

```
1 {
2   "name": "react-fe-counsweb",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.14.1",
7     "@testing-library/react": "^11.2.7",
8     "@testing-library/user-event": "^12.8.3",
9     "axios": "^0.21.4",
10    "bootstrap": "^4.6.0",
11    "dayjs": "^1.10.6",
12    "react": "^17.0.2",
13    "react-bootstrap": "^1.6.1",
14    "react-calendar": "^3.4.0",
15    "react-dom": "^17.0.2",
16    "react-icons": "^4.2.0",
17    "react-router-dom": "^5.2.0",
18    "react-scripts": "4.0.3",
19    "web-vitals": "^1.1.2"
20  },
21  "scripts": {
22    "start": "react-scripts start",
23    "build": "react-scripts build",
24    "test": "react-scripts test",
25    "eject": "react-scripts eject"
26  },
27  "eslintConfig": {
28    "extends": [
29      "react-app",
30      "react-app/jest"
31    ]
32  },
33  "browserslist": {
34    "production": [
35      ">0.2%",
36      "not dead",
37      "not op_mini all"
38    ],
39    "development": [
40      "last 1 chrome version",
41      "last 1 firefox version",
42      "last 1 safari version"
43    ]
44  }
45 }
46
```

```
1 import React, { useState } from 'react';
2 import 'bootstrap/dist/css/bootstrap.min.css';
3 import FormPage from './components/FormPage/FormPage';
4 import EditBookings from './components/AdminPage/bookings/EditBookings';
5 import EditCalendar from './components/AdminPage/calendar/EditCalendar';
6 import EditEmail from './components/AdminPage/emails/EditEmail';
7 import Support from './components/Support';
8 import { BrowserRouter, Route, Switch } from 'react-router-dom';
9 import Sidebar2 from './components/Sidebar2';
10 import PrivateRoute from './PrivateRoute';
11
12 function App() {
13     const [admin, setAdmin] = useState(false)
14     return (
15         <>
16             <BrowserRouter>
17                 <Sidebar2 admin={admin} setAdmin={setAdmin}/>
18                 <Switch>
19                     <Route path='/' exact component={FormPage} />
20                     <Route path='/support' exact component={Support}>
21                 />
22                     <PrivateRoute admin={admin} path='/admin/booking' exact component={EditBookings} />
23                     <PrivateRoute admin={admin} path='/admin/calendar' exact component={EditCalendar} />
24                     <PrivateRoute admin={admin} path='/admin/emails' exact component={EditEmail} />
25                 </Switch>
26             </BrowserRouter>
27         </>
28     );
29 }
30
31 export default App;
32
```

```
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
```

```
1 .formAlign {
2     padding: 2em 1em 2em 1em;
3     width: 40rem;
4     max-width: 100%;
5     margin: auto;
6 }
7
8 .calendarAlign {
9     padding: 1em 1em 3em 1em;
10    width: 38rem;
11    max-width: 100%;
12    margin: auto;
13 }
14
15 .timeBox {
16     text-align: center;
17     position: relative;
18     width: 38rem;
19     max-width: 100%;
20     padding: 1rem;
21     margin: auto;
22
23     border: 0.3rem solid #ececfc;
24     border-radius: 8px;
25     color: #212529;
26 }
27
28 .submitAlign {
29     text-align: center;
30     position: relative;
31     width: 38rem;
32     max-width: 100%;
33     padding: 1rem;
34     margin: auto;
35 }
36
37 .timeTitle {
38     /* text-decoration: underline; */
39     font-size: 1.5em;
40     font-weight: normal;
41     text-decoration-thickness: 0.1rem;
42 }
43
44 .localTime {
45     padding-top: 0.5rem;
46 }
47
48 .localTimeTitle {
49     text-decoration: underline;
50     font-size: 1.3em;
51     font-weight: lighter;
52     text-decoration-thickness: 0.1rem;
53 }
54
55 .ourTime {
56     padding-top: 0.5rem;
57 }
```

```
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
```

```
115 .booked:hover {  
116     color: hsla(3, 100%, 20%, 1) !important;  
117     background-color: rgba(255, 65, 54, 0.55) !important;  
118 }  
119  
120 .booked {  
121     color: hsla(3, 100%, 20%, 1) !important;  
122     background-color: rgba(255, 65, 54, 0.6) !important;  
123 }  
124  
125 .regionBooked {  
126     color: hsla(350, 65%, 15%) !important;  
127     background-color: rgba(194, 43, 65, 0.6) !important;  
128 }  
129  
130 .regionBooked:hover {  
131     color: hsla(350, 65%, 15%) !important;  
132     background-color: rgba(194, 43, 65, 0.55) !important;  
133 }  
134  
135 .available {  
136     color: hsla(127, 63%, 15%, 1) !important;  
137     background-color: rgba(1, 255, 112, 0.7) !important;  
138 }  
139  
140 .available:hover {  
141     color: hsla(127, 63%, 15%, 1) !important;  
142     background-color: rgba(1, 255, 112, 0.5) !important;  
143 }  
144  
145 .selected {  
146     color: #111111 !important;  
147     background-color: rgba(13, 110, 253 0.85) !important;  
148 }  
149  
150 .selected:hover {  
151     color: #111111 !important;  
152     background-color: rgba(13, 110, 253 0.8) !important;  
153 }  
154  
155 .invalidCalendar {  
156     transition: border-color 0.15s ease-in-out, box-shadow 0.15s ease  
     -in-out;  
157     border-radius: 0.25rem;  
158     border: 1.5px solid #dc3545;  
159     padding: 0.5em;  
160 }  
161  
162 .invalidText {  
163     width: 100%;  
164     margin-top: 0.3rem;  
165     margin-left: 0;  
166     margin-right: 0;  
167     font-size: 0.875em;  
168     color: #dc3545;  
169     display: block;  
170     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
21                 setShow(false)
22             }
23         })
24         .catch(err =>{
25             console.log(err)
26         });
27     }
28
29     return (
30         <Modal show={show} onHide={cancel}>
31             <Modal.Header closeButton>
32                 <Modal.Title>Login</Modal.Title>
33             </Modal.Header>
34             <Modal.Body>
35                 <Form>
36                     <Form.Group>
37                         <Form.Label> Email </Form.Label>
38                         <Form.Control type='text' onChange={(e) => {
39                             setLoginfo({...loginfo,email:e.target.value})}}/>
40                     </Form.Group>
41                     <Form.Group>
42                         <Form.Label> Password </Form.Label>
43                         <Form.Control type='password' onChange={(e) => {
44                             setLoginfo({...loginfo,password:e.target.value})}}/>
45                     </Form.Group>
46                     {error?
47                         <div className="incorrect">
48                             Incorrect Password/Email
49                         </div>
50                     :<div></div>
51                 }
52             </Form>
53             <Modal.Footer>
54                 <Button onClick={()=>{handleSubmit()}}>Submit</Button>
55             </Modal.Footer>
56         </Modal>
57     )
58 }

