch((err) => res.status(400).json('Error2: ' + err));
    } else {
        res.status(403).json('Invalid Token')
    }
});

// Create
router.post('/create', (req, res) => {
    const name = req.body.name;
    const counsEmail = req.body.counsEmail;
    const receiveEmail = req.body.receiveEmail;
    let token = req.body.adminToken

    if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
        const newCounsEmail = new counsellorEmail({
            name,
            counsEmail,
            receiveEmail,
        });

        newCounsEmail.save()
        .then(() => res.json('Counsellor email added'))
        .catch((err) => res.status(400).json('Error: ' + err));
```

```
57      } else {
58          res.status(403).json('Invalid Token')
59      }
60  });
61
62  router.delete('/remove', (req, res) => {
63      let token = req.body.adminToken
64
65      if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
66          counsellorEmail.findByIdAndDelete(req.body.id)
67          .then(() => {res.json('Couns deleted');})
68          .catch((err) => {res.status(400).json('Error: ' + err)});
69      } else {
70          res.status(403).json('Invalid Token')
71      }
72  });
73
74  export default router;
75
```

```
// Login route, featuring all endpoints

import express from 'express';
import dotenv from 'dotenv';
import jwt from 'jsonwebtoken'
import counsellorEmail from '../models/counsellorEmail.model.js';

dotenv.config();

const router = express.Router();

router.post('/login', async (req,res) =>{
    const password = req.body.password;
    const email = req.body.email;

    const exists = await counsellorEmail.exists({counsEmail:email});

    if (password === process.env.ADMIN_PASSWORD && exists){
        const token = jwt.sign({email:email},process.env.JWT_SECRET,{ expiresIn: 86400 });
        res.status(200).json({adminToken:token});
    } else{
        res.status(200).json('FALSE');
    }
});

router.post('/verify', (req,res) =>{
    const token = req.body.adminToken;
    if(!!token){
        const verified = jwt.verify(token, process.env.JWT_SECRET)
        if (verified){
            res.status(200).json(true);
        } else {
            res.status(200).json(false)
        }
    } else {
        res.status(200).json(false)
    }
});

export default router;
```

```
// Booking route, featuring all endpoints

import express, { query } from 'express';
import bookingDateInfo from '../models/bookingDateInfo.model.js';
import counsellorEmail from '../models/counsellorEmail.model.js';
import nodemailer from 'nodemailer';
import dotenv from 'dotenv';
import axios from 'axios'
import jwt from 'jsonwebtoken'

dotenv.config();

// Create nodemail transporter with gmail authorization
let transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        type: 'OAUTH2',
        user: 'bishvirtualsignup@gmail.com',
        clientId: process.env.OAUTH_CLIENTID,
        clientSecret: process.env.OAUTH_CLIENT_SECRET,
        refreshToken: process.env.OAUTH_REFRESH_TOKEN,
        accessToken: process.env.OAUTH_ACCESS_TOKEN
    },
    tls: {
        rejectUnauthorized: false,
    },
});

async function getLogo(uniName){
    let query = (uniName.replace(/ /g,"+")) + "+Logo" // Create query string
    let queryString =
    `https://serpapi.com/search.json?engine=google&q=${query}&google_domain=google.com&gl=us&hl=en&safe=active&tbm=isch&api_key=${process.env.IMAGE_API_KEY}`
    let res = await axios.get(queryString) // Get result
    let imageURL = await res.data.images_results[0].thumbnail // Filter for first result
    return await imageURL
}

const router = express.Router();

// Get data for calendar formatting
router.get('/readforform', (req, res) => {
    bookingDateInfo.find(
        { aptDate: { $gte: new Date() } },
        { aptDate: 1, status: 1, 'booking.uniRegion': 1 })
    .sort('aptDate')
    .then((bookingDateInfo) => res.json(bookingDateInfo))
    .catch((err) => res.status(400).json('Error: ' + err));
});

// Create a booking
router.post('/create', (req, res) => {
    const aptDate = req.body.aptDate;
    const status = req.body.status;
```

