



Web App Design with React Final Project

LisaSmith — 2022-10-05-fesd-nashua — originally due: 3/5/22

With a 3 week extension, now due: 3/26/22 — Thank you Tyler!

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In VS Code, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your JavaScript project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

1. Using an online API of your choice, or multiple APIs (or if no API can be found, using an array for in-memory storage is okay as well), create a React project of your choice. You will be working on this for the next three weeks.
2. Project must meet the following criteria:
 - a. Use React Router and have at least 3 pages
 - b. Use React Bootstrap or an alternative styling library
 - c. Contain at least 10 components
 - d. Allow for all CRUD operations



Description:

My Web API called Find-A-Gig, is a working API, which connects Musicians to Available Gigs, and Gig Organizers to Available Musicians. When I started the Front End Coding Bootcamp, my intention was always to connect a front end to that back end. At the moment, my Front End React Final Project — Find-A-Gig-App is a work in progress. I have completed the following:

1. The API used was crudcrud with the following endpoints:

- <https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs>
- <https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/users>
- <https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/instruments>

2. My React Project, Find-A-Gig-App:

- **4 pages:** Home, Users, Gigs & Instruments
- **Uses:** React Router v6 & the React Bootstrap library
- **Contains 12 components:** gigs, home, instruments, modal, mybutton, navLinks, newGigForm, newGigInstrumentForm, newInstrumentForm, newUserForm, newUserInstrumentForm, users.
- **Implements all 4 CRUD Operations:** Create, Read, Update, & Delete
- **All API calls are in:** GigApi, InstrumentApi, & UserApi

URL to GitHub Repository: <https://github.com/sw-dev-lisa-s-nh/React-Final-Project>

Screenshots of Code:

index.js:

```
find-a-gig-app > src > JS index.js
  1  import React from 'react';
  2  import ReactDOM from 'react-dom';
  3  import 'bootstrap/dist/css/bootstrap.min.css';
  4  import './index.css';
  5  import { BrowserRouter } from 'react-router-dom';
  6  import App from './App';
  7  import reportWebVitals from './reportWebVitals';

  8
  9  ReactDOM.render(
 10    <BrowserRouter>
 11      |   <App />
 12      |   </BrowserRouter>,
 13      |   document.getElementById('root')
 14    );
 15    |
 16    |   reportWebVitals();
 17
```



PROMINEO TECH

App.js:

```
find-a-gig-app > src > js App.js > [e] default
  1  import React from 'react';
  2  import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
  3  import Gigs from './components/gigs';
  4  import Users from './components/users';
  5  import Instruments from './components/instruments';
  6  import Navigation from './components/navLinks';
  7  import HomePage from './components/home';
  8  import { Routes, Route } from 'react-router-dom';
  9
 10
 11 function App() {
 12   return (
 13     <div>
 14       <Navigation />
 15       <Routes>
 16         <Route path="/" element={ <HomePage />} />
 17         <Route path="/users" element={ <Users />} />
 18         <Route path="/instruments" element={ <Instruments />} />
 19         <Route path="/gigs" element={ <Gigs />} />
 20       </Routes>
 21     </div>
 22   );
 23 }
 24
 25 export default App;
```

index.css:

```
find-a-gig-app > src > # index.css > [e] h1
  1
  2 body {
  3   margin: 2;
  4   color: ■rgb(107, 107, 206);
  5   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
  6   | 'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue', 'Fantasy',
  7   sans-serif;
  8   -webkit-font-smoothing: antialiased;
  9   -moz-osx-font-smoothing: grayscale;
 10   background-image: url('../music.jpg');
 11 }
 12
 13
 14 code {
 15   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
 16   monospace;
 17 }
 18
 19 h1,h2,h3,h4,h5,h6, ul, th, td, table, .accord-custom [e]
 20   padding: 2%;
 21   margin: 2%;
 22   font-family: fantasy;
 23   font-display: ■rgb(107, 107, 206);
 24   color: ■rgb(107, 107, 206);
 25   background: ■rgb(174, 174, 230);
 26   border: ■rgb(107, 107, 206);
 27   box-shadow: 5px 5px 5px ■rgb(107, 107, 206);
 28
 29
 30
```