```

```
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); // Separate 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}>/>
34 Admin</span>} className="nav-text">
35                     <NavDropdown.Item href="#edit">
36                         <Nav.Link as={NavLink} to='/admin/booking' className='nopad'>
37                             <span><FaCalendarPlus size={20}>/> Bookings
38                         </Nav.Link>
39                     </NavDropdown.Item>
40
41                     <NavDropdown.Item href="#dates">
42                         <Nav.Link as={NavLink} to='/admin/calendar' className='nopad'>
43                             <span><FaCalendarAlt size={20}>/> Calendar
44                         </Nav.Link>
45                     </NavDropdown.Item>
46
47                     <NavDropdown.Item href="#email">
48                         <Nav.Link as={NavLink} to='/admin/emails' className='nopad'>
49                     </Nav.Link>
50                 </NavDropdown.Item>
51             )
52         }
53     }
54
55     return(
56         <div>
57             <Navbar>
58                 <Navbar.Brand>React App</Navbar.Brand>
59                 <Navbar.Toggle>
60                 <Navbar.Collapse>
61                     <Nav>
62                         <Nav.Link href="#">Home</Nav.Link>
63                         <Nav.Link href="#">About</Nav.Link>
64                         <Nav.Link href="#">Contact</Nav.Link>
65                         <Nav.Link href="#">Logout</Nav.Link>
66                     </Nav>
67                     <NavDropdown title="Admin" show={show} onToggle={toggleSidebar}>
68                         {isAdmin()}
69                     </NavDropdown>
70                 </Navbar.Collapse>
71             </Navbar>
72             <div>
73                 <h1>Welcome to Admin Panel</h1>
74                 <p>This is the Admin Panel for the application. You can manage bookings, calendar, and emails here.</p>
75             </div>
76         </div>
77     )
78 }
79
80 export default Sidebar2;
```

```

48                      <span><MdEmail size={20}>/> Emails</span>
49                  </Nav.Link>
50              </NavDropdown.Item>
51          </NavDropdown>
52      )
53  } else {
54      return(
55          <Nav.Link onClick={()=>{setShow(true);setError(false)}}>
56              <span><AiFillLock size={22}>/> Admin</span>
57          </Nav.Link>
58      )
59  }
60 }
61
62 function cancel(){
63     setShow(false)
64 }
65
66 return (
67     <>
68     <div className='wCube'></div>
69     <Login show={show} cancel={cancel} setShow={setShow} error={error} setError={setError} setAdmin={setAdmin}/>
70     <div>
71         <Navbar bg="Light">
72             <Navbar.Brand className='bars' as={Link} onClick={()=>{toggleSidebar()}}>
73                 <div className='burgerpad'>
74                     <div className={!sidebar?'burger':'burger-close'}></div>
75                 </div>
76             </Navbar.Brand>
77             <Nav className={sidebar?"me-auto sidebar-active":"me-auto sidebar-inactive"}>
78                 <Nav.Link as={NavLink} to='/' className="nav-text">
79                     <span className='nav-icon'><BsPencilSquare size={22}>/> Form</span>
80                 </Nav.Link>
81                 {isAdmin()}
82                 <Nav.Link as={NavLink} to='/support' className="nav-text">
83                     <span><AiOutlineQuestionCircle size={22}>/> Help</span>
84                 </Nav.Link>
85             </Nav>
86         </Navbar>
87     </div>
88     </>
89 )
90 }
91
92 export default Sidebar2
93

```

```
1 .nopad{  
2     padding: 0 0 0 0 !important;  
3 }  
4  
5 .wCube{  
6     position: absolute;  
7     top:0;  
8     left:0;  
9     height:64px;  
10    width:64px;  
11    background-color: #f8f9fa;  
12    z-index: 1;  
13 }  
14  
15 .bars{  
16     z-index: 2;  
17     margin-right: 0;  
18 }  
19  
20 .sidebar-active{  
21     left: 0rem;  
22     position: relative;  
23     transition: 500ms;  
24 }  
25  
26 .sidebar-inactive{  
27     left: -20rem;  
28     position: absolute;  
29     transition: 500ms;  
30 }  
31  
32 .burger{  
33     width:40px;  
34     height: 6px;  
35     background: rgba(0,0,0,0.8);  
36     border-radius: 3px;  
37     box-shadow: 0 2px 2px rgba(255,101,47,.2);  
38     transition: all .5s ease-in-out;  
39 }  
40  
41 .burger::before,.burger::after{  
42     content:'';  
43     position: absolute;  
44     width:40px;  
45     height: 6px;  
46     background: rgba(0,0,0,0.8);  
47     border-radius: 3px;  
48     box-shadow: 0 2px 2px rgba(255,101,47,.2);  
49     transition: all .5s ease-in-out;  
50 }  
51  
52 .burger::before {  
53     transform: translateY(-12px);  
54 }  
55  
56 .burger::after {  
57     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         };

```



```

101                     .subtract(7, 'day')
102                     .format('DD/MM/YYYY');
103
104             if (index - 1 >= 0) {
105                 // Checks last available day
106                 if (calendarData[index - 1].status === 'Available' && dayjs(calendarData[index - 1])
107                     .isBetween(lowB, uppB, null, '[]')) {
108                     regionBooked.push(dayjs(calendarData[
109                         index - 1].aptDate).format('DD/MM/YYYY'));
110                     calendarData[index - 1].status = 'Region
111                     booked'; // Prevent further formatting
112                     }
113             }
114
115             if (index + 1 <= calendarData.length - 1) {
116                 // Checks next available day
117                 if (calendarData[index + 1].status === 'Available' && dayjs(calendarData[index + 1])
118                     .isBetween(lowB, uppB, null, '[]')) {
119                     regionBooked.push(dayjs(calendarData[
120                         index + 1].aptDate).format('DD/MM/YYYY'));
121                     calendarData[index + 1].status = 'Region
122                     booked';
123                     }
124             }
125
126             calendarData.forEach((instance) => {
127                 if (instance.status === 'Available') {
128                     available.push(dayjs(instance.aptDate).format('DD
129                     /MM/YYYY'));
130
131                 } else if (instance.status === 'Booked') {
132                     booked.push(dayjs(instance.aptDate).format('DD/MM
133                     /YYYY'));
134
135                 }
136             });
137
138             // Deselect current date if it became booked/region
139             // booked during selection
140             if (booked.includes(dayjs(dateData).format('DD/MM/YYYY'))
141                 || regionBooked.includes(dayjs(dateData).format('DD/MM/YYYY'))) {
142                 setDateData(new Date());
143
144             }
145
146             setKey(!key); // Manually re-render
147
148         },
149         [formData.uniRegion, checkDate]); // Dependencies which, after
150         // change, will trigger the useEffect
151
152
153     return (
154         <div className='container'>
155             <div className='row align-items-center'>
156                 <div className='col-6'>

```

```

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

```

```

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 controlId='form.
15                     University'>
16                     <Form.Label>University</Form.Label>
17                     <Form.Control
18                         required
19                         readOnly={isReadOnly ? true : false}
20                         as='input'
21                         type='text'
22                         defaultValue={formData.uniName}
23                         onChange={(e) => {
24                             setFormData({...formData, uniName: e.target
25                             .value,});
26                             delete errors.uniRepName;
27                         }
28                     }>
29                     <Form.Control.Feedback type='invalid'>
30                         {errors.uniName}
31                     </Form.Control.Feedback>
32                 </Form.Group>
33
34                 <Form.Group controlId='form.FullName'>
35                     <Form.Label>Full name</Form.Label>
36                     <Form.Control
37                         required
38                         readOnly={isReadOnly ? true : false}
39                         as='input'
40                         type='text'
41                         defaultValue={formData.uniRepName}
42                         onChange={(e) => {
43                             setFormData({...formData, uniRepName: e.
44                             target.value,});
45                             delete errors.uniRepName;
46                         }
47                     }>
48                     <Form.Control.Feedback type='invalid'>
49                         {errors.uniRepName}
50                     </Form.Control.Feedback>
51                 </Form.Group>
52
53                 <Form.Group controlId='form.JobTitle'>
54                     <Form.Label>Job title</Form.Label>
55                     <Form.Control

