```
{
  "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"
    ]
  }
}

```

```jsx
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} />

                    <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
```

```
 1 import {Route, Redirect} from 'react-router-dom'
 2
 3 const PrivateRoute = ({component, path, admin}) => {
 4     if(admin){
 5         return(<Route path={path} exact component={component}/>)
 6     } else {
 7         return(<Redirect to='/' />)
 8     }
 9 }
10
11 export default PrivateRoute
12
```

```
 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
   )
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
```

```
 1 import React, { useEffect, useState } from 'react';
 2 import { BsPencilSquare } from 'react-icons/bs';
 3 import { FaCalendarPlus, FaCalendarAlt,  } from 'react-icons/fa'
 4 import { AiFillLock, AiFillUnlock, AiOutlineQuestionCircle } from '
   react-icons/ai';
 5 import { MdEmail } from 'react-icons/md'
 6 import { Link, NavLink } from 'react-router-dom';
 7 import './sidebar2.css';
 8 import Navbar from 'react-bootstrap/Navbar'
 9 import {Nav, NavDropdown} from 'react-bootstrap';
10 import Login from './Login';
11 import axios from 'axios';
12
13 const Sidebar2 = ({admin,setAdmin}) => {
14     const [sidebar, setSidebar] = useState(false); // Toggle sidebar
   visibility
15     const toggleSidebar = () => setSidebar(!sidebar); // Seperate
   function as bugfix
16
17     const [show, setShow] = useState(false);
18     const [error, setError] = useState(false);
19
20     useEffect(async () => {
21         const token = localStorage.getItem('adminToken')
22         if(!!token){
23             const verify = await axios.post('http://localhost:5000/
   login/verify', {adminToken:token})
24             if(verify){
25                 setAdmin(true)
26             }
27         }
28     }, [])
29
30     function isAdmin(){
31         if(admin){
32             return(
33                 <NavDropdown title={<span><AiFillUnlock size={22}/>
   Admin</span>} className="nav-text">
34                     <NavDropdown.Item href="#edit">
35                         <Nav.Link as={NavLink} to='/admin/booking'
   className='nopad'>
36                             <span><FaCalendarPlus size={20}/> Bookings
   </span>
37                         </Nav.Link>
38                     </NavDropdown.Item>
39
40                     <NavDropdown.Item href="#dates">
41                         <Nav.Link as={NavLink} to='/admin/calendar'
   className='nopad'>
42                             <span><FaCalendarAlt size={20}/> Calendar
   </span>
43                         </Nav.Link>
44                     </NavDropdown.Item>
45
46                     <NavDropdown.Item href="#email">
47                         <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
```

```
 1 import React, { useState, useEffect } from 'react';
 2 import 'bootstrap/dist/css/bootstrap.min.css';
 3 import Button from 'react-bootstrap/Button';
 4 import TextForm from './TextForm';
 5 import TimeDisplay from './TimeDisplay';
 6 import DateCalendar from './DateCalendar';
 7 import axios from 'axios';
 8 import dayjs from 'dayjs';
 9 import isBetween from 'dayjs/plugin/isBetween';
10 import validateInfo from './validateForm';
11
12 dayjs.extend(isBetween);
13
14 // Initialize calendar states
15 let available = [];
16 let booked = [];
17 let regionBooked = [];
18
19 function FormPage() {
20     let calendarData;
21
22     const [checkDate, setCheckDate] = useState(false); // Used to
   manually re-render after validation
23     const [key, setKey] = useState(false); // Used to manually re-
   render after validation
24
25     const [validated, setValidated] = useState(false); // Used to
   prevent validation tooltips from appearing before submission
26     const [error, setError] = useState({}); // Array of validation
   errors
27     const [success, setSuccess] = useState(false);
28     const [dateData, setDateData] = useState(new Date()); // Store
   appointment date
29     const [formData, setFormData] = useState({ // Store booking
   information
30         uniName: '',
31         uniRepName: '',
32         uniRepJobTitle: '',
33         uniRepEmail: '',
34         uniRegion: '',
35     });
36
37
38     // Submission handler
39     const handleSubmit = (e) => {
40         e.preventDefault();
41
42         const bookingDateInfoInstance = {
43             aptDate: dateData,
44             status: 'Booked',
45             booking: {
46                 uniName: formData.uniName,
47                 uniRepName: formData.uniRepName,
48                 uniRepJobTitle: formData.uniRepJobTitle,
49                 uniRepEmail: formData.uniRepEmail,
50                 uniRegion: formData.uniRegion,
51             },};
```

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

```
                        .subtract(7, 'day')
                        .format('DD/MM/YYYY');

                    if (index - 1 >= 0) {
                        // Checks last available day
                        if (calendarData[index - 1].status === '
    Available' && dayjs(calendarData[index - 1])
                        .isBetween(lowB,uppB,null,'[]')) {
                            regionBooked.push(dayjs(calendarData[
    index - 1].aptDate).format('DD/MM/YYYY'));
                            calendarData[index - 1].status = 'Region
    booked'; // Prevent further formatting
                        }
                    }

                    if (index + 1 <= calendarData.length - 1) {
                        // Checks next available day
                        if (calendarData[index + 1].status === '
    Available' && dayjs(calendarData[index + 1])
                        .isBetween(lowB,uppB,null,'[]')) {
                            regionBooked.push(dayjs(calendarData[
    index + 1].aptDate).format('DD/MM/YYYY'));
                            calendarData[index + 1].status = 'Region
    booked';
                        }
                    }
                }
            });

            calendarData.forEach((instance) => {
                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'));
                }
            });

            // Deselect current date if it became booked/region
    booked during selection
            if (booked.includes(dayjs(dateData).format('DD/MM/YYYY'
    )) || regionBooked.includes(dayjs(dateData).format('DD/MM/YYYY'))) {
                setDateData(new Date());
            }

            setKey(!key); // Manually re-render
        }
    }, [formData.uniRegion, checkDate]); // Dependencies which, after
     change, will trigger the useEffect


    return (
        <div className='container'>
            <div className='row align-items-center'>
                <div className='col-6'>
```

```
                    <TextForm
                        validated={validated}
                        errors={error}
                        isReadOnly={success}
                        formData={formData}
                        setFormData={setFormData}
                    />
                </div>
                <div className='col-6'>
                    <DateCalendar
                        validated={validated}
                        errors={error}
                        available={available}
                        booked={booked}
                        regionBooked={regionBooked}
                        success={success}
                        selectedDate={dateData}
                        setSelectedDate={setDateData}
                    />
                </div>
            </div>
            <div className='row align-items-center'>
                <div className='col-6'>
                    <TimeDisplay date={dateData} />
                </div>

                <div className='col-6'>
                    <div className='submitAlign'>
                        <Button
                            className={success ? 'disabled button' :
    'button'}
                            as='input'
                            type='submit'
                            value='Submit'
                            onClick={handleSubmit}
                        />
                    </div>
                </div>
            </div>
            <h5
                id='successSubmit'
                className={success ? 'alert-success' : 'hidden'}>
                Thank you for signing up for the BISH virtual
    presentation, you
                will receive an email shortly with the details of
    your booking.
            </h5>
        </div>
    );
}

export default FormPage;