```
 54     const uniName = req.body.booking.uniName;
 55     const uniRepName = req.body.booking.uniRepName;
 56     const uniRepJobTitle = req.body.booking.uniRepJobTitle;
 57     const uniRepEmail = req.body.booking.uniRepEmail;
 58     const uniRegion = req.body.booking.uniRegion;
 59
 60     getLogo(uniName)
 61     .then( logoUrl => {
 62         const newBooking = {
 63             aptDate: aptDate,
 64             status: status,
 65             booking: {
 66                 uniName,
 67                 uniRepName,
 68                 uniRepJobTitle,
 69                 uniRepEmail,
 70                 uniRegion,
 71                 logoUrl,
 72             },
 73         };
 74
 75         bookingDateInfo.findOneAndUpdate(
 76             { aptDate: aptDate },
 77         newBooking)
 78         .then(() => {
 79             res.json('Booking created');
 80             let mailOptions = {
 81                 from: 'BISH Signup <bishvirtualsignup@gmail.com> ',
 82                 to: uniRepEmail,
 83                 subject: 'Your presentation information',
 84                 text: `Thank you for signing up ${uniRepName}!
 85                 Your presentation is at ${aptDate}`,
 86             };
 87
 88             transporter.sendMail(mailOptions, (err, info) => {
 89                 if (err) {
 90                     console.log(err);
 91                 } else {
 92                     console.log('Message sent');
 93                 }
 94             });
 95
 96             counsellorEmail.find({ receiveEmail: true })
 97             .then((counsList)=>{
 98                 counsList.forEach((couns)=>{
 99                     let counsMailOptions = {
100                         from: 'BISH Signup <bishvirtualsignup@gmail.
    com> ',
101                         to: couns.counsEmail,
102                         subject: 'Booking Created',
103                         text: `${uniRepName} has just registered for
    the: ${uniName}!
104                         The date they registered for is: ${aptDate}
105                         Representative's job title: ${uniRepJobTitle}
106                         Representative's email: ${uniRepEmail}
107                         University's region: ${uniRegion}`,
108                     }
```

```
109
110                     transporter.sendMail(counsMailOptions, (err, info
    ) => {
111                         if (err) {
112                             console.log(err);
113                         } else {
114                             console.log('Message sent');
115                         }
116                     });
117                 })
118             })
119         })
120         .catch((err) => res.status(400).json('Error: ' + err));
121     })
122 });
123
124
125 // Get data for display
126 router.get('/readfordisplay', (req, res) => {
127     bookingDateInfo.find(
128         { aptDate: { $gte: new Date() }, status:"Booked"},
129         { aptDate: 1, 'booking': 1 })
130         .sort('aptDate')
131     .then((bookingDateInfo) => res.json(bookingDateInfo))
132     .catch((err) => res.status(400).json('Error: ' + err));
133 });
134
135 // Edit a booking
136 router.patch('/editbooking', (req, res) => {
137     let allowMail = req.body.allowMail;
138     let token = req.body.adminToken
139
140     if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
141         bookingDateInfo.findById(req.body.id)
142         .then((bookingDateInfo) => {
143             bookingDateInfo.booking.uniName = req.body.booking.
    uniName;
144             bookingDateInfo.booking.uniRepName = req.body.booking.
    uniRepName;
145             bookingDateInfo.booking.uniRepJobTitle = req.body.booking
    .uniRepJobTitle;
146             bookingDateInfo.booking.uniRepEmail = req.body.booking.
    uniRepEmail;
147             bookingDateInfo.booking.uniRegion = req.body.booking.
    uniRegion;
148             bookingDateInfo.booking.logoUrl = req.body.booking.
    logoUrl;
149             bookingDateInfo.aptDate = req.body.aptDate;
150
151             bookingDateInfo.save()
152             .then(() => {
153                 res.json('Booking updated')
154                 if(allowMail){
155                     let mailOptions = {
156                         from: 'BISH Signup <bishvirtualsignup@gmail.
    com> ',
157                         to: req.body.booking.uniRepEmail,
```

```
158                         subject: 'Updated presentation information',
159                         text: `Your booking information for a
    presentation with BISH has changed, new information:
160                         University Name: ${req.body.booking.uniName}
161                         Name ${req.body.booking.uniRepName}
162                         Job title: ${req.body.booking.uniRepJobTitle}
163                         Representative's email: ${req.body.booking.
    uniRepEmail}
164                         University's region: ${req.body.booking.
    uniRegion}
165                         Presentation date: ${req.body.aptDate}`,
166                     };
167 
168                     transporter.sendMail(mailOptions, (err, info
    ) => {
169                         if (err) {
170                             console.log(err);
171                         } else {
172                             console.log('Message sent');
173                         }
174                     });
175 
176                     counsellorEmail.find({ receiveEmail: true })
177                     .then((counsList)=>{
178                         counsList.forEach((couns)=>{
179                             let counsMailOptions = {
180                                 from: 'BISH Signup <bishvirtualsignup
    @gmail.com> ',
181                                 to: couns.counsEmail,
182                                 subject: 'Updated booking information
    ',
183                                 text: `Updates have been made to a
    registered booking, new information:
184                                 University Name: ${req.body.booking.
    uniName}
185                                 Name ${req.body.booking.uniRepName}
186                                 Job title: ${req.body.booking.
    uniRepJobTitle}
187                                 Representative's email: ${req.body.
    booking.uniRepEmail}
188                                 University's region: ${req.body.
    booking.uniRegion}
189                                 Presentation date: ${req.body.aptDate
    }
190                                 Logo URL: ${req.body.booking.logoUrl
    }`,
191                             }
192 
193                             transporter.sendMail(counsMailOptions, (
    err, info) => {
194                                 if (err) {
195                                     console.log(err);
196                                 } else {
197                                     console.log('Message sent');
198                                 }
199                             });
200                         })
```