```

```

53                     required
54                     isValid={false}
55                     readOnly={isReadOnly ? true : false}
56                     as='input'
57                     type='text'
58                     defaultValue={formData.uniRepJobTitle}
59                     onChange={(e) => {
60                         setFormData({...formData, uniRepJobTitle:
61                         e.target.value,});
62                         delete errors.uniRepJobTitle;
63                     }
64                     />
65                     <Form.Control.Feedback type='invalid'>
66                         {errors.uniRepJobTitle}
67                     </Form.Control.Feedback>
68                     </Form.Group>
69
70                     <Form.Group className='mb-3' controlId='form.
71                     EmailAddress'>
72                         <Form.Label>Email address</Form.Label>
73                         <Form.Control
74                             required
75                             readOnly={isReadOnly ? true : false}
76                             as='input'
77                             type='email'
78                             defaultValue={formData.uniRepEmail}
79                             onChange={(e) => {
80                                 setFormData({...formData, uniRepEmail: e.
81                                 target.value,});
82                                 delete errors.uniRepEmail;
83                             }
84                         />
85                         <Form.Control.Feedback type='invalid'>
86                             {errors.uniRepEmail}
87                         </Form.Control.Feedback>
88                     </Form.Group>
89
90                     <Form.Group className='mb-3' controlId='form.Region'>
91                         <Form.Label>Region</Form.Label>
92                         <Form.Control
93                             required
94                             disabled={isReadOnly ? true : false}
95                             as='select'
96                             defaultValue={formData.uniRegion}
97                             onChange={(e) =>
98                                 setFormData({...formData, uniRegion: e.
99                                 target.value,})
100                             }
101                             <option hidden={formData.uniRegion !== '' ?
102                             true : false} selected='selected'></option>
103                             <option value='USA'>USA</option>
104                             <option value='CANADA'>Canada</option>
105                             <option value='EUROPE'>Europe</option>
106                             <option value='UNITED KINGDOM'>United Kingdom
107                             </option>
108                             <option value='OTHER'>Other</option>
109                         </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
```

```
1 abbr[title] {
2     border-bottom: none !important;
3     cursor: inherit !important;
4     text-decoration: none !important;
5 }
6 .react-calendar {
7     width: 38rem;
8     max-width: 100%;
9     background: white;
10    /* border: 1px solid #a0a096; */
11    font-family: 'system-ui';
12    line-height: 1.125em;
13 }
14 .react-calendar--doubleView {
15     width: 700px;
16 }
17 .react-calendar--doubleView .react-calendar__viewContainer {
18     display: flex;
19     margin: -0.5em;
20 }
21 .react-calendar--doubleView .react-calendar__viewContainer > * {
22     width: 50%;
23     margin: 0.5em;
24 }
25 .react-calendar,
26 .react-calendar *,
27 .react-calendar *:before,
28 .react-calendar *:after {
29     -moz-box-sizing: border-box;
30     -webkit-box-sizing: border-box;
31     box-sizing: border-box;
32 }
33 .react-calendar button {
34     margin: 0;
35     border: 0;
36     outline: none;
37 }
38 .react-calendar button:enabled:hover {
39     cursor: pointer;
40 }
41 .react-calendar__navigation {
42     height: 44px;
43     margin-bottom: 1em;
44 }
45 .react-calendar__navigation button {
46     min-width: 44px;
47     background: none;
48 }
49 .react-calendar__navigation button:enabled:hover,
50 .react-calendar__navigation button:enabled:active {
51     background-color: #e6e6e6;
52 }
53 /* .react-calendar__navigation button[disabled] {
54     background-color: #f0f0f0; */
55 /*} */
56 .react-calendar__month-view__weekdays {
57     text-align: center;
```

```
58     text-transform: uppercase;
59     font-weight: bold;
60     font-size: 0.75em;
61 }
62 .react-calendar__month-view__weekdays__weekday {
63     padding: 0.5em;
64 }
65 .react-calendar__month-view__weekNumbers {
66     font-weight: bold;
67 }
68 .react-calendar__month-view__weekNumbers .react-calendar__tile {
69     display: flex;
70     align-items: center;
71     justify-content: center;
72     font-size: 0.75em;
73     padding: calc(0.75em / 0.75) calc(0.5em / 0.75);
74 }
75 .react-calendar__month-view__days__day--weekend {
76     color: #d10000;
77 }
78 .react-calendar__month-view__days__day--neighboringMonth {
79     color: #757575;
80 }
81 .react-calendar__year-view .react-calendar__tile,
82 .react-calendar__decade-view .react-calendar__tile,
83 .react-calendar__century-view .react-calendar__tile {
84     padding: 2em 0.5em;
85 }
86 .react-calendar__tile {
87     max-width: 100%;
88     text-align: center;
89     padding: 0.75em 0.5em;
90     background: none;
91 }
92 .react-calendar__tile:disabled {
93     background-color: #f0f0f0;
94 }
95 .react-calendar__tile:enabled:hover,
96 .react-calendar__tile:enabled:focus {
97     background-color: #e6e6e6;
98 }
99 .react-calendar__tile--now {
100     background: #ffff76;
101 }
102 .react-calendar__tile--now:enabled:hover,
103 .react-calendar__tile--now:enabled:focus {
104     background: #ffffa9;
105 }
106 .react-calendar__tile--hasActive {
107     background: #76baff;
108 }
109 .react-calendar__tile--hasActive:enabled:hover,
110 .react-calendar__tile--hasActive:enabled:focus {
111     background: #a9d4ff;
112 }
113 .react-calendar__tile--active {
114     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
```

```
1 import React from 'react';
2 import dayjs from 'dayjs';
3 import utc from 'dayjs/plugin/utc';
4 import isToday from 'dayjs/plugin/isToday';
5 import advancedFormat from 'dayjs/plugin/advancedFormat';
6 import timezone from 'dayjs/plugin/timezone';
7
8 dayjs.extend(isToday);
9 dayjs.extend(utc);
10 dayjs.extend(timezone);
11 dayjs.extend(advancedFormat);
12
13 const TimeDisplay = ({ date }) => {
14     let aptDate = dayjs(date).utc().hour(13).minute(30).second(0); // Sets date to predetermined appointment time
15     let visible = dayjs(date).format('DD/MM/YYYY') === dayjs(new Date()).format('DD/MM/YYYY'); // Checks if a date has been selected
16
17     return (
18         <div className='timeBox'>
19             <h1 className='timeTitle'>Your appointment is at</h1>
20             <div className='localTime'>
21                 <h1 className='localTimeTitle'>Local time</h1>
22                 {visible ? dayjs(aptDate).local().format('h:mm A') : dayjs(aptDate).local().format('MMMM D, h:mm A')}
23                 <br />
24                 {'GMT' + dayjs(aptDate).local().format('z (z)')}
25             </div>
26             <br />
27
28             <div className='ourTime'>
29                 <h1 className='ourTimeTitle'>Our time</h1>
30                 {visible ? dayjs(aptDate).tz('America/Chicago').format('h:mm A') : dayjs(aptDate).tz('America/Chicago').format('MMMM D, h:mm A')}
31                 <br />
32                 {'GMT' + dayjs(aptDate).tz('America/Chicago').format('z (z)')}
33             </div>
34         </div>
35     );
36 };
37
38 export default TimeDisplay;
39
```

```

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')
51             ') !== dayjs(selectedDate).format('DD/MM/YYYY')) {
52                 return true;
53             } else {
54                 if (available.includes(dayjs(date).format
55 ('DD/MM/YYYY')))) {
56                     return false;
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
```

```
1 .cardUni{  
2     font-size: 1.5rem;  
3     font-weight: 300;  
4 }  
5  
6 .cardRegion{  
7     font-size: 1rem;  
8     font-weight: 500;  
9     color: #6c757d;  
10 }  
11  
12 .cardImage{  
13     width: calc(100% + 2px);  
14     border-radius: 0;  
15     margin-left: -1px;  
16     border-left: 1px solid #DFDFDF;  
17     border-right: 1px solid #DFDFDF;  
18 }  
19  
20 .cardDate{  
21 }  
22 }  
23  
24 .cardTitle{  
25 }  
26 }  
27  
28 .cardEmail{  
29 }  
30 }  
31  
32 .pad{  
33     margin-top: 1rem;  
34     margin-bottom: 1rem;  
35 }  
36  
37 .cardButton{  
38     width: 100%;  
39 }  
40  
41 .login-container{  
42     display: flex;  
43     justify-content: center;  
44     align-items: center;  
45 }  
46  
47 .ccontainer {  
48     margin-left: calc(50vw - 20rem);  
49     margin-top: calc(50vh - 15rem);  
50 }  
51  
52 .cbooked {  
53     color: hsla(3, 100%, 20%, 1) !important;  
54     background-color: rgba(255, 65, 54, 0.6) !important;  
55 }  
56  
57 .cbooked:hover {
```

```
58     color: hsla(3, 100%, 20%, 1) !important;
59     background-color: rgba(255, 65, 54, 0.55) !important;
60 }
61
62 .cavailable {
63     color: hsla(127, 63%, 15%, 1) !important;
64     background-color: rgba(1, 255, 112, 0.7) !important;
65 }
66
67 .cavailable:hover {
68     color: hsla(127, 63%, 15%, 1) !important;
69     background-color: rgba(1, 255, 112, 0.5) !important;
70 }
```

```
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
53 /verify', {adminToken:token});
54             } else {
55                 verify = false;
56             }
57         }
58     }
59
60     return (
61         <Card>
62             <Card.Body>
63                 <Form>
64                     <Form.Group controlId="formBasicEmail">
65                         <Form.Label>Email address</Form.Label>
66                         <Form.Control type="email" value={counsInfo.receiveEmail} />
67                     </Form.Group>
68                     <Form.Group controlId="formBasicPassword">
69                         <Form.Label>Password</Form.Label>
70                         <Form.Control type="password" value={counsInfo.receiveEmail} />
71                     </Form.Group>
72                     <Form.Group controlId="formBasicName">
73                         <Form.Label>Name</Form.Label>
74                         <Form.Control type="text" value={counsInfo.name} />
75                     </Form.Group>
76                     <Form.Group controlId="formBasicEmail2">
77                         <Form.Label>Couns Email</Form.Label>
78                         <Form.Control type="text" value={counsInfo.counsEmail} />
79                     </Form.Group>
80                     <Form.Group controlId="formBasicEmail3">
81                         <Form.Label>Receive Email</Form.Label>
82                         <Form.Control type="text" value={counsInfo.receiveEmail} />
83                     </Form.Group>
84                     <Form.Group controlId="formBasicEmail4">
85                         <Form.Label>Verify</Form.Label>
86                         <Form.Control type="text" value={verify} />
87                     </Form.Group>
88                     <Form.Group controlId="formBasicEmail5">
89                         <Form.Label>Error</Form.Label>
90                         <Form.Control type="text" value={error} />
91                     </Form.Group>
92                     <Form.Group controlId="formBasicEmail6">
93                         <Form.Label>Validated</Form.Label>
94                         <Form.Control type="text" value={validated} />
95                     </Form.Group>
96                     <Form.Group controlId="formBasicEmail7">
97                         <Form.Label>Show</Form.Label>
98                         <Form.Control type="text" value={show} />
99                     </Form.Group>
100                    <Form.Group controlId="formBasicEmail8">
101                        <Form.Label>Cancel</Form.Label>
102                        <Form.Control type="button" value="Cancel" onClick={cancel} />
103                    </Form.Group>
104                    <Form.Group controlId="formBasicEmail9">
105                        <Form.Label>Handle Open</Form.Label>
106                        <Form.Control type="button" value="Handle Open" onClick={handleOpen} />
107                    </Form.Group>
108                    <Form.Group controlId="formBasicEmail10">
109                        <Form.Label>Edit</Form.Label>
110                        <Form.Control type="button" value="Edit" onClick={edit} />
111                    </Form.Group>
112                </Form>
113            </Card.Body>
114        </Card>
115    )
116  }
117
```

```

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',
64             counsInfo);
65             setShow(false);
66
67         } else{
68             alert('Invalid authentication token')
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/
79             verify', {adminToken:token});
80         } else {
81             verify = false;
82         }
83
84         if(verify){
85             axios.delete('http://localhost:5000/couns/remove', {data
86             : {id:instance._id, adminToken:token}})
87             .then(()=>{
88                 // eslint-disable-next-line
89                 getEmail()
90                 .then((res)=>{
91                     setEmails(res.data);
92                 })
93             }).then(setShow(false));
94         } else{
95             alert('Invalid authentication token')
96         }
97
98     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             >
117                 <Form.Control
118                     type='text'
119                     defaultValue={tempCounsInfo.name}
120                     onChange={(e) => {
121                         setCounsInfo({...counsInfo, name:
122                             e.target.value,})
123                     }}
124                     isValid={validated?!error.name:
125                         false}
126                     isInvalid={validated?!!error.name:
127                         false}
128             >
129             </Form.Group>
130             <Form.Group>
131                 <Form.Label> Email </Form.Label>
132                 <Form.Control
133                     type='text'
134                     defaultValue={tempCounsInfo.
135                         counsEmail}
136                     onChange={(e) => {
137                         setCounsInfo({...counsInfo,
138                             counsEmail: e.target.value,})
139                     }}
140                     isInvalid={validated?!!error.
141                         counsEmail:false}
142                     isValid={validated?!error.counsEmail:
143                         false}
144             >
145             </Form.Group>
146             <Form.Group>
147                 <Form.Label> Receive Emails? </Form.Label>
148             >
149                 <Form.Control
150                     type='checkbox'
151                     defaultChecked={tempCounsInfo.
152                         receiveEmail}
153                     onChange={(e)=> {
154                         setCounsInfo({...counsInfo,
155                             receiveEmail: e.target.value,})
156                     }}
157                     isInvalid={validated?!!error.
158                         receiveEmail:false}
159                     isValid={validated?!error.
160                         receiveEmail:false}
161             >
162             </Form.Group>
163         </Form>
164     </Modal.Body>
165     <Modal.Footer>
166         <Button variant="danger" onClick={()=>{remove
167             ()}} style={{'margin-right':'auto'}}>
168             Delete Booking