```

```
 1 import React from 'react';
 2 import Form from 'react-bootstrap/Form';
 3
 4 const TextForm = ({
 5     validated,
 6     errors,
 7     isReadOnly,
 8     formData,
 9     setFormData
10 }) => {
11     return (
12         <div className='formAlign'>
13             <Form validated={validated}>
14                 <Form.Group className='mb-3' controlId='form.
   University'>
15                     <Form.Label>University</Form.Label>
16                     <Form.Control
17                         required
18                         readOnly={isReadOnly ? true : false}
19                         as='input'
20                         type='text'
21                         defaultValue={formData.uniName}
22                         onChange={(e) => {
23                             setFormData({...formData,uniName: e.target
   .value,});
24                             delete errors.uniRepName;
25                         }}
26                     />
27                     <Form.Control.Feedback type='invalid'>
28                         {errors.uniName}
29                     </Form.Control.Feedback>
30                 </Form.Group>
31
32                 <Form.Group className='mb-3' controlId='form.FullName'
   >
33                     <Form.Label>Full name</Form.Label>
34                     <Form.Control
35                         required
36                         readOnly={isReadOnly ? true : false}
37                         as='input'
38                         type='text'
39                         defaultValue={formData.uniRepName}
40                         onChange={(e) => {
41                             setFormData({...formData,uniRepName: e.
   target.value,});
42                             delete errors.uniRepName;
43                         }}
44                     />
45                     <Form.Control.Feedback type='invalid'>
46                         {errors.uniRepName}
47                     </Form.Control.Feedback>
48                 </Form.Group>
49
50                 <Form.Group className='mb-3' controlId='form.JobTitle'
   >
51                     <Form.Label>Job title</Form.Label>
52                     <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>
```

```
104                     <Form.Control.Feedback type='invalid'>
105                         {errors.uniRegion}
106                     </Form.Control.Feedback>
107                 </Form.Group>
108             </Form>
109         </div>
110     );
111 };
112
113 export default TextForm;
114
```

```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
```

```
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 (
        <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;

```

```
 1 import React from 'react';
 2 import Calendar from 'react-calendar';
 3 import './Calendar.css'
 4 // import 'react-calendar/dist/Calendar.css'; <- Old CSS file
 5 import dayjs from 'dayjs';
 6
 7 const DateCalendar = ({
 8     errors,
 9     validated,
10     available,
11     booked,
12     regionBooked,
13     success,
14     selectedDate,
15     setSelectedDate,
16 }) => {
17     return (
18         <div className='calendarAlign disableCalendar'>
19             <div className={validated && !!errors.date ? '
   invalidCalendar' : ''}> {/*Control validation formatting*/}
20                 <Calendar
21                     onChange={setSelectedDate}
22                     value={selectedDate}
23                     maxDate={new Date(dayjs().add(11, 'month').endOf('
   month').toDate())}
24                     minDate={new Date()}
25                     next2Label=''
26                     prev2Label=''
27                     minDetail='year'
28                     showNeighboringMonth='false'
29                     tileClassName={({ date, view }) => { {/*
   Conditional formatting*/}
30                         if (dayjs(date).format('DD/MM/YYYY') === dayjs
   (selectedDate).format('DD/MM/YYYY')) {
31                             return 'selected';
32
33                         } else if (available.includes(dayjs(date).
   format('DD/MM/YYYY'))) {
34                             return 'available';
35
36                         } else if (booked.includes(dayjs(date).format(
   'DD/MM/YYYY'))) {
37                             return 'booked';
38
39                         } else if (regionBooked.includes(dayjs(date).
   format('DD/MM/YYYY'))) {
40                             return 'regionBooked';
41
42                         } else if (date > new Date()) {
43                             return 'unavailable';
44
45                         } else {
46                             return 'past';
47                         }
48                     }}
49                     tileDisabled={({ date, view }) => { {/*Conditional
    disabling of dates*/}
```

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

```
 1 import dayjs from 'dayjs';
 2
 3 export default function validateInfo(info) {
 4     let errors = {};
 5     const regex =
 6     /^(([^<>()[\]\\.,;:\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,}))$/;
 7
 8     if (dayjs(info.aptDate).format('DD/MM/YYYY') === dayjs(new Date
   ()).format('DD/MM/YYYY')) {
 9         errors.date = 'Date required';
10     }
11     if (info.booking.uniName === '') {
12         errors.uniName = 'University name required';
13     }
14     if (info.booking.uniRepName === '') {
15         errors.uniRepName = 'Name required';
16     }
17     if (info.booking.uniRepJobTitle === '') {
18         errors.uniRepJobTitle = 'Job title required';
19     }
20     if (info.booking.uniRepEmail === '') {
21         errors.uniRepEmail = 'Email required';
22
23     } else if (!regex.test(info.booking.uniRepEmail)) {
24         errors.uniRepEmail = 'Invalid email';
25     }
26     if (info.booking.uniRegion === '') {
27         errors.uniRegion = 'Region required';
28     }
29
30     return errors;
31 }
32
```

```css
.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;
}
```

```
 1 import {useState} from "react";
 2 import Card from 'react-bootstrap/Card'
 3 import '../admin.css'
 4 import Button from 'react-bootstrap/Button';
 5 import Modal from 'react-bootstrap/Modal';
 6 import Form from 'react-bootstrap/Form'
 7 import axios from 'axios';
 8 import validateCouns from "./validateCouns";
 9
10 const CCard = ({instance,setEmails}) => {
11     const [show, setShow] = useState(false);
12     const [validated, setValidated] = useState(false);
13     const [error, setError] = useState({});
14
15     const [counsInfo, setCounsInfo] = useState({
16         name:instance.name,
17         counsEmail:instance.counsEmail,
18         receiveEmail:instance.receiveEmail,
19     })
20
21     const [tempCounsInfo, setTempCounsInfo] = useState({
22         name:instance.name,
23         counsEmail:instance.counsEmail,
24         receiveEmail:instance.receiveEmail,
25     })
26
27     function cancel(){
28         setCounsInfo(tempCounsInfo)
29         setShow(false)
30     }
31
32     function handleOpen(){
33         setTempCounsInfo(counsInfo)
34         setShow(true);
35         setValidated(false);
36     }
37
38     async function getEmail(){
39         let res = await  axios.get('http://localhost:5000/couns/read')
40         return res
41     }
42
43     async function edit(){
44         if(!(Object.keys(validateCouns(counsInfo)).length === 0)){
45             setError(validateCouns(counsInfo));
46             setValidated(true);
47         } else{
48             const token = localStorage.getItem('adminToken')
49             let verify;
50
51             if(!!token) {
52                 verify = await axios.post('http://localhost:5000/login
   /verify', {adminToken:token});
53             } else {
54                 verify =  false;
55             }
56
```