PROMINEO TECH

index.css continued:

```
nd-a-gig-app > src > # index.css > .new-user

31  .navbar-class, .nav-link, .nav-bar-title, .my-button {
32    padding: 2%;
33    margin: 2%;
34    size: 36;
35    text-emphasis-color: ■rgb(107, 107, 206);
36    font-family: fantasy;
37    font-display: ■rgb(107, 107, 206);;
38    color: ■rgb(107, 107, 206);
39    background: ■rgb(174, 174, 230);
40    border: ■rgb(107, 107, 206);
41    box-shadow: 5px 5px 5px ■rgb(107, 107, 206);
42  }

43
44
45 .new-user, .new-card, .new-instrument, .info-button, .instrument-list {
46   margin: 2%;
47   padding: 1%;
48   font-family: fantasy;
49   align-content: space-between;
50   color: ■rgb(66, 66, 97);
51   background: ■rgb(198, 198, 207);
52   border-radius: 4px;
53   border-color: ■rgb(107, 107, 206);
54   border-color: ■rgb(107, 107, 206); |
55   box-shadow: 5px 5px 5px ■rgb(107, 107, 206);
56 }
57
58 .new-form {
59   font-family: fantasy;
60   color: ■rgb(34, 34, 202);
61   border-color: ■rgb(107, 107, 206);
62   border-color: ■rgb(107, 107, 206);
63   box-shadow: 5px 5px 5px ■rgb(107, 107, 206);
64 }
65
66
67 .button-custom {
68   size: lg;
69   size: 36px;
70   border-color: ■rgb(107, 107, 206);
71   box-shadow: 5px 5px 5px ■rgb(107, 107, 206);
72 }
73
74
75 ul {
76   font-size: 24px;
77 }
78
79 .gig-card {
80   background-color: ■rgb(107, 107, 206);
81 }
82
83 .info-gig-card {
84   background-color: ■rgb(209, 209, 250);
85 }
```



GigApi.js:

```
node-a-gig-app > src > rest > js GigApi.js > ...
1  const GIG_ENDPOINT = 'https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs';
2
3  class GigApi {
4    get = async () => {
5      try {
6        console.log(GIG_ENDPOINT)
7        const resp = await fetch(`${GIG_ENDPOINT}`);
8        console.log(resp);
9        const data = await resp.json();
10       console.log(data);
11       return data;
12     } catch(e) {
13       console.log('Error:  get gigs had an issue.', e);
14     }
15   }
16
17   getOne = async (id) => {
18     try {
19       console.log(GIG_ENDPOINT)
20       const resp = await fetch(`${GIG_ENDPOINT}/${id}`);
21       console.log(resp);
22       const data = await resp.json();
23       console.log(data);
24       return data;
25     } catch(e) {
26       console.log(`Error:  get gig id: ${id} had an issue.`, e);
27     }
28   }
29
30
31   post = async (gig) => {
32     try {
33       const resp = await fetch(`${GIG_ENDPOINT}`, {
34         method: 'POST',
35         headers: {
36           'Content-Type' : 'application/json'
37         },
38         body: JSON.stringify(gig)
39       });
40       return await resp.json();
41     } catch(e) {
42       console.log('Error:  post gig had an issue.', e);
43     }
44   }
45
46
47   put = async (gig) => {
48     try {
49       console.log(GIG_ENDPOINT);
50       const resp = await fetch(`${GIG_ENDPOINT}/${gig._id}`, {
51         method: 'PUT',
52         headers: {
53           'Content-Type' : 'application/json'
54         },
55         body: JSON.stringify({ "name": gig.name, "date": gig.date, "address": gig.address,
56           "phone": gig.phone, "description": gig.description, "instruments": gig.instruments}),
57       });
58       console.log(resp);
59       return await resp;
60     } catch(e) {
61       console.log('Error:  update gig had an issue.', e);
62     }
63   }
64
65   delete = async (gigId) => {
66     try {
67       console.log(GIG_ENDPOINT);
68       const resp = await fetch(`${GIG_ENDPOINT}/${gigId}`, {
69         method: 'DELETE'
70       });
71       console.log(resp);
72     } catch(e) {
73       console.log('Error:  delete gig had an issue.', e);
74     }
75   }
76
77 // create an instance of this class, and we can import that instance.
78 export const gigApi = new GigApi();
79
```