```

```
155                     </Button>
156                     <Button variant="primary" onClick={()=>{edit
157                         ()}} >
158                         Save Changes
159                     </Button>
160                 </Modal.Footer>
161             </Modal>
162         <div className='col-6 pad'>
163             <Card style={{ width: '100%' }}>
164                 <Card.Body>
165                     <Card.Title className='cardUni'>{counsInfo.
166                         name}</Card.Title>
166                     <Card.Text className='cardEmail'>{counsInfo.
167                         counsEmail}</Card.Text>
167                     <Card.Text className='cardTitle'>{'Receive
168                         Emails?: '+counsInfo.receiveEmail}</Card.Text>
168                     <Button variant="primary" className="
169                         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
53 /verify', {adminToken:token});
54             } else {
55                 verify = false;
56             }
57         }
58     }
59
60     function handleCancel(e) {
61         e.preventDefault();
62         setShow(false);
63     }
64
65     function handleOk(e) {
66         e.preventDefault();
67         if(show) {
68             handleOpen();
69         } else {
70             handleCancel();
71         }
72     }
73
74     return (
75         <Modal show={show} onHide={cancel}>
76             <Modal.Header>
77                 <Modal.Title>Add Counsellor</Modal.Title>
78             </Modal.Header>
79             <Modal.Body>
80                 <Form>
81                     <Form.Group controlId="formGroupCounsName">
82                         <Form.Label>Name</Form.Label>
83                         <Form.Control type="text" value={counsInfo.name} />
84                     </Form.Group>
85                     <Form.Group controlId="formGroupCounsEmail">
86                         <Form.Label>Email</Form.Label>
87                         <Form.Control type="text" value={counsInfo.counsEmail} />
88                     </Form.Group>
89                     <Form.Group controlId="formGroupReceiveEmail">
90                         <Form.Check type="checkbox" checked={counsInfo.receiveEmail} />
91                         <Form.Label>Receive Email</Form.Label>
92                     </Form.Group>
93                 </Form>
94             </Modal.Body>
95             <Modal.Footer>
96                 <Button variant="primary" onClick={handleOk}>Add</Button>
97                 <Button variant="secondary" onClick={handleCancel}>Cancel</Button>
98             </Modal.Footer>
99         </Modal>
100    );
101}
102
```

```

57         if(verify){
58             setValidated(false);
59
60             counsInfo.adminToken = token
61
62             axios.post('http://localhost:5000/couns/create',
63             counsInfo)
64                 .then(()=>{
65                     getEmail()
66                     .then((res)=>{
67                         setEmails(res.data)
68                     })
69                 }).catch((err)=>{
70                     console.log(err)
71                     alert(err)
72                 })
73             setShow(false);
74         } else{
75             alert('Invalid authentication token')
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:
93                                     e.target.value,})
94                             }}
95                             isValid={validated?!error.name:
96                             false}
97                         />
98                         <Form.Group>
99                             <Form.Label> Email </Form.Label>
100                         <Form.Control
101                             type='text'
102                             onChange={(e) => {
103                                 setCounsInfo({...counsInfo,
104                                     counsEmail: e.target.value,})
105                             }}
106                             isValid={validated?!error.counsEmail:
107                             false}