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

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

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

```
 1 import {useState} from "react";
 2 import Card from 'react-bootstrap/Card'
 3 import '../admin.css'
 4 import Button from 'react-bootstrap/Button';
 5 import Modal from 'react-bootstrap/Modal';
 6 import Form from 'react-bootstrap/Form'
 7 import axios from 'axios';
 8 import validateCouns from "./validateCouns";
 9
10 const CCardAdd = ({setEmails}) => {
11     const [show, setShow] = useState(false);
12     const [validated, setValidated] = useState(false);
13     const [error, setError] = useState({});
14
15     const [counsInfo, setCounsInfo] = useState({
16         name: "Name",
17         counsEmail: "Email",
18         receiveEmail:true,
19     })
20
21     const [tempCounsInfo, setTempCounsInfo] = useState({
22         name: "Name",
23         counsEmail: "Email",
24         receiveEmail:true,
25     })
26
27     function cancel(){
28         setCounsInfo(tempCounsInfo)
29         setShow(false)
30     }
31
32     function handleOpen(){
33         setTempCounsInfo(counsInfo)
34         setShow(true);
35         setValidated(false);
36     }
37
38     async function getEmail(){
39         let res = await  axios.get('http://localhost:5000/couns/read')
40         return res
41     }
42
43     async function create(){
44         if(!(Object.keys(validateCouns(counsInfo)).length === 0)){
45             setError(validateCouns(counsInfo));
46             setValidated(true);
47         } else{
48             const token = localStorage.getItem('adminToken')
49             let verify;
50
51             if(!!token) {
52                 verify = await axios.post('http://localhost:5000/login
   /verify', {adminToken:token});
53             } else {
54                 verify =  false;
55             }
56
```

```
 57              if(verify){
 58                  setValidated(false);
 59
 60                  counsInfo.adminToken = token
 61
 62                  axios.post('http://localhost:5000/couns/create',
    counsInfo)
 63                  .then(()=>{
 64                      getEmail()
 65                      .then((res)=>{
 66                          setEmails(res.data)
 67                      })
 68                  }).catch((err)=>{
 69                      console.log(err)
 70                      alert(err)
 71                  })
 72                  setShow(false);
 73              } else{
 74                  alert('Invalid authentication token')
 75              }
 76          }
 77      }
 78
 79      return (
 80          <>
 81            <Modal show={show} onHide={cancel} backdrop="static">
 82                  <Modal.Header closeButton>
 83                      <Modal.Title>Name</Modal.Title>
 84                  </Modal.Header>
 85                  <Modal.Body>
 86                      <Form>
 87                          <Form.Group>
 88                              <Form.Label> Counsellor Name </Form.Label
    >
 89                              <Form.Control
 90                                  type='text'
 91                                  onChange={(e) => {
 92                                      setCounsInfo({...counsInfo,name:
    e.target.value,})
 93                                  }}
 94                                  isInvalid={validated?!!error.name:
    false}
 95                                  isValid={validated?!error.name:false}
 96                              />
 97                          </Form.Group>
 98                          <Form.Group>
 99                              <Form.Label> Email </Form.Label>
100                              <Form.Control
101                                  type='text'
102                                  onChange={(e) => {
103                                      setCounsInfo({...counsInfo,
    counsEmail: e.target.value,})
104                                  }}
105                                  isInvalid={validated?!!error.
    counsEmail:false}
106                                  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>
                            <Button variant="success" className="
cardButton" onClick={() => {handleOpen()}}>Add</Button>
                        </Card.Body>
                    </Card>
                </div>
            </>
        )
}

export default CCardAdd
```

```
 1 import { useEffect, useState } from "react"
 2 import axios from 'axios';
 3 import CCard from "./CCard";
 4 import CCardAdd from "./CCardAdd";
 5
 6 const EditEmail = () => {
 7     const [emails, setEmails] = useState([])
 8
 9     useEffect(async () => {
10         async function getEmail(){
11             let res = await  axios.get('http://localhost:5000/couns/
   read')
12             return res
13         }
14
15         getEmail().then((res)=>setEmails(res.data))
16     }, [])
17
18     return (
19         <div className='container'>
20             <div className='row pad'>
21                 {emails.map(instance=>{
22                     return(
23                       <CCard instance={instance} setEmails={setEmails
   }/>
24                     )
25                 })}
26                 <CCardAdd setEmails={setEmails}/>
27             </div>
28         </div>
29     )
30 }
31
32 export default EditEmail
33
```

```
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
}
```

```
 1 import {useState} from "react";
 2 import Card from 'react-bootstrap/Card'
 3 import image from './undraw_Page_not_found_re_e9o6.png'
 4 import dayjs from "dayjs";
 5 import '../admin.css'
 6 import Button from 'react-bootstrap/Button';
 7 import Modal from 'react-bootstrap/Modal';
 8 import Form from 'react-bootstrap/Form'
 9 import axios from 'axios';
10 import validateInfo from "./validateCreate";
11
12 const BCard = ({instance,setBooking}) => {
13     const [show, setShow] = useState(false);
14     const [validated, setValidated] = useState(false);
15     const [error, setError] = useState({date:''});
16
17     const [date, setDate] = useState(instance.aptDate);
18     const [tempDate, setTempDate] = useState(instance.aptDate);
19
20     const [sendEmail, setSendEmail] = useState(false)
21
22     const [bookingData, setBookingData] = useState({
23         uniName: instance.booking.uniName,
24         uniRegion: instance.booking.uniRegion,
25         uniRepJobTitle: instance.booking.uniRepJobTitle,
26         uniRepName: instance.booking.uniRepName,
27         uniRepEmail: instance.booking.uniRepEmail,
28         logoUrl:instance.booking.logoUrl,
29     });
30     const [tempBooking, setTempBooking] = useState({
31         uniName: instance.booking.uniName,
32         uniRegion: instance.booking.uniRegion,
33         uniRepJobTitle: instance.booking.uniRepJobTitle,
34         uniRepName: instance.booking.uniRepName,
35         uniRepEmail: instance.booking.uniRepEmail,
36         logoUrl:instance.booking.logoUrl,
37     });
38
39     function cancel(){
40         setBookingData(tempBooking)
41         setDate(tempDate)
42         setShow(false);
43     }
44
45     function handleOpen(){
46         setTempBooking(bookingData);
47         setTempDate(date);
48         setValidated(false);
49         setShow(true);
50     }
51
52     async function getDisplayData() { // Async to prevent stalling
53         let res = await axios.get('http://localhost:5000/booking/
   readfordisplay');
54         return res;
55     }
56
```