```
                    })
                }
            })
            .catch((err) => res.status(400).json('Error: ' + err));
        })
        .catch((err) => res.status(400).json('Error: ' + err));
    } else {
        res.status(403).json('Invalid Token')
    }
});

// Remove a booking
router.delete('/remove', (req, res) => {
    let token = req.body.adminToken

    if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
        bookingDateInfo.findByIdAndDelete(req.body.id)
        .then(() => {res.json('Booking deleted');})
        .catch((err) => {res.status(400).json('Error: ' + err)});
    } else {
        res.status(403).json('Invalid Token')
    }
});

// Edit dates
router.post('/editcalendar', (req, res) =>{
    let changeLog = req.body.changeLog
    let token = req.body.adminToken

    if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
        let changes =  []
        Object.keys(changeLog).forEach((key) => {
            if(changeLog[key]==true){
                changes.push({'insertOne':{"document":{aptDate:key,
    status:"Available"}}})
            } else {
                changes.push({'deleteOne':{"filter":{aptDate:key}}})
            }
        });

        bookingDateInfo.bulkWrite(changes)
        .then(() => {
            res.json('Dates updated')
            counsellorEmail.find(
                { receiveEmail: true },
                { counsEmail: 1 }
            )
            .then((counsList)=>{
                Object.keys(changeLog).forEach((key) => {
                    let text = '';
                    if(changeLog[key]==true){
                        text = `${text} \n ${key} - Added`
                    } else {
                        text = `${text} \n ${key} - Removed`
                    }
                })

```

```
257                 counsList.forEach((couns)=>{
258                     let counsMailOptions = {
259                         from: 'BISH Signup <bishvirtualsignup@gmail.
    com> ',
260                         to: couns.counsEmail,
261                         subject: 'Changes to available dates',
262                         text: `Updates have been made to the list of
    available dates:
263                         ${text}`,
264                     }
265 
266                     transporter.sendMail(counsMailOptions, (err, info
    ) => {
267                         if (err) {
268                             console.log(err);
269                         } else {
270                             console.log('Message sent');
271                         }
272                     });
273                 })
274             })
275         })
276         .catch((err) => {res.status(400).json('Error: ' + err)})
277     } else {
278         res.status(403).json('Invalid Token')
279     }
280 });
281 
282 
283 // Manually create a booking for insomnia
284 router.post('/createm', (req, res) => {
285     const aptDate = req.body.aptDate;
286     const status = req.body.status;
287     const uniName = req.body.booking.uniName;
288     const uniRepName = req.body.booking.uniRepName;
289     const uniRepJobTitle = req.body.booking.uniRepJobTitle;
290     const uniRepEmail = req.body.booking.uniRepEmail;
291     const uniRegion = req.body.booking.uniRegion;
292 
293     getLogo(uniName).then( logoUrl => {
294         const newBooking = new bookingDateInfo({
295             aptDate: aptDate,
296             status: status,
297             booking: {
298                 uniName,
299                 uniRepName,
300                 uniRepJobTitle,
301                 uniRepEmail,
302                 uniRegion,
303                 logoUrl,
304             },
305         });
306 
307         newBooking.save()
308         .then(() => res.json('Booking created'))
309         .catch((err) => res.status(400).json('Error: ' + err));
310     })
```