```

```

107          />
108      </Form.Group>
109      <Form.Group>
110          <Form.Label> Receive Emails? </Form.Label>
111      >
112          <Form.Control
113              type='checkbox'
114              defaultChecked={true}
115              onClick={()=>{setCounsInfo({
116                  counsInfo, receiveEmail:!counsInfo.receiveEmail})}}
117                  isInvalid={validated?!error.
118                  receiveEmail:false}
119                  isValid={validated?!error.
120                  receiveEmail:false}
121          >
122              />
123          </Form.Group>
124      </Form>
125      </Modal.Body>
126      <Modal.Footer>
127          <Button variant="success" onClick={()=>{create
128              ()}}>
129              Add Counsellor
130          </Button>
131      </Modal.Footer>
132  </Modal>
133  <div className='col-6 pad'>
134      <Card style={{ width: '100%' }}>
135          <Card.Body>
136              <Card.Title className='cardUni'>Name</Card.
137              Title>
138              <Card.Text className='cardTitle'>Email</Card.
139              Text>
140              <Card.Text className='cardEmail'>Receive
141                  Emails?: true/false</Card.Text>
142                  <Button variant="success" className="
143                  cardButton" onClick={() => {handleOpen()}}>Add</Button>
144          </Card.Body>
145      </Card>
146  </div>
147      </>
148  )
149 }
150
151 export default CCardAdd
152

```

```
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
```

```
1 export default function validateCouns(counsInfo) {
2     let errors = {};
3     const regex =
4         /^(([^<>()[]\.,;:\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,}))$/;
5
6     if (counsInfo.name === '' || counsInfo.name === 'Name') {
7         errors.name = 'University name required';
8     }
9
10    if (counsInfo.counsEmail === '' || counsInfo.counsEmail === 'Email') {
11        errors.counsEmail = 'Email required';
12    }
13    } else if (!regex.test(counsInfo.counsEmail)) {
14        errors.counsEmail = 'Invalid email';
15    }
16
17    return errors
18 }
```

```

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/
67 login/verify', {adminToken:token});
68             } else {
69                 verify = false;
70             }
71
72             if(verify){
73                 setValidated(false);
74                 let bookingDateInfoInstance = {
75                     id:instance._id,
76                     aptDate: date,
77                     status: 'Booked',
78                     booking: {
79                         uniName: bookingData.uniName,
80                         uniRepName: bookingData.uniRepName,
81                         uniRepJobTitle: bookingData.uniRepJobTitle,
82                         uniRepEmail: bookingData.uniRepEmail,
83                         uniRegion: bookingData.uniRegion,
84                         logoUrl: bookingData.logoUrl,
85                     },
86                     allowMail: sendEmail,
87                     adminToken: token,
88                 };
89
90                 axios.patch('http://localhost:5000/booking/
91 editbooking', bookingDateInfoInstance);
92                 setShow(false);
93             } else{
94                 alert('Invalid authentication token')
95             }
96
97         }
98
99         async function remove(){
100             const token = localStorage.getItem('adminToken')
101             let verify;
102
103             if(!token) {
104                 verify = await axios.post('http://localhost:5000/login/
105 verify', {adminToken:token});
106             } else {
107                 verify = false;
108             }
109
110             if(verify){
111                 axios.delete('http://localhost:5000/booking/remove', {
112

```

```

109 data: {id:instance._id,adminToken:token}})
110         .then(()=>{
111             // eslint-disable-next-line
112             getDisplayData()
113             .then((res)=>{
114                 setBooking(res.data);
115             })
116             }).then(setShow(false));
117         } else{
118             alert('Invalid authentication token')
119         }
120     }
121
122     return (
123         <>
124             <Modal show={show} onHide={cancel} backdrop="static">
125                 <Modal.Header closeButton>
126                     <Modal.Title>{tempBooking.uniName}</Modal.Title>
127                 </Modal.Header>
128                 <Modal.Body>
129                     <Form >
130                         <Form.Group>
131                             <Form.Label> ID </Form.Label>
132                             <Form.Control
133                                 type='text'
134                                 value={instance._id}
135                                 readOnly={true}
136                             />
137                         </Form.Group>
138                         <Form.Group>
139                             <Form.Label> University Name </Form.Label>
140                         >
141                             <Form.Control
142                                 type='text'
143                                 defaultValue={tempBooking.uniName}
144                                 onChange={(e) => {
145                                     setBookingData({...bookingData,
146                                     uniName: e.target.value,})
147                                 }}
148                                 isValidated={validated?!!error.uniName:
149                                     false}
150                                 isInvalid={validated?!!error.uniName:
151                                     false}
152                             />
153                         </Form.Group>
154                         <Form.Group>
155                             <Form.Label> University Region </Form.
156                             Label>
157                             <Form.Control
158                                 type='text'
159                                 defaultValue={tempBooking.uniRegion}
160                                 onChange={(e) => {
161                                     setBookingData({...bookingData,
162                                     uniRegion: e.target.value,})
163                                 }}
164                                 isInvalid={validated?!!error.
165                                     uniRegion:!!error.uniRegion}
166                         >
167                     </Form.Group>
168                 </Modal.Body>
169             </Modal>
170         </>
171     )
172 
```

```

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

```

```

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

```

```
252             <Card.Body>
253                 <Card.Text className='cardDate'>{dayjs(date).
254                     format('dddd, MMMM D')}</Card.Text>
255                 <Card.Text className='cardTitle'>{bookingData
256                     .uniRepJobTitle + ': ' + bookingData.uniRepName}</Card.Text>
257                 <Card.Text className='cardEmail'>{bookingData
258                     .uniRepEmail}</Card.Text>
259                 <Button variant="primary" className="cardButton" onClick={() => {handleOpen()}}>Edit</Button>
260             </Card.Body>
261         </div>
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
18     // appointment date
19     const [tempDate, setTempDate] = useState("Month/Day/Year"); // Store for cancel
20
21     const [bookingData, setBookingData] = useState({ // Store booking
22         information
23             uniName: "University Name",
24             uniRegion: "University Region",
25             uniRepJobTitle: "Rep Title",
26             uniRepName: "Rep Name",
27             uniRepEmail: "Rep Email",
28             logoUrl: "",
29     });
30     const [tempBooking, setTempBooking] = useState({ // Store for
31         cancel
32             uniName: "University Name",
33             uniRegion: "University Region",
34             uniRepJobTitle: "Rep Title",
35             uniRepName: "Rep Name",
36             uniRepEmail: "Rep Email",
37             logoUrl: "",
38     });
39
40     function cancel(){
41         setBookingData(tempBooking)
42         setDate(tempDate)
43         setShow(false);
44     }
45
46     function handleOpen(){
47         setTempBooking(bookingData);
48         setTempDate(date);
49         setValidated(false);
50         setShow(true);
51     }
52
53     async function getDisplayData() { // Async to prevent stalling
54         let response = await axios.get('http://localhost:5000/booking/
55         readfordisplay');
56         return response;
57     }