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

```
data: {id:instance._id,adminToken:token}})
            .then(()=>{
                // eslint-disable-next-line
                getDisplayData()
                .then((res)=>{
                    setBooking(res.data);
                })
            }).then(setShow(false));
        } else{
            alert('Invalid authentication token')
        }
    }

    return (
        <>
            <Modal show={show} onHide={cancel} backdrop="static">
                <Modal.Header closeButton>
                    <Modal.Title>{tempBooking.uniName}</Modal.Title>
                </Modal.Header>
                <Modal.Body>
                    <Form >
                        <Form.Group>
                            <Form.Label> ID </Form.Label>
                            <Form.Control
                                type='text'
                                value={instance._id}
                                readOnly={true}
                            />
                        </Form.Group>
                        <Form.Group>
                            <Form.Label> University Name </Form.Label
    >
                            <Form.Control
                                type='text'
                                defaultValue={tempBooking.uniName}
                                onChange={(e) => {
                                    setBookingData({...bookingData,
    uniName: e.target.value,})
                                }}
                                isInvalid={validated?!!error.uniName:
    false}
                                isValid={validated?!error.uniName:
    false}
                            />
                        </Form.Group>
                        <Form.Group>
                            <Form.Label> University Region </Form.
    Label>
                            <Form.Control
                                type='text'
                                defaultValue={tempBooking.uniRegion}
                                onChange={(e) => {
                                    setBookingData({...bookingData,
    uniRegion: e.target.value,})
                                }}
                                isInvalid={validated?!!error.
    uniRegion:false}
```

```
                                    isValid={validated?!error.uniRegion:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Date </Form.Label>
                                <Form.Control
                                    type='text'
                                    defaultValue={dayjs(tempDate).format('M/D/YYYY')}
                                    onChange={(e)=> {
                                        setDate(new Date(dayjs(e.target.value)))

                                    }}
                                    isInvalid={validated?!!error.date:false}
                                    isValid={validated?!error.date:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Rep Title </Form.Label>
                                <Form.Control
                                    type='text'
                                    defaultValue={tempBooking.uniRepJobTitle}
                                    onChange={(e) => {
                                        setBookingData({...bookingData, uniRepJobTitle: e.target.value,})
                                    }}
                                    isInvalid={validated?!!error.uniRepJobTitle:false}
                                    isValid={validated?!error.uniRepJobTitle:false}
                                />
                            </Form.Group>
                            <Form.Group>
                                <Form.Label> Rep Name </Form.Label>
                                <Form.Control
                                    type='text'
                                    defaultValue={tempBooking.uniRepName}
                                    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'
                                    defaultValue={tempBooking.uniRepEmail}
```

```
204                                 onChange={(e) => {
205                                     setBookingData({...bookingData,
    uniRepEmail: e.target.value,})
206                                 }}
207                                 isInvalid={validated?!!error.
    uniRepEmail:false}
208                                 isValid={validated?!error.uniRepEmail
    :false}
209                             />
210                         </Form.Group>
211                         <Form.Group>
212                             <Form.Label> Image URL </Form.Label>
213                             <Form.Control
214                                 type='text'
215                                 defaultValue={tempBooking.logoUrl}
216                                 onChange={(e) => {
217                                     setBookingData({...bookingData,
    logoUrl: e.target.value,})
218                                 }}
219                                 isValid={validated?true:false}
220                             />
221                         </Form.Group>
222                         <Form.Group>
223                             <Form.Label> Send Email? </Form.Label>
224                             <Form.Control
225                                 type='checkbox'
226                                 defaultChecked={true}
227                                 onChange={(e) => {
228                                     setSendEmail(e.target.value)
229                                 }}
230                                 isValid={validated?true:false}
231                             />
232                         </Form.Group>
233                     </Form>
234                 </Modal.Body>
235                 <Modal.Footer>
236                     <Button variant="danger" onClick={()=>{remove
    ()}} style={{'margin-right':'auto'}}>
237                         Delete Booking
238                     </Button>
239                     <Button variant="primary" onClick={()=>{edit
    ()}} >
240                         Save Changes
241                     </Button>
242                 </Modal.Footer>
243             </Modal>
244 
245             <div className='col-4 pad'>
246                 <Card style={{ width: '100%' }}>
247                     <Card.Body>
248                         <Card.Title className='cardUni'>{bookingData.
    uniName}</Card.Title>
249                         <Card.Subtitle className='cardRegion'>{
    bookingData.uniRegion}</Card.Subtitle>
250                     </Card.Body>
251                     <Card.Img variant="top" src={!!bookingData.
    logoUrl?bookingData.logoUrl:image} className='cardImage'/>
```