InstrumentApi.js:

```
find-a-gig-app > src > rest > js UserApi.js > ...
1  const USER_ENDPOINT = 'https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/users';
2
3  class UserApi {
4      get = async () => {
5          try {
6              console.log(`${USER_ENDPOINT}`);
7              const resp = await fetch(`${USER_ENDPOINT}`);
8              console.log(resp);
9              const data = await resp.json();
10             console.log(data);
11             return data;
12         } catch(e) {
13             console.log('Error: get users had an issue.', e);
14         }
15     }
16
17
18     getOne = async (id) => {
19         try {
20             console.log(`${USER_ENDPOINT}`);
21             const resp = await fetch(`${USER_ENDPOINT}/${id}`);
22             console.log(resp);
23             const data = await resp.json();
24             console.log(data);
25             return data;
26         } catch(e) {
27             console.log('Error: get user id: ${id} had an issue.', e);
28         }
29     }
30
31
32
33     post = async (user) => {
34         try {
35             const resp = await fetch(`${USER_ENDPOINT}`, {
36                 method: 'POST',
37                 headers: {
38                     'Content-Type': 'application/json'
39                 },
40                 body: JSON.stringify(user)
41             });
42             return await resp.json();
43         } catch(e) {
44             console.log('Error: post users had an issue.', e);
45         }
46     }
47
48     put = async (user) => {
49         try {
50             console.log(USER_ENDPOINT);
51             const resp = await fetch(`${USER_ENDPOINT}/${user._id}`, {
52                 method: 'PUT',
53                 headers: {
54                     'Content-Type': 'application/json'
55                 },
56                 body: JSON.stringify({
57                     "firstname": user.firstname,
58                     "lastname": user.lastname,
59                     "address": user.address,
60                     "city": user.city,
61                     "state": user.state,
62                     "phone": user.phone,
63                     "instruments": user.instruments
64                 });
65             console.log(resp);
66             //return resp.json();
67             return resp;
68         } catch(e) {
69             console.log('Error: update users had an issue.', e);
70         }
71     }
72
73     delete = async (userId) => {
74         try {
75             console.log(USER_ENDPOINT);
76             const resp = await fetch(`${USER_ENDPOINT}/${userId}`, {
77                 method: 'DELETE'
78             });
79             console.log(resp);
80         } catch(e) {
81             console.log('Error: delete user had an issue.', e);
82         }
83     }
84 }
85
86 // create an instance of this class, and we can import that instance.
87 export const userApi = new UserApi();
```



PROMINEO TECH

UserApi.js:

```
find-a-gig-app > src > rest > js UserApi.js > ...
1  const USER_ENDPOINT = 'https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/users';
2
3  class UserApi {
4      get = async () => {
5          try {
6              console.log(` ${USER_ENDPOINT}`);
7              const resp = await fetch(` ${USER_ENDPOINT}`);
8              console.log(resp);
9              const data = await resp.json();
10             console.log(data);
11             return data;
12         } catch(e) {
13             console.log('Error: get users had an issue.', e);
14         }
15     }
16
17     getOne = async (id) => {
18         try {
19             console.log(` ${USER_ENDPOINT}`);
20             const resp = await fetch(` ${USER_ENDPOINT}/${id}`);
21             console.log(resp);
22             const data = await resp.json();
23             console.log(data);
24             return data;
25         } catch(e) {
26             console.log('Error: get user id: ${id} had an issue.', e);
27         }
28     }
29
30
31
32     post = async (user) => {
33         try {
34             const resp = await fetch(` ${USER_ENDPOINT}`, {
35                 method: 'POST',
36                 headers: {
37                     'Content-Type' : 'application/json'
38                 },
39                 body: JSON.stringify(user)
40             });
41             return await resp.json();
42         } catch(e) {
43             console.log('Error: post users had an issue.', e);
44         }
45     }
46
47     put = async (user) => {
48         try {
49             console.log(USER_ENDPOINT);
50             const resp = await fetch(` ${USER_ENDPOINT}/${user._id}`, {
51                 method: 'PUT',
52                 headers: {
53                     'Content-Type' : 'application/json'
54                 },
55                 body: JSON.stringify({
56                     "firstname" : user.firstname,
57                     "lastname" : user.lastname,
58                     "address" : user.address,
59                     "city" : user.city,
60                     "state" : user.state,
61                     "phone" : user.phone,
62                     "instruments": user.instruments
63                 });
64             console.log(resp);
65             //return resp.json();
66             return resp;
67         } catch(e) {
68             console.log('Error: update users had an issue.', e);
69         }
70     }
71
72     delete = async (userId) => {
73         try {
74             console.log(USER_ENDPOINT);
75             const resp = await fetch(` ${USER_ENDPOINT}/${userId}`, {
76                 method: 'DELETE'
77             });
78             console.log(resp);
79         } catch(e) {
80             console.log('Error: delete user had an issue.',e);
81         }
82     }
83
84 }
85
86 // create an instance of this class, and we can import that instance.
87 export const userApi = new UserApi();
88
```