```

```

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/
65 login/verify', {adminToken:token});
66         } else {
67           verify =  false;
68         }
69
70         if(verify){
71           setValidated(false);
72
73           let bookingDateInfoInstance = {
74             aptDate: date,
75             status: 'Booked',
76             booking: {
77               uniName: bookingData.uniName,
78               uniRepName: bookingData.uniRepName,
79               uniRepJobTitle: bookingData.uniRepJobTitle,
80               uniRepEmail: bookingData.uniRepEmail,
81               uniRegion: bookingData.uniRegion,
82               logoUrl: bookingData.logoUrl,
83             },
84           };
85
86           axios.post('http://localhost:5000/booking/create',
87 bookingDateInfoInstance)
88             .then(()=>{
89               // eslint-disable-next-line
90               getDisplayData()
91               .then((res)=>{
92                 setBooking(res.data);
93               })
94             }).then(setShow(false));
95           } else{
96             alert('Invalid authentication token')
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          >
110              <Form.Control
111                  type='text'
112                  onChange={(e) => {
113                      setBookingData({...bookingData,
114                          uniName: e.target.value,})
115                  }}
116                  isValid={validated?!error.uniName:
117                      false}
118                  isValid={validated?!error.uniName:
119                      false}
120          />
121          </Form.Group>
122          <Form.Group>
123              <Form.Label> University Region </Form.
124          Label>
125              <Form.Control
126                  type='text'
127                  onChange={(e) => {
128                      setBookingData({...bookingData,
129                          uniRegion: e.target.value,})
130                  }}
131                  isValid={validated?!error.
132                      uniRegion:false}
133                  isValid={validated?!error.uniRegion:
134                      false}
135          />
136          </Form.Group>
137          <Form.Group>
138              <Form.Label> Date </Form.Label>
139              <Form.Control
140                  type='text'
141                  onChange={(e)=> {
142                      setDate(new Date(dayjs(e.target.
143                      value)))
144                  }}
145                  isValid={validated?!error.date:
146                      false}
147                  isValid={validated?!error.date:false}
148          />
149          </Form.Group>
150          <Form.Group>
151              <Form.Label> Rep Title </Form.Label>
152              <Form.Control
153                  type='text'
154                  onChange={(e) => {
155                      setBookingData({...bookingData,
156                          uniRepJobTitle: e.target.value,})
157                  }}
158                  isValid={validated?!error.
159                      uniRepJobTitle:false}
160                  isValid={validated?!error.
161                      uniRepJobTitle:false}
162          />

```

```

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,
158                                  uniRepName: e.target.value,})
159                              }}
160                              isInvalid={validated?!error.
161                                  uniRepName:false}
162                                  isInvalid={validated?!error.uniRepName:
163                                  false}
164                                  />
165                      </Form.Group>
166                      <Form.Group>
167                          <Form.Label> Rep Email </Form.Label>
168                          <Form.Control
169                              type='text'
170                              onChange={(e) => {
171                                  setBookingData({...bookingData,
172                                  uniRepEmail: e.target.value,})
173                              }}
174                              isInvalid={validated?!error.
175                                  uniRepEmail:false}
176                                  isInvalid={validated?!error.uniRepEmail
177                                  :false}
178                                  />
179                      </Form.Group>
180                      <Form.Group>
181                          <Form.Label> Image URL </Form.Label>
182                          <Form.Control
183                              type='text'
184                              onChange={(e) => {
185                                  setBookingData({...bookingData,
186                                  logoUrl: e.target.value,})
187                              }}
188                              isInvalid={validated?true:false}
189                          />
190                      </Form.Group>
191                  </Modal.Body>
192                  <Modal.Footer>
193                      <Button variant="success" onClick={()=>{create
194                          ()}}>
195                          Create Booking
196                      </Button>
197                  </Modal.Footer>
198              </Modal>
199              <div className='col-4 pad'>
200                  <Card style={{ width: '100%' }}>
201                      <Card.Body>
202                          <Card.Title className='cardUni'>University
203                              Name</Card.Title>
204                          <Card.Subtitle className='cardRegion'>
205                              University Region</Card.Subtitle>
206                          </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
14            let res = await axios.get('http://localhost:5000/booking/
readfordisplay');
15            return res;
16        }
17
18        getDisplayData().then((res)=>setBooking(res.data))
19        [], []
20
21        return (
22            <>
23            <div className='container'>
24                <div className='row pad'>
25                    {booking.map(instance=>{
26                        return(
27                            <BCard instance={instance} setBooking={
28                                setBooking}>
29                            )
30                        })
31                    <BCardAdd setBooking={setBooking}/>
32                </div>
33            </div>
34        );
35    }
36    export default EditBookings;
37
```

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

```
1 abbr[title] {
2   border-bottom: none !important;
3   cursor: inherit !important;
4   text-decoration: none !important;
5 }
6 .react-calendar {
7   width: 40rem;
8   max-width: 100%;
9   background: white;
10  /* border: 1px solid #a0a096; */
11  font-family: "system-ui";
12  line-height: 1.125em;
13 }
14 .react-calendar--doubleView {
15   width: 700px;
16 }
17 .react-calendar--doubleView .react-calendar__viewContainer {
18   display: flex;
19   margin: -0.5em;
20 }
21 .react-calendar--doubleView .react-calendar__viewContainer > * {
22   width: 50%;
23   margin: 0.5em;
24 }
25 .react-calendar,
26 .react-calendar *,
27 .react-calendar *:before,
28 .react-calendar *:after {
29   -moz-box-sizing: border-box;
30   -webkit-box-sizing: border-box;
31   box-sizing: border-box;
32 }
33 .react-calendar button {
34   margin: 0;
35   border: 0;
36   outline: none;
37 }
38 .react-calendar button:enabled:hover {
39   cursor: pointer;
40 }
41 .react-calendar__navigation {
42   height: 44px;
43   margin-bottom: 1em;
44 }
45 .react-calendar__navigation button {
46   min-width: 44px;
47   background: none;
48 }
49 .react-calendar__navigation button:enabled:hover,
50 .react-calendar__navigation button:enabled:active {
51   background-color: #e6e6e6;
52 }
53 /* .react-calendar__navigation button[disabled] {
54   /* background-color: #f0f0f0; */
55 /* } */
56 .react-calendar__month-view__weekdays {
57   text-align: center;
```

```
58  text-transform: uppercase;
59  font-weight: bold;
60  font-size: 0.75em;
61
62 }
63 .react-calendar__month-view__weekdays__weekday {
64  padding: 0.5em;
65 }
66 .react-calendar__month-view__weekNumbers {
67  font-weight: bold;
68 }
69 .react-calendar__month-view__weekNumbers .react-calendar__tile {
70  display: flex;
71  align-items: center;
72  justify-content: center;
73  font-size: 0.75em;
74  padding: calc(0.75em / 0.75) calc(0.5em / 0.75);
75 }
76 .react-calendar__month-view__days__day--weekend {
77  color: #d10000;
78 }
79 /* .react-calendar__month-view__days__day--neighboringMonth {
80  color: #757575;
81 } */
82 .react-calendar__year-view .react-calendar__tile,
83 .react-calendar__decade-view .react-calendar__tile,
84 .react-calendar__century-view .react-calendar__tile {
85  padding: 2em 0.5em;
86 }
87 .react-calendar__tile {
88  max-width: 100%;
89  text-align: center;
90  padding: 0.75em 0.5em;
91  background: none;
92 }
93 .react-calendar__tile:disabled {
94  background-color: #f0f0f0;
95 }
96 .react-calendar__tile:enabled:hover,
97 .react-calendar__tile:enabled:focus {
98  background-color: #fff;
99 }
100 .react-calendar__tile--now {
101  background: #fffff76;
102 }
103 .react-calendar__tile--now:enabled:hover,
104 .react-calendar__tile--now:enabled:focus {
105  background: #fffffa9;
106 }
107 .react-calendar__tile--hasActive {
108  background: #76baff;
109 }
110 .react-calendar__tile--hasActive:enabled:hover,
111 .react-calendar__tile--hasActive:enabled:focus {
112  background: #a9d4ff;
113 }
114 .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 }
```

```

54
55      if(verify){
56          let submitData = {
57              changeLog: changeLog,
58              adminToken: token,
59          }
60
61          axios.post('http://localhost:5000/booking/editcalendar',
62          submitData)
63          setChangeLog({})
64      } else {
65          alert('Invalid authentication token')
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!==
78                          changeDate)
79
80                          setDates({...dates,available:newArray})
81
82                          if(tempDates.includes(changeDate)){
83                              setChangeLog({...changeLog, [e]:false})
84                          } else {
85                              let newChanges = changeLog
86                              delete newChanges[e]
87                              setChangeLog(newChanges)
88                          }
89
90                      } else { // Add
91                          setDates({...dates,available:[...dates.
92                          available,changeDate]})}
93
94                          if(tempDates.includes(changeDate)){
95                              let newChanges = changeLog
96                              delete newChanges[e]
97                              setChangeLog(newChanges)
98                          } else {
99                              setChangeLog({...changeLog, [e]:true})
100                         }
101
102                      tileClassName={({ date, view }) => {
103                          if (dates.available.includes(dayjs(date).format(
104                          'DD/MM/YYYY')))) {
105                              return 'cavailable';
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2
```

```
106          } else if (dates.booked.includes(dayjs(date).format('DD/MM/YYYY')))) {
107              return 'cbooked';
108          }}}
```

```
109          tileDisabled={({ date, view }) => {
110              if (date>new Date()) {
111                  return false;
112              } else {
113                  return true;
114              }
115          }}
```

```
116          value=''
117          showNeighboringMonth='false'
118      />
119      <Button
120          disabled={Object.keys(changeLog).length === 0}
121          onClick={()=>{handleSubmit()}}>Save changes</Button>
122      <Button
123          disabled={Object.keys(changeLog).length === 0}
124          onClick={async ()=>{
125              let res = await resetCalendar();
126              setTempDates(res.available);
127              setDates(res);
128          }}
```

```
129          }>Cancel</Button>
130      </div>
131  )
132 )
133 </div>
134 )
135 }
136
137 export default EditCalendar
138
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1
" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="Web site created using create-react-app"
11    />
12    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
14
15    <title>React App</title>
16  </head>
17  <body>
18    <noscript>You need to enable JavaScript to run this app.</noscript
>
19    <div id="root">
20
21    </div>
22  </body>
23 </html>
24
```