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

```
 1 import {useState} from "react";
 2 import Card from 'react-bootstrap/Card'
 3 import image from './undraw_No_data_re_kwbl.png'
 4 import dayjs from "dayjs";
 5 import '../admin.css'
 6 import Button from 'react-bootstrap/Button';
 7 import Modal from 'react-bootstrap/Modal';
 8 import Form from 'react-bootstrap/Form'
 9 import axios from 'axios';
10 import validateInfo from "./validateCreate";
11
12 const BCardAdd = ({setBooking}) => {
13     const [show, setShow] = useState(false);
14     const [validated, setValidated] = useState(false);
15     const [error, setError] = useState(false)
16
17     const [date, setDate] = useState("Month/Day/Year"); // Store
   appointment date
18     const [tempDate, setTempDate] = useState("Month/Day/Year"); //
   Store for cancel
19
20     const [bookingData, setBookingData] = useState({ // Store booking
   information
21         uniName: "University Name",
22         uniRegion: "University Region",
23         uniRepJobTitle: "Rep Title",
24         uniRepName: "Rep Name",
25         uniRepEmail: "Rep Email",
26         logoUrl: "",
27     });
28     const [tempBooking, setTempBooking] = useState({ // Store for
   cancel
29         uniName: "University Name",
30         uniRegion: "University Region",
31         uniRepJobTitle: "Rep Title",
32         uniRepName: "Rep Name",
33         uniRepEmail: "Rep Email",
34         logoUrl: "",
35     });
36
37     function cancel(){
38         setBookingData(tempBooking)
39         setDate(tempDate)
40         setShow(false);
41     }
42
43     function handleOpen(){
44         setTempBooking(bookingData);
45         setTempDate(date);
46         setValidated(false);
47         setShow(true);
48     }
49
50     async function getDisplayData() { // Async to prevent stalling
51         let response = await axios.get('http://localhost:5000/booking/
   readfordisplay');
52         return response;
```

```
 53     }
 54
 55     async function create(){
 56         if(!(Object.keys(validateInfo(bookingData,date)).length === 0
    )){
 57             setError(validateInfo(bookingData,date));
 58             setValidated(true);
 59         } else{
 60             const token = localStorage.getItem('adminToken')
 61             let verify;
 62
 63             if(!!token) {
 64                 verify = await axios.post('http://localhost:5000/
    login/verify', {adminToken:token});
 65             } else {
 66                 verify =  false;
 67             }
 68
 69             if(verify){
 70                 setValidated(false);
 71
 72                 let bookingDateInfoInstance = {
 73                     aptDate: date,
 74                     status: 'Booked',
 75                     booking: {
 76                         uniName: bookingData.uniName,
 77                         uniRepName: bookingData.uniRepName,
 78                         uniRepJobTitle: bookingData.uniRepJobTitle,
 79                         uniRepEmail: bookingData.uniRepEmail,
 80                         uniRegion: bookingData.uniRegion,
 81                         logoUrl: bookingData.logoUrl,
 82                     },
 83                 };
 84
 85                 axios.post('http://localhost:5000/booking/create',
    bookingDateInfoInstance)
 86                 .then(()=>{
 87                     // eslint-disable-next-line
 88                     getDisplayData()
 89                     .then((res)=>{
 90                         setBooking(res.data);
 91                     })
 92                 }).then(setShow(false));
 93             } else{
 94                 alert('Invalid authentication token')
 95             }
 96         }
 97     }
 98
 99     return (
100         <>
101             <Modal show={show} onHide={cancel} backdrop="static">
102                 <Modal.Header closeButton>
103                     <Modal.Title>University Name</Modal.Title>
104                 </Modal.Header>
105                 <Modal.Body>
106                     <Form>
```

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

```
151                         </Form.Group>
152                         <Form.Group>
153                             <Form.Label> Rep Name </Form.Label>
154                             <Form.Control
155                                 type='text'
156                                 onChange={(e) => {
157                                     setBookingData({...bookingData,
    uniRepName: e.target.value,})
158                                 }}
159                                 isInvalid={validated?!!error.
    uniRepName:false}
160                                 isValid={validated?!error.uniRepName:
    false}
161                             />
162                         </Form.Group>
163                         <Form.Group>
164                             <Form.Label> Rep Email </Form.Label>
165                             <Form.Control
166                                 type='text'
167                                 onChange={(e) => {
168                                     setBookingData({...bookingData,
    uniRepEmail: e.target.value,})
169                                 }}
170                                 isInvalid={validated?!!error.
    uniRepEmail:false}
171                                 isValid={validated?!error.uniRepEmail
    :false}
172                             />
173                         </Form.Group>
174                         <Form.Group>
175                             <Form.Label> Image URL </Form.Label>
176                             <Form.Control
177                                 type='text'
178                                 onChange={(e) => {
179                                     setBookingData({...bookingData,
    logoUrl: e.target.value,})
180                                 }}
181                                 isValid={validated?true:false}
182                             />
183                         </Form.Group>
184                     </Form>
185                 </Modal.Body>
186                 <Modal.Footer>
187                     <Button variant="success" onClick={()=>{create
    ()}} >
188                         Create Booking
189                     </Button>
190                 </Modal.Footer>
191             </Modal>
192             <div className='col-4 pad'>
193                 <Card style={{ width: '100%' }}>
194                     <Card.Body>
195                         <Card.Title className='cardUni'>University
    Name</Card.Title>
196                         <Card.Subtitle className='cardRegion'>
    University Region</Card.Subtitle>
197                     </Card.Body>
```

```
198                     <Card.Img variant="top" src={image} className='
    cardImage'/>
199                     <Card.Body>
200                         <Card.Text className='cardDate'>Date</Card.
    Text>
201                         <Card.Text className='cardTitle'>Rep Title:
    Rep Name</Card.Text>
202                         <Card.Text className='cardEmail'>Rep Email</
    Card.Text>
203                         <Button variant="success" className="
    cardButton" onClick={() => {handleOpen()}}>Add</Button>
204                     </Card.Body>
205                 </Card>
206             </div>
207         </>
208     )
209 }
210
211 export default BCardAdd
212
213
```

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

13             let res = await axios.get('http://localhost:5000/booking/
   readfordisplay');
14             return res;
15         }
16
17         getDisplayData().then((res)=>setBooking(res.data))
18     }, [])
19
20     return (
21         <>
22         <div className='container'>
23             <div className='row pad'>
24                 {booking.map(instance=>{
25                     return(
26                       <BCard instance={instance} setBooking={
   setBooking}/>
27                     )
28                 })}
29                 <BCardAdd setBooking={setBooking}/>
30             </div>
31         </div>
32         </>
33     );
34 };
35
36 export default EditBookings;
37
```