```
311 });
312
313 export default router;
314
```

```
// Electron portion of the app (Mostly boilerplate)

const { app, BrowserWindow } = require('electron')

const createWindow = () => { // Create an electron window
    const win = new BrowserWindow({
      width: 800,
      height: 600
    })

    win.loadURL('http://localhost:3000') // Display the electron app onto the window
  }

app.whenReady().then(() => {
    createWindow()
})
```

```
 1 # Getting Started with Create React App
 2
 3 This project was bootstrapped with [Create React App](https://github.
   com/facebook/create-react-app).
 4
 5 ## Available Scripts
 6
 7 In the project directory, you can run:
 8
 9 ### `npm start`
10
11 Runs the app in the development mode.\
12 Open [http://localhost:3000](http://localhost:3000) to view it in your
    browser.
13
14 The page will reload when you make changes.\
15 You may also see any lint errors in the console.
16
17 ### `npm test`
18
19 Launches the test runner in the interactive watch mode.\
20 See the section about [running tests](https://facebook.github.io/
   create-react-app/docs/running-tests) for more information.
21
22 ### `npm run build`
23
24 Builds the app for production to the `build` folder.\
25 It correctly bundles React in production mode and optimizes the build
   for the best performance.
26
27 The build is minified and the filenames include the hashes.\
28 Your app is ready to be deployed!
29
30 See the section about [deployment](https://facebook.github.io/create-
   react-app/docs/deployment) for more information.
31
32 ### `npm run eject`
33
34 **Note: this is a one-way operation. Once you `eject`, you can't go
   back!**
35
36 If you aren't satisfied with the build tool and configuration choices
   , you can `eject` at any time. This command will remove the single
   build dependency from your project.
37
38 Instead, it will copy all the configuration files and the transitive
   dependencies (webpack, Babel, ESLint, etc) right into your project so
   you have full control over them. All of the commands except `eject`
   will still work, but they will point to the copied scripts so you can
   tweak them. At this point you're on your own.
39
40 You don't have to ever use `eject`. The curated feature set is
   suitable for small and middle deployments, and you shouldn't feel
   obligated to use this feature. However we understand that this tool
   wouldn't be useful if you couldn't customize it when you are ready for
    it.
41
```

```
## Learn More

You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/getting-started).

To learn React, check out the [React documentation](https://reactjs.org/).

### Code Splitting

This section has moved here: [https://facebook.github.io/create-react-app/docs/code-splitting](https://facebook.github.io/create-react-app/docs/code-splitting)

### Analyzing the Bundle Size

This section has moved here: [https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size](https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size)

### Making a Progressive Web App

This section has moved here: [https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app](https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app)

### Advanced Configuration

This section has moved here: [https://facebook.github.io/create-react-app/docs/advanced-configuration](https://facebook.github.io/create-react-app/docs/advanced-configuration)

### Deployment

This section has moved here: [https://facebook.github.io/create-react-app/docs/deployment](https://facebook.github.io/create-react-app/docs/deployment)

### `npm run build` fails to minify

This section has moved here: [https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify](https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify)
```

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*

```

```
{
  "name": "electron-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.2",
    "@testing-library/react": "^12.1.4",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^0.26.1",
    "bootstrap": "^4.6.0",
    "concurrently": "^7.0.0",
    "cross-env": "^7.0.3",
    "dayjs": "^1.10.8",
    "electron": "^17.1.2",
    "react": "^17.0.2",
    "react-bootstrap": "^1.6.1",
    "react-dom": "^17.0.2",
    "react-scripts": "5.0.0",
    "wait-on": "^6.0.1",
    "web-vitals": "^2.1.4"
  },
  "main": "main.js",
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "watch": "webpack --config webpack.common.js --watch",
    "electron:start": "electron .",
    "electron-react": "concurrently \"cross-env BROWSER=none npm start
  \" \"wait-on http://localhost:3000 && electron .\""
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

```
 1 import Display from "./components/Display"
 2
 3 const App = () => {
 4     return(
 5     <div>
 6       <Display/>
 7     </div>
 8   )
 9 }
10
11 export default App
12
```

```
 1 import React from 'react';
 2 import ReactDOM from 'react-dom';
 3 import './index.css';
 4 import App from './App';
 5
 6 ReactDOM.render(
 7   <React.StrictMode>
 8     <App />
 9   </React.StrictMode>,
10   document.getElementById('root')
11 );
12
13
14
```

```
.cardUni{
    font-size: 1.5rem;
    font-weight: 300;
}

.cardRegion{
    font-size: 1rem;
    font-weight: 500;
    color: #6c757d;
}

.cardImage{
    width: calc(100% + 2px);
    border-radius: 0;
    margin-left: -1px;
    border-left: 1px solid #DFDFDF;
    border-right: 1px solid #DFDFDF;
}