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

```
1 {
2   "short_name": "React App",
3   "name": "Create React App Sample",
4   "icons": [
5     {
6       "src": "",
7       "sizes": "64x64 32x32 24x24 16x16",
8       "type": "image/x-icon"
9     },
10    {
11      "src": "",
12      "type": "image/png",
13      "sizes": "192x192"
14    },
15    {
16      "src": "",
17      "type": "image/png",
18      "sizes": "512x512"
19    }
20  ],
21  "start_url": ".",
22  "display": "standalone",
23  "theme_color": "#000000",
24  "background_color": "#ffffff"
25 }
```

```
1 PORT=5000
2 ATLAS_URI=mongodb+srv://userNirav:userNirav8821@cluster0.k3lyw.mongodb
 .net/counsellorWebsite?retryWrites=true&w=majority
3 MAIL_USERNAME=bishvirtualsignup@gmail.com
4 OAUTH_CLIENTID=549431409213-a77901fhj77v8bjmgt6df6jbikm9tak0.apps.
 googleusercontent.com
5 OAUTH_CLIENT_SECRET=byzmdnWwZnSyyXoONTFgaTIh
6 OAUTH_REFRESH_TOKEN=1//04g53NsW8lntqCgYIARAAGAQSNwF-
 L9IrGGn4g1X15SXPjj0HYp5g-
 nnaFChWVID01mPr02tA344qmb2V6yBNHVeJ2K7YVmjHFpw
7 OAUTH_ACCESS_TOKEN=ya29.A0ARrdaM-
 WyL_ZdBFiZuzwzp9BLRCRGKAE9SImtKVgdQMj10vjAphQJkdn_LvMaZ73XdVdIcQ7zWGsZ
 KBz3fqMptmpXNGEv4oqnx8BIsdsAuCzWRJojcom9_-bnULd-D1XT04j324S-
 1FLnopUXRXPEyFZENPr
8 JWT_SECRET=376*QWe@4i5&@7!dp3!&Jg7As7N9p$Bji2^^
9 ADMIN_PASSWORD=123A
10 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
```

```
1 {
2   "name": "server",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "start": "nodemon index.js"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "axios": "^0.24.0",
15    "cors": "^2.8.5",
16    "dotenv": "^10.0.0",
17    "express": "^4.17.1",
18    "googleapis": "^84.0.0",
19    "jsonwebtoken": "^8.5.1",
20    "mongoose": "^5.13.5",
21    "nodemailer": "^6.6.3",
22    "nodemon": "^2.0.12"
23  }
24 }
25
```

```
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')

```

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

```
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 nodemailer transporter with gmail authorization
12 let transporter = nodemailer.createTransport({
13     service: 'gmail',
14     auth: {
15         type: 'OAuth2',
16         user: 'bishvirtalsignup@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
29     let queryString =
30     `https://serpapi.com/search.json?engine=google&q=${query}&
31     google_domain=google.com&gl=us&hl=en&safe=active&tbo=isch&api_key=${
32     process.env.IMAGE_API_KEY}`
33     let res = await axios.get(queryString) // Get result
34     let imageURL = await res.data.images_results[0].thumbnail // Filter for first result
35     return await imageURL
36 }
37
38 const router = express.Router();
39
40 // Get data for calendar formatting
41 router.get('/readforform', (req, res) => {
42     bookingDateInfo.find(
43         { aptDate: { $gte: new Date() } },
44         { aptDate: 1, status: 1, 'booking.uniRegion': 1 })
45     .sort('aptDate')
46     .then((bookingDateInfo) => res.json(bookingDateInfo))
47     .catch((err) => res.status(400).json('Error: ' + err));
48 });
49
50 // Create a booking
51 router.post('/create', (req, res) => {
52     const aptDate = req.body.aptDate;
53     const status = req.body.status;
54     const uniName = req.body.booking.uniName;
55     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 ${aptDate}`,
84          };
85
86          transporter.sendMail(mailOptions, (err, info) => {
87              if (err) {
88                  console.log(err);
89              } else {
90                  console.log('Message sent');
91              }
92          });
93
94          counsellorEmail.find({ receiveEmail: true })
95          .then((counsList)=>{
96              counsList.forEach((couns)=>{
97                  let counsMailOptions = {
98                      from: 'BISH Signup <bishvirtualsignup@gmail.
99                      com> ',
100                     to: couns.counsEmail,
101                     subject: 'Booking Created',
102                     text: `${uniRepName} has just registered for
103                     the: ${uniName}!
104                     The date they registered for is: ${aptDate}
105                     Representative's job title: ${uniRepJobTitle}
106                     Representative's email: ${uniRepEmail}
107                     University's region: ${uniRegion}`,
108                 }
109
110                 transporter.sendMail(counsMailOptions, (err, info

```

```

108  ) => {
109      if (err) {
110          console.log(err);
111      } else {
112          console.log('Message sent');
113      }
114  });
115      })
116  })
117  })
118  .catch((err) => res.status(400).json('Error: ' + err));
119  })
120 });
121
122
123 // Get data for display
124 router.get('/readfordisplay', (req, res) => {
125     bookingDateInfo.find(
126         { aptDate: { $gte: new Date() }, status:"Booked"}, 
127         { aptDate: 1, 'booking': 1 })
128         .sort('aptDate')
129     .then((bookingDateInfo) => res.json(bookingDateInfo))
130     .catch((err) => res.status(400).json('Error: ' + err));
131 });
132
133 // Edit a booking
134 router.patch('/editbooking', (req, res) => {
135     let allowMail = req.body.allowMail;
136     let token = req.body.adminToken
137
138     if(!token && jwt.verify(token, process.env.JWT_SECRET)){
139         bookingDateInfo.findById(req.body.id)
140         .then((bookingDateInfo) => {
141             bookingDateInfo.booking.uniName = req.body.booking.
142                 uniName;
143                 bookingDateInfo.booking.uniRepName = req.body.booking.
144                     uniRepName;
145                 bookingDateInfo.booking.uniRepJobTitle = req.body.booking.
146                     uniRepJobTitle;
147                 bookingDateInfo.booking.uniRepEmail = req.body.booking.
148                     uniRepEmail;
149                 bookingDateInfo.booking.uniRegion = req.body.booking.
150                     uniRegion;
151                 bookingDateInfo.booking.logoUrl = req.body.booking.
152                     logoUrl;
153                 bookingDateInfo.aptDate = req.body.aptDate;
154
155                 bookingDateInfo.save()
156                 .then(() => {
157                     res.json('Booking updated')
158                     if(allowMail){
159                         let mailOptions = {
160                             from: 'BISH Signup <bishvirtualsignup@gmail.
161                             com> ',
162                             to: req.body.booking.uniRepEmail,
163                             subject: 'Updated presentation information',
164                             text: `Your booking information for a