```
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
```

```
 1 import Calendar from 'react-calendar';
 2 import Button from 'react-bootstrap/Button'
 3 import './Calendar.css'
 4 import { useState } from 'react';
 5 import { useEffect } from 'react';
 6 import dayjs from 'dayjs';
 7 import axios from 'axios'
 8
 9 const EditCalendar = () => {
10     const [dates, setDates] = useState({available:[],booked:[]})
11     const [tempDates, setTempDates] = useState({})
12     const [changeLog, setChangeLog] = useState({})
13
14     async function resetCalendar(){
15         async function getRegionData() { // Async to prevent stalling
16             let response = await axios.get('http://localhost:5000/
   booking/readforform');
17             return response;
18         }
19
20         // eslint-disable-next-line
21         let calendarData = (await getRegionData()).data;
22
23         let available = [];
24         let booked = [];
25
26         calendarData.forEach((instance) => {
27             if (instance.status === 'Available') {
28                 available.push(dayjs(instance.aptDate).format('DD/MM/
   YYYY'));
29
30             } else if (instance.status === 'Booked') {
31                 booked.push(dayjs(instance.aptDate).format('DD/MM/YYYY
   '));
32             }
33         });
34
35         return {available:available,booked:booked}
36     }
37
38
39     useEffect(async ()=>{
40         let res = await resetCalendar()
41         setTempDates(res.available);
42         setDates(res);
43     }, [])
44
45     async function handleSubmit(){
46         const token = localStorage.getItem('adminToken')
47         let verify;
48
49         if(!!token) {
50             verify = await axios.post('http://localhost:5000/login/
   verify', {adminToken:token});
51         } else {
52             verify =  false;
53         }
```

```
 54
 55         if(verify){
 56             let submitData = {
 57                 changeLog: changeLog,
 58                 adminToken: token,
 59             }
 60
 61             axios.post('http://localhost:5000/booking/editcalendar',
    submitData)
 62             setChangeLog({})
 63         } else {
 64             alert('Invalid authentication token')
 65         }
 66     }
 67
 68     return (
 69         <div className='ccontainer'>
 70             <Calendar
 71                 onChange={(e)=>{
 72                     let changeDate = dayjs(e).format('DD/MM/YYYY')
 73
 74                     if(dates.available.includes(changeDate)){ //
    Remove
 75
 76                         let Arr = dates.available
 77                         let newArray = Arr.filter(day=>day!==
    changeDate)
 78
 79                         setDates({...dates,available:newArray})
 80
 81                         if(tempDates.includes(changeDate)){
 82                             setChangeLog({...changeLog, [e]:false})
 83                         } else {
 84                             let newChanges = changeLog
 85                             delete newChanges[e]
 86                             setChangeLog(newChanges)
 87                         }
 88
 89                     } else { // Add
 90                         setDates({...dates,available:[...dates.
    available,changeDate]})
 91
 92                         if(tempDates.includes(changeDate)){
 93                             let newChanges = changeLog
 94                             delete newChanges[e]
 95                             setChangeLog(newChanges)
 96                         } else {
 97                             setChangeLog({...changeLog, [e]:true})
 98                         }
 99                     }
100                 }}
101
102                 tileClassName={({ date, view }) => {
103                     if (dates.available.includes(dayjs(date).format('
    DD/MM/YYYY'))) {
104                         return 'cavailable';
105
```

```
106                     } else if (dates.booked.includes(dayjs(date).
    format('DD/MM/YYYY'))) {
107                         return 'cbooked';
108                 }}}
109 
110                 tileDisabled={({ date, view }) => {
111                     if (date>new Date()) {
112                         return false;
113                     } else {
114                         return true;
115                     }
116                 }}
117 
118                 value=''
119                 showNeighboringMonth='false'
120             />
121             <Button
122             disabled={Object.keys(changeLog).length === 0}
123             onClick={(()=>{handleSubmit()})}>Save changes</Button>
124             <Button
125             disabled={Object.keys(changeLog).length === 0}
126             onClick={async ()=>{
127                 let res = await resetCalendar();
128                 setTempDates(res.available);
129                 setDates(res);
130             }
131 
132             }>Cancel</Button>
133         </div>
134     )
135 }
136 
137 export default EditCalendar
138 
```

```
<!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
```

```
 1 import express from 'express';
 2 import mongoose from 'mongoose';
 3 import cors from 'cors';
 4 import dotenv from 'dotenv';
 5
 6 dotenv.config();
 7
 8 const app = express();
 9
10 // Some middleware
11 app.use(express.urlencoded({ extended: true }));
12 app.use(express.json({ extended: true }));
13 app.use(cors());
14
15 // Routes
16 import bookingRouter from './API/routes/booking.js';
17 import counsRouter from './API/routes/couns.js';
18 import loginRouter from './API/routes/login.js'
19
20 app.use('/booking', bookingRouter);
21 app.use('/couns', counsRouter);
22 app.use('/login', loginRouter)
23 app.use('*', (req, res) => res.status(404).json({ error: 'Page not
   found' }));
24
25 // Connect to database
26 const url = process.env.ATLAS_URI;
27 mongoose.connect(url, { useNewUrlParser: true, useUnifiedTopology:
   true, 'useCreateIndex': true});
28 const connection = mongoose.connection;
29 connection.once('open', () => {console.log('Connected to MongoDB');});
30
31 // Start listening
32 const PORT = process.env.PORT || 5000;
33 app.listen(PORT, () => console.log(`Server running on port: ${PORT
   }`));
34
35 mongoose.set('useFindAndModify', false);
36
```

```
{
  "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 import mongoose from 'mongoose';
 2 mongoose.pluralize(null); // Prevent database from pluralizing
 3
 4 const schema = mongoose.Schema({
 5     aptDate: { type: Date, required: true, unique: true },
 6     status: { type: String, required: true },
 7     booking: {
 8         uniName: String,
 9         uniRepName: String,
10         uniRepJobTitle: String,
11         uniRepEmail: String,
12         uniRegion: String,
13         logoUrl: String,
14     },
15 });
16
17 const bookingDateInfo = mongoose.model('bookingDateInfo', schema);
18
19 export default bookingDateInfo;
20
```

```
 1 import mongoose from 'mongoose';
 2 mongoose.pluralize(null); // Prevent database from pluralizing
 3
 4 const schema = mongoose.Schema({
 5     name: String,
 6     counsEmail: {type:String, required:true, unique: true},
 7     receiveEmail: {type:Boolean ,required:true}
 8 });
 9
10 const counsellorEmail = mongoose.model('counsellorEmail', schema);
11
12 export default counsellorEmail;
13
```