.cardDate{

}

.cardTitle{

}

.cardEmail{

}

.pad{
    margin-top: 1rem;
    margin-bottom: 1rem;
}

.cardButton{
    width: 100%;
}

.login-container{
    display: flex;
    justify-content: center;
    align-items: center;
}

.ccontainer {
    margin-left: calc(50vw - 20rem);
    margin-top: calc(50vh - 15rem);
}

.cbooked {
    color: hsla(3, 100%, 20%, 1) !important;
    background-color: rgba(255, 65, 54, 0.6) !important;
}

.cbooked:hover {
```

```
    color: hsla(3, 100%, 20%, 1) !important;
    background-color: rgba(255, 65, 54, 0.55) !important;
}

.cavailable {
    color: hsla(127, 63%, 15%, 1) !important;
    background-color: rgba(1, 255, 112, 0.7) !important;
}

.cavailable:hover {
    color: hsla(127, 63%, 15%, 1) !important;
    background-color: rgba(1, 255, 112, 0.5) !important;
}

.topbar{
    height: 10vh;
    background: #f8f9fa;
    position:absolute;
    width:100%;
    text-align: center;
    padding-top: 2vh;
}

.botbar{
    bottom: 0;
    height: 10vh;
    background: #f8f9fa;
    position:absolute;
    width:100%;
}

.va{
    display: flex;
    justify-content: center;
    align-items: center;
    padding-top: 15vh;
}
```

```
 1 body {
 2   margin: 0;
 3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto'
   , 'Oxygen',
 4     'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue'
   ,
 5     sans-serif;
 6   -webkit-font-smoothing: antialiased;
 7   -moz-osx-font-smoothing: grayscale;
 8 }
 9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New'
   ,
12     monospace;
13 }
14
```

```
// Booking card template

import {useState} from "react";
import dayjs from "dayjs";
import Card from 'react-bootstrap/Card'
import image from './undraw_Page_not_found_re_e9o6.png'
import '../admin.css'


const BCard = ({instance}) => {
  const [date, setDate] = useState(instance.aptDate);
  const [bookingData, setBookingData] = useState({
        uniName: instance.booking.uniName,
        uniRegion: instance.booking.uniRegion,
        uniRepJobTitle: instance.booking.uniRepJobTitle,
        uniRepName: instance.booking.uniRepName,
        uniRepEmail: instance.booking.uniRepEmail,
        logoUrl:instance.booking.logoUrl,
    });

  return (
      <>
      <div className='col-4 pad'>
        <Card style={{ width: '100%' }}>
            <Card.Body>
                <Card.Title className='cardUni'>{bookingData.uniName
}</Card.Title>
                <Card.Subtitle className='cardRegion'>{bookingData.
uniRegion}</Card.Subtitle>
            </Card.Body>
            <Card.Img variant="top" src={!!bookingData.logoUrl?
bookingData.logoUrl:image} className='cardImage'/>
            <Card.Body>
                <Card.Text className='cardDate'>{dayjs(date).format('
dddd, MMMM D')}</Card.Text>
                <Card.Text className='cardTitle'>{bookingData.
uniRepJobTitle + ': ' + bookingData.uniRepName}</Card.Text>
                <Card.Text className='cardEmail'>{bookingData.
uniRepEmail}</Card.Text>
            </Card.Body>
        </Card>
      </div>
      </>
  )
}

export default BCard
```

```
 1 // Iterate through bookings and use instances to display booking cards
 2
 3 import {useEffect, useState} from "react";
 4 import BCard from "./BCard";
 5 import '../admin.css'
 6 import axios from 'axios';
 7
 8 const Display = () => {
 9     const [booking, setBooking] = useState([])
10
11     // eslint-disable-next-line
12     useEffect(async () => {
13         async function getDisplayData() { // Async to prevent stalling

14             let res = await axios.get('http://localhost:5000/booking/
   readfordisplay');
15             return res;
16         }
17
18         getDisplayData().then((res)=>setBooking(res.data.slice(0,3)))
   // Only get the first 3 upcoming bookings
19     }, [])
20
21     return (
22         <>
23         <div className='topbar'><h1>Upcoming Bookings</h1></div>
24         <div className='botbar'></div>
25         <div className='container va'>
26             {booking.map(instance=>{
27                 return(
28                     <BCard instance={instance} setBooking={setBooking
   }/>
29                 )
30             })}
31         </div>
32         </>
33     )
34 }
35
36 export default Display
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link
  rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
  integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
  crossorigin="anonymous"
/>
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <script src="./build/app.js"></script>
  </body>
</html>

```

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}

```