```



```

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,
233 status:"Available"}}})
234             } else {
235                 changes.push({'deleteOne':{"filter":{aptDate:key}}})
236             }
237         });
238
239         bookingDateInfo.bulkWrite(changes)
240         .then(() => {
241             res.json('Dates updated')
242             counsellorEmail.find(
243                 { receiveEmail: true },
244                 { counsEmail: 1 }
245             )
246             .then((counsList)=>{
247                 Object.keys(changeLog).forEach((key) => {
248                     let text = '';
249                     if(changeLog[key]==true){
250                         text = `${text} \n ${key} - Added`
251                     } else {
252                         text = `${text} \n ${key} - Removed`
253                     }
254
255                     counsList.forEach((couns)=>{
256                         let counsMailOptions = {

```

```

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

```

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

```
1 const { app, BrowserWindow } = require('electron')
2
3 const createWindow = () => {
4     const win = new BrowserWindow({
5         width: 800,
6         height: 600
7     })
8
9     win.loadURL('http://localhost:3000')
10 }
11
12 app.whenReady().then(() => {
13     createWindow()
14 })
```

```
1 # Getting Started with Create React App
2
3 This project was bootstrapped with [Create React App](https://github.
4 com/facebook/create-react-app).
5
6 ## Available Scripts
7
8 In the project directory, you can run:
9
10 #### `npm start`
11 Runs the app in the development mode.\
12 Open [http://localhost:3000](http://localhost:3000) to view it in your
13 browser.
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/
21 create-react-app/docs/running-tests) for more information.
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
26 for the best performance.
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-
31 react-app/docs/deployment) for more information.
32 #### `npm run eject`
33
34 **Note: this is a one-way operation. Once you `eject`, you can't go
35 back!**
36 If you aren't satisfied with the build tool and configuration choices
37 , you can `eject` at any time. This command will remove the single
38 build dependency from your project.
39 Instead, it will copy all the configuration files and the transitive
40 dependencies (webpack, Babel, ESLint, etc) right into your project so
41 you have full control over them. All of the commands except `eject`
42 will still work, but they will point to the copied scripts so you can
43 tweak them. At this point you're on your own.
```

```
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
```

```
1 # See https://help.github.com/articles/ignoring-files/ for more about
  ignoring files.
2
3 # dependencies
4 /node_modules
5 /.pnp
6 .pnp.js
7
8 # testing
9 /coverage
10
11 # production
12 /build
13
14 # misc
15 .DS_Store
16 .env.local
17 .env.development.local
18 .env.test.local
19 .env.production.local
20
21 npm-debug.log*
22 yarn-debug.log*
23 yarn-error.log*
24
```

```

1  {
2    "name": "electron-app",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.16.2",
7      "@testing-library/react": "^12.1.4",
8      "@testing-library/user-event": "^13.5.0",
9      "axios": "^0.26.1",
10     "bootstrap": "^4.6.0",
11     "concurrently": "^7.0.0",
12     "cross-env": "^7.0.3",
13     "dayjs": "^1.10.8",
14     "electron": "^17.1.2",
15     "react": "^17.0.2",
16     "react-bootstrap": "^1.6.1",
17     "react-dom": "^17.0.2",
18     "react-scripts": "5.0.0",
19     "wait-on": "^6.0.1",
20     "web-vitals": "^2.1.4"
21   },
22   "main": "main.js",
23   "scripts": {
24     "start": "react-scripts start",
25     "build": "react-scripts build",
26     "test": "react-scripts test",
27     "eject": "react-scripts eject",
28     "watch": "webpack --config webpack.common.js --watch",
29     "electron:start": "electron .",
30     "electron-react": "concurrently \\\"cross-env BROWSER=none npm start \\\" \\\"wait-on http://localhost:3000 && electron .\\\""
31   },
32   "eslintConfig": {
33     "extends": [
34       "react-app",
35       "react-app/jest"
36     ]
37   },
38   "browserslist": {
39     "production": [
40       ">0.2%",
41       "not dead",
42       "not op_mini all"
43     ],
44     "development": [
45       "last 1 chrome version",
46       "last 1 firefox version",
47       "last 1 safari version"
48     ]
49   }
50 }
51

```

```
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
```

```
1 .cardUni{  
2     font-size: 1.5rem;  
3     font-weight: 300;  
4 }  
5  
6 .cardRegion{  
7     font-size: 1rem;  
8     font-weight: 500;  
9     color: #6c757d;  
10 }  
11  
12 .cardImage{  
13     width: calc(100% + 2px);  
14     border-radius: 0;  
15     margin-left: -1px;  
16     border-left: 1px solid #DFDFDF;  
17     border-right: 1px solid #DFDFDF;  
18 }  
19  
20 .cardDate{  
21 }  
22 }  
23  
24 .cardTitle{  
25 }  
26 }  
27  
28 .cardEmail{  
29 }  
30 }  
31  
32 .pad{  
33     margin-top: 1rem;  
34     margin-bottom: 1rem;  
35 }  
36  
37 .cardButton{  
38     width: 100%;  
39 }  
40  
41 .login-container{  
42     display: flex;  
43     justify-content: center;  
44     align-items: center;  
45 }  
46  
47 .ccontainer {  
48     margin-left: calc(50vw - 20rem);  
49     margin-top: calc(50vh - 15rem);  
50 }  
51  
52 .cbooked {  
53     color: hsla(3, 100%, 20%, 1) !important;  
54     background-color: rgba(255, 65, 54, 0.6) !important;  
55 }  
56  
57 .cbooked:hover {
```

```
58     color: hsla(3, 100%, 20%, 1) !important;
59     background-color: rgba(255, 65, 54, 0.55) !important;
60 }
61
62 .cavailable {
63     color: hsla(127, 63%, 15%, 1) !important;
64     background-color: rgba(1, 255, 112, 0.7) !important;
65 }
66
67 .cavailable:hover {
68     color: hsla(127, 63%, 15%, 1) !important;
69     background-color: rgba(1, 255, 112, 0.5) !important;
70 }
71
72 .topbar{
73     height: 10vh;
74     background: #f8f9fa;
75     position: absolute;
76     width:100%;
77     text-align: center;
78     padding-top: 2vh;
79 }
80
81 .botbar{
82     bottom: 0;
83     height: 10vh;
84     background: #f8f9fa;
85     position: absolute;
86     width:100%;
87 }
88
89 .vaf{
90     display: flex;
91     justify-content: center;
92     align-items: center;
93     padding-top: 15vh;
94 }
```

```
1 body {  
2   margin: 0;  
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto'  
4   , 'Oxygen',  
5   'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue'  
6   ,  
7   sans-serif;  
8   -webkit-font-smoothing: antialiased;  
9   -moz-osx-font-smoothing: grayscale;  
10 }  
11  
12 code {  
13   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New'  
14   ,  
15   monospace;  
16 }  
17  
18
```

```
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?
28                        bookingData.logoUrl:image} className='cardImage'>
29                    </Card.Img>
30                    <Card.Text className='cardDate'>{dayjs(date).format('
31 dddd, MMMM D')}</Card.Text>
32                    <Card.Text className='cardTitle'>{bookingData.
33 uniRepJobTitle + ' : ' + bookingData.uniRepName}</Card.Text>
34                    <Card.Text className='cardEmail'>{bookingData.
35 uniRepEmail}</Card.Text>
36                </Card.Body>
37            </div>
38        </>
39    )
40 }
41
42 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/
13            readfordisplay');
14            return res;
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                    )})
30                </div>
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="sha384-1BtW8XqETiQVcW6fO8V3mGuXNq5FkqVZBt+ndW6U8v
iJTQU0hcWr7x9JvoRxT2M Zw1T"
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
```

```
1 {
2   "short_name": "React App",
3   "name": "Create React App Sample",
4   "icons": [
5     {
6       "src": "favicon.ico",
7       "sizes": "64x64 32x32 24x24 16x16",
8       "type": "image/x-icon"
9     },
10    {
11      "src": "logo192.png",
12      "type": "image/png",
13      "sizes": "192x192"
14    },
15    {
16      "src": "logo512.png",
17      "type": "image/png",
18      "sizes": "512x512"
19    }
20  ],
21  "start_url": ".",
22  "display": "standalone",
23  "theme_color": "#000000",
24  "background_color": "#ffffff"
25 }
26
```