```
 1 import express from 'express';
 2 import counsellorEmail from '../models/counsellorEmail.model.js';
 3 import jwt from 'jsonwebtoken'
 4 import dotenv from 'dotenv';
 5
 6 dotenv.config();
 7
 8 const router = express.Router();
 9
10 // Read
11 router.get('/read', (req, res) => {
12     counsellorEmail.find()
13         .then((counsellorEmail) => res.json(counsellorEmail))
14         .catch((err) => res.status(400).json('Error: ' + err));
15 });
16
17 router.patch('/edit', (req, res) => {
18     let token = req.body.adminToken
19
20     if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
21         counsellorEmail.findById(req.body.id)
22         .then((counsellorEmail) => {
23             counsellorEmail.name = req.body.name;
24             counsellorEmail.counsEmail = req.body.counsEmail;
25             counsellorEmail.receiveEmail = req.body.receiveEmail;
26
27
28             counsellorEmail.save()
29                 .then(() => res.json('Couns updated'))
30                 .catch((err) => res.status(400).json('Error1: ' + err
   ));
31         })
32         .catch((err) => res.status(400).json('Error2: ' + err));
33     } else {
34         res.status(403).json('Invalid Token')
35     }
36 });
37
38 // Create
39 router.post('/create', (req, res) => {
40     const name = req.body.name;
41     const counsEmail = req.body.counsEmail;
42     const receiveEmail = req.body.receiveEmail;
43     let token = req.body.adminToken
44
45     if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
46         const newCounsEmail = new counsellorEmail({
47             name,
48             counsEmail,
49             receiveEmail,
50         });
51
52         newCounsEmail.save()
53         .then(() => res.json('Counsellor email added'))
54         .catch((err) => res.status(400).json('Error: ' + err));
55     } else {
56         res.status(403).json('Invalid Token')
```

```
    }
});

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

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

export default router;

```

```
 1 import express from 'express';
 2 import dotenv from 'dotenv';
 3 import jwt from 'jsonwebtoken'
 4 import counsellorEmail from '../models/counsellorEmail.model.js';
 5
 6 dotenv.config();
 7
 8 const router = express.Router();
 9
10 router.post('/login', async (req,res) =>{
11     const password = req.body.password;
12     const email = req.body.email;
13
14     const exists = await counsellorEmail.exists({counsEmail:email});
15
16     if (password === process.env.ADMIN_PASSWORD && exists){
17         const token = jwt.sign({email:email},process.env.JWT_SECRET,{
   expiresIn: 86400 });
18         res.status(200).json({adminToken:token});
19     } else{
20         res.status(200).json('FALSE');
21     }
22 });
23
24 router.post('/verify', (req,res) =>{
25     const token = req.body.adminToken;
26     if(!!token){
27         const verified = jwt.verify(token, process.env.JWT_SECRET)
28         if (verified){
29             res.status(200).json(true);
30         } else {
31             res.status(200).json(false)
32         }
33     } else {
34         res.status(200).json(false)
35     }
36 });
37
38 export default router;
```

```
 1 import express, { query } from 'express';
 2 import bookingDateInfo from '../models/bookingDateInfo.model.js';
 3 import counsellorEmail from '../models/counsellorEmail.model.js';
 4 import nodemailer from 'nodemailer';
 5 import dotenv from 'dotenv';
 6 import axios from 'axios'
 7 import jwt from 'jsonwebtoken'
 8
 9 dotenv.config();
10
11 // Create nodemail transporter with gmail authorization
12 let transporter = nodemailer.createTransport({
13     service: 'gmail',
14     auth: {
15         type: 'OAUTH2',
16         user: 'bishvirtualsignup@gmail.com',
17         clientId: process.env.OAUTH_CLIENTID,
18         clientSecret: process.env.OAUTH_CLIENT_SECRET,
19         refreshToken: process.env.OAUTH_REFRESH_TOKEN,
20         accessToken: process.env.OAUTH_ACCESS_TOKEN
21     },
22     tls: {
23         rejectUnauthorized: false,
24     },
25 });
26
27 async function getLogo(uniName){
28     let query = (uniName.replace(/ /g,"+")) + "+Logo" // Create query
   string
29     let queryString =
30     `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}`
31     let res = await axios.get(queryString) // Get result
32     let imageURL = await res.data.images_results[0].thumbnail //
   Filter for first result
33     return await imageURL
34 }
35
36 const router = express.Router();
37
38 // Get data for calendar formatting
39 router.get('/readforform', (req, res) => {
40     bookingDateInfo.find(
41         { aptDate: { $gte: new Date() } },
42         { aptDate: 1, status: 1, 'booking.uniRegion': 1 })
43     .sort('aptDate')
44     .then((bookingDateInfo) => res.json(bookingDateInfo))
45     .catch((err) => res.status(400).json('Error: ' + err));
46 });
47
48 // Create a booking
49 router.post('/create', (req, res) => {
50     const aptDate = req.body.aptDate;
51     const status = req.body.status;
52     const uniName = req.body.booking.uniName;
53     const uniRepName = req.body.booking.uniRepName;
```

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

```
) => {
                        if (err) {
                            console.log(err);
                        } else {
                            console.log('Message sent');
                        }
                    });
                })
            })
        })
        .catch((err) => res.status(400).json('Error: ' + err));
    })
});


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

// Edit a booking
router.patch('/editbooking', (req, res) => {
    let allowMail = req.body.allowMail;
    let token = req.body.adminToken

    if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
        bookingDateInfo.findById(req.body.id)
        .then((bookingDateInfo) => {
            bookingDateInfo.booking.uniName = req.body.booking.
uniName;
            bookingDateInfo.booking.uniRepName = req.body.booking.
uniRepName;
            bookingDateInfo.booking.uniRepJobTitle = req.body.booking
.uniRepJobTitle;
            bookingDateInfo.booking.uniRepEmail = req.body.booking.
uniRepEmail;
            bookingDateInfo.booking.uniRegion = req.body.booking.
uniRegion;
            bookingDateInfo.booking.logoUrl = req.body.booking.
logoUrl;
            bookingDateInfo.aptDate = req.body.aptDate;

            bookingDateInfo.save()
            .then(() => {
                res.json('Booking updated')
                if(allowMail){
                    let mailOptions = {
                        from: 'BISH Signup <bishvirtualsignup@gmail.
com> ',
                        to: req.body.booking.uniRepEmail,
                        subject: 'Updated presentation information',
                        text: `Your booking information for a
```

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