modal.js:

```
JS mybutton.js 1   JS instruments.js 1   JS modal.js M X  JS gigs.js 1   JS users.js 1   JS home.js
find-a-gig-app > src > components > JS modal.js > ModalWindow
1 import React from 'react';
2 import { useState } from 'react';
3 import { Modal } from 'react-bootstrap';
4
5 function ModalWindow() {
6   const [show, setShow] = useState(false);
7   const handleClose = () => setShow(false);
8   const handleShow = () => setShow(true);
9
10  return(
11    <>
12      <button onClick={handleShow}>
13        Find-A-Gig-App
14      </button>
15
16      <Modal show={show} onHide={handleClose}>
17        <Modal.Header closeButton>
18          <Modal.Title>Find-A-Gig-App Description</Modal.Title>
19        </Modal.Header>
20        <Modal.Body>
21          <strong>Find-A-Gig-App</strong> has a two-fold purpose. <br /> <br /> It is a way for
22          <strong>Musicians</strong> to register, listing the instruments that they play; and to
23          request a booking to play at Open Gigs. <br /> <br /> It also allows <strong>Event
24          Planners</strong> to register gigs, listing instruments needed; and to connect and
25          hire musicians!
26        </Modal.Body>
27        <Modal.Footer>
28          <button onClick={handleClose}>
29            Close
30          </button>
31        </Modal.Footer>
32      </Modal>
33    </>
34  );
35}
36
```

home.js:

```
JS mybutton.js 1   JS instruments.js 1   JS gigs.js 1   JS users.js 1   JS home.js M X
find-a-gig-app > src > components > JS home.js > ...
1 import React from 'react';
2 import '../../../../../node_modules/bootstrap/dist/css/bootstrap.min.css';
3 // import { Popover } from 'react-bootstrap';
4
5 import MyButton from './mybutton';
6
7
8 function HomePage (props) {
9   return (
10     <div className="container">
11       <div className="row mt-5">
12         <div className="col-md-8 mx-auto">
13           <h1 className="text-center">Welcome to Find-A-Gig</h1>
14         </div>
15       </div>
16       <div className="col-md-2 flex mx-auto" >
17         <MyButton className="info-button" />
18       </div>
19     </div>
20   )
21 }
22
23 export default HomePage;
24
```



PROMINEO TECH

mybutton.js:

```
js mybutton.js 1 X js instruments.js 1 js gigs.js 1 js users.js 1
find-a-gig-app > src > components > js mybutton.js > MyButton
1   import React from 'react';
2   import { useState } from 'react';
3   import '../../../../../node_modules/bootstrap/dist/css/bootstrap.min.css';
4   // import { Button } from 'react-bootstrap';
5   import ModalWindow from './modal';
6
7   function MyButton (props) {
8
9     const { label } = props;
10    const [show, setShow] = useState(false);
11
12    return (
13      <div className="container my-3">
14        <button className="info-button"
15          onClick={() => setShow(true)}
16        >
17        <br />
18        <strong>{label} </strong>
19        <ModalWindow />
20        <br /><br />
21        </button>
22      </div>
23    );
24
25  }
26
27  export default MyButton;
28
```