```
201             })
202             .catch((err) => res.status(400).json('Error: ' + err));
203         })
204         .catch((err) => res.status(400).json('Error: ' + err));
205     } else {
206         res.status(403).json('Invalid Token')
207     }
208 });
209
210 // Remove a booking
211 router.delete('/remove', (req, res) => {
212     let token = req.body.adminToken
213
214     if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
215         bookingDateInfo.findByIdAndDelete(req.body.id)
216         .then(() => {res.json('Booking deleted');})
217         .catch((err) => {res.status(400).json('Error: ' + err)});
218     } else {
219         res.status(403).json('Invalid Token')
220     }
221 });
222
223 // Edit dates
224 router.post('/editcalendar', (req, res) =>{
225     let changeLog = req.body.changeLog
226     let token = req.body.adminToken
227
228     if(!!token && jwt.verify(token, process.env.JWT_SECRET)){
229         let changes =  []
230         Object.keys(changeLog).forEach((key) => {
231             if(changeLog[key]==true){
232                 changes.push({'insertOne':{"document":{aptDate:key,
    status:"Available"}}})
233             } else {
234                 changes.push({'deleteOne':{"filter":{aptDate:key}}})
235             }
236         });
237
238         bookingDateInfo.bulkWrite(changes)
239         .then(() => {
240             res.json('Dates updated')
241             counsellorEmail.find(
242                 { receiveEmail: true },
243                 { counsEmail: 1 }
244             )
245             .then((counsList)=>{
246                 Object.keys(changeLog).forEach((key) => {
247                     let text = '';
248                     if(changeLog[key]==true){
249                         text = `${text} \n ${key} - Added`
250                     } else {
251                         text = `${text} \n ${key} - Removed`
252                     }
253                 })
254
255                 counsList.forEach((couns)=>{
256                     let counsMailOptions = {
```

```
                        from: 'BISH Signup <bishvirtualsignup@gmail.com> ',
                        to: couns.counsEmail,
                        subject: 'Changes to available dates',
                        text: `Updates have been made to the list of available dates:
                        ${text}`,
                    }

                    transporter.sendMail(counsMailOptions, (err, info) => {
                        if (err) {
                            console.log(err);
                        } else {
                            console.log('Message sent');
                        }
                    });
                })
            })
        })
        .catch((err) => {res.status(400).json('Error: ' + err)})
    } else {
        res.status(403).json('Invalid Token')
    }
});


// Manually create a booking for insomnia
router.post('/createm', (req, res) => {
    const aptDate = req.body.aptDate;
    const status = req.body.status;
    const uniName = req.body.booking.uniName;
    const uniRepName = req.body.booking.uniRepName;
    const uniRepJobTitle = req.body.booking.uniRepJobTitle;
    const uniRepEmail = req.body.booking.uniRepEmail;
    const uniRegion = req.body.booking.uniRegion;

    getLogo(uniName).then( logoUrl => {
        const newBooking = new bookingDateInfo({
            aptDate: aptDate,
            status: status,
            booking: {
                uniName,
                uniRepName,
                uniRepJobTitle,
                uniRepEmail,
                uniRegion,
                logoUrl,
            },
        });

        newBooking.save()
        .then(() => res.json('Booking created'))
        .catch((err) => res.status(400).json('Error: ' + err));
    })
});

```

```
311 export default router;
312
```

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

const createWindow = () => {
    const win = new BrowserWindow({
      width: 800,
      height: 600
    })

    win.loadURL('http://localhost:3000')
  }

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
```

```
42 ## Learn More
43
44 You can learn more in the [Create React App documentation](https://
   facebook.github.io/create-react-app/docs/getting-started).
45
46 To learn React, check out the [React documentation](https://reactjs.
   org/).
47
48 ### Code Splitting
49
50 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)
51
52 ### Analyzing the Bundle Size
53
54 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)
55
56 ### Making a Progressive Web App
57
58 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)
59
60 ### Advanced Configuration
61
62 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)
63
64 ### Deployment
65
66 This section has moved here: [https://facebook.github.io/create-react-
   app/docs/deployment](https://facebook.github.io/create-react-app/docs/
   deployment)
67
68 ### `npm run build` fails to minify
69
70 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)
71
```

```
# 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
```

```
 1 import {useState} from "react";
 2 import dayjs from "dayjs";
 3 import Card from 'react-bootstrap/Card'
 4 import image from './undraw_Page_not_found_re_e9o6.png'
 5 import '../admin.css'
 6
 7
 8 const BCard = ({instance}) => {
 9   const [date, setDate] = useState(instance.aptDate);
10   const [bookingData, setBookingData] = useState({
11         uniName: instance.booking.uniName,
12         uniRegion: instance.booking.uniRegion,
13         uniRepJobTitle: instance.booking.uniRepJobTitle,
14         uniRepName: instance.booking.uniRepName,
15         uniRepEmail: instance.booking.uniRepEmail,
16         logoUrl:instance.booking.logoUrl,
17     });
18
19   return (
20       <>
21       <div className='col-4 pad'>
22         <Card style={{ width: '100%' }}>
23             <Card.Body>
24                 <Card.Title className='cardUni'>{bookingData.uniName
   }</Card.Title>
25                 <Card.Subtitle className='cardRegion'>{bookingData.
   uniRegion}</Card.Subtitle>
26             </Card.Body>
27             <Card.Img variant="top" src={!!bookingData.logoUrl?
   bookingData.logoUrl:image} className='cardImage'/>
28             <Card.Body>
29                 <Card.Text className='cardDate'>{dayjs(date).format('
   dddd, MMMM D')}</Card.Text>
30                 <Card.Text className='cardTitle'>{bookingData.
   uniRepJobTitle + ': ' + bookingData.uniRepName}</Card.Text>
31                 <Card.Text className='cardEmail'>{bookingData.
   uniRepEmail}</Card.Text>
32             </Card.Body>
33         </Card>
34       </div>
35       </>
36   )
37 }
38
39 export default BCard
```

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

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

```
 1 <!DOCTYPE html>
 2 <html lang="en">
 3   <head>
 4     <meta charset="utf-8" />
 5     <link
 6   rel="stylesheet"
 7   href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.
   min.css"
 8   integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
   iJTQUOhcWr7x9JvoRxT2MZw1T"
 9   crossorigin="anonymous"
10 />
11     <title>React App</title>
12   </head>
13   <body>
14     <noscript>You need to enable JavaScript to run this app.</noscript
   >
15     <div id="root"></div>
16     <script src="./build/app.js"></script>
17   </body>
18 </html>
19
```

```
{
  "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"
}

```