gigs.js:

```
find-a-gig-app | src > components > Gigs.js | ↵ Gigs | ↵ useEffect() callback
1 import React from 'react';
2 import { useState, useEffect } from 'react';
3 import { Card } from 'react-bootstrap';
4 import NewGigForm from './newGigForm';
5 import { gigApi } from '../rest/GigApi.js';
6 import { gigApi } from './rest/GigApi.js';
7
8
9 function Gigs (props) {
10   const [error] = useState(null);
11   const [isLoaded, setIsLoaded] = useState(false);
12   const [gig, setGig] = useState([]);
13   // const [gigInstruments, setGigInstruments] = useState([]);
14
15   const createGig = async (newGig) => {
16     await gigApi.postNewGig();
17     runThisEveryTime();
18   };
19
20   const deleteGig = sync (gigId) => {
21     await gigApi.delete(gigId);
22     runThisEveryTime();
23   };
24
25   const removeGigInstrument = async (gigId, instrumentId) => {
26     console.log('removeGigInstrument', gigId);
27     let updatedGig = await gigApi.getOne(gigId);
28     console.log('In removeGigInstrument, after getOne', newGig);
29     const updatedGig = [
30       ...newGig,
31       instruments: newGig.instruments.filter((x) => x._id !== instrumentId)
32     ];
33     console.log('updatedGig: ' + updatedGig);
34     gigApi.put(updatedGig);
35     runThisEveryTime();
36   };
37   const runThisEveryTime = () => (
38     fetch('https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs')
39     .then(res => res.json())
40     .then(
41       (result) => {
42         setIsLoaded(true);
43         setGig(result);
44         console.log(result);
45       },
46       (error) => {
47         setIsLoaded(true);
48         setGig(error);
49         console.log(error);
50       }
51     )
52   );
53
54  useEffect(() => {
55    runThisEveryTime();
56  }, []);
57
58  if (error) {
59    return <div>An error was encountered: {error.message}</div>
60  } else {
61    return (
62      <div className="container">
63        <div className="row my-4">
64          <div className="col-md-8 mx-auto">
65            <h2 className="text-center"><strong>Available Gigs</strong></h2>
66          </div>
67        </div>
68        <div className="row">
69          {gig.map((g, idx) => (
70            <div className="gig-card col-4 my-3" key={g._id}>
71              <Card className="info-gig-card m-2 p-2">
72                <Card.Body>
73                  <Card.Title>{g._id}&nbsp;<strong>{g.name}</strong></Card.Title>
74                  <Card.Text>
75                    Id: {g._id} <br />
76                    Address: {g.address} <br />
77                    City: {g.city} <br />
78                    State: {g.state} <br />
79                    Description: {g.description}<br />
80                    Contact Info: {g.phone} <br />
81                    Instruments Requested:<br />
82                    <ul>
83                      {g.instruments.map((inst, j) => (
84                        <li className="border" key={inst._id}> &nbsp;
85                          <strong>{inst.name}</strong>
86                          <br />
87                          <button className="btn-my-color rounded" onClick={(e) =>
88                            removeGigInstrument(g._id, inst._id)}>Remove </button>
89                        </li>
90                      ))
91                    )}
92                  </Card.Text>
93                </Card.Body>
94                <Card.Footer>
95                  <button className="btn-my-color rounded" onClick={(e) =>
96                    deleteGig(g._id)}>Delete Gig </button>
97                </Card.Footer>
98              </Card>
99            </div>
100          ))
101        </div>
102        <NewGigForm createGig={createGig} />
103      </div>
104      <NewGigForm createGig={createGig} />
105    );
106  }
107}
108
109 export default Gigs;
```

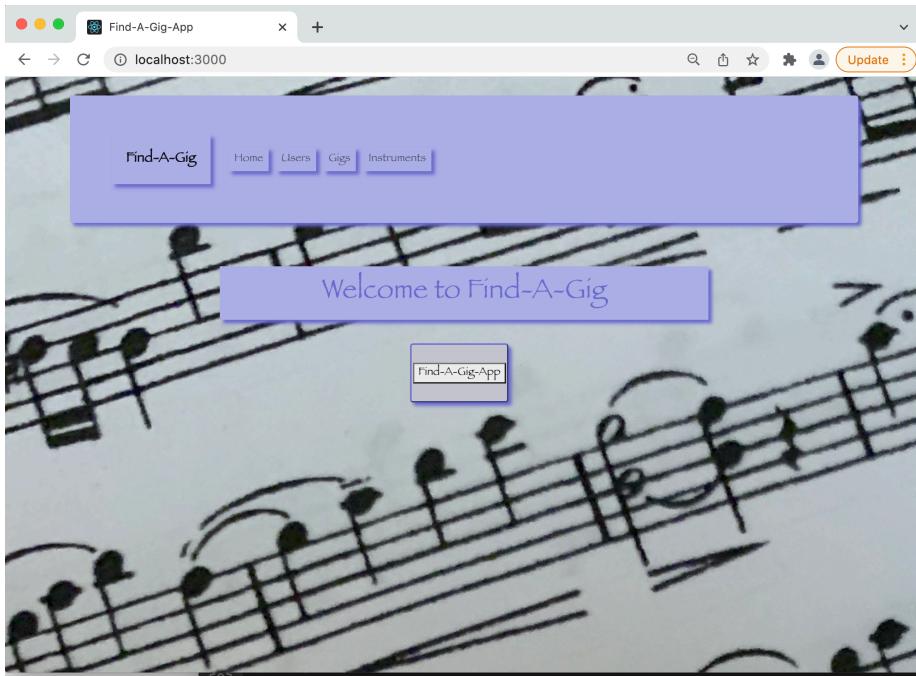
See the following files on GitHub: instruments.js, users.js, newGigForm.js, newGigInstrumentForm.js, newInstrumentForm.js, newUserForm.js, newUserInstrumentForm.js



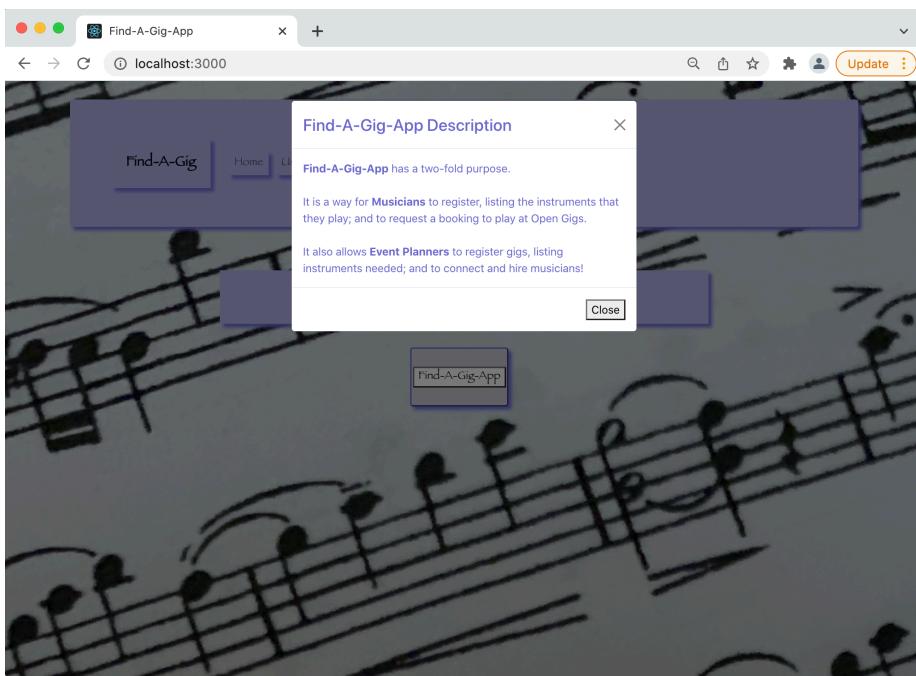
PROMINEO TECH

Screenshots of Running Application:

Home Page:



Click on the button (pops up a modal):





PROMINEO TECH

Users Page (Read):

The screenshot shows a web browser window for the 'Find-A-Gig-App' at localhost:3000/users. The page has a purple header bar with the 'Find-A-Gig' logo and navigation links for Home, Users, Gigs, and Instruments. Below the header is a section titled 'Available Musicians' containing a table with two rows of data. The first row represents a musician named Donald Duck with ID `621eef656780a03e8ddb659`, address 123 Main St., Nashua, NH, phone 987-654-3210, and instruments Cello and Piano. The second row represents a musician named Minnie Mouse with ID `6226b4c06f047803e8ae0d57`, address 345 Main St., Disneyland, CA, phone 987-654-3210, and instrument Piano. Each row includes 'Delete User' and 'Remove' buttons. At the bottom of the page is a modal dialog titled 'Enter a New Musician' with fields for First Name, Last Name, Address, City, State, and Phone, and an 'Add Musician' button.

ID	First Name	Last Name	Address	Phone	Instruments
<code>621eef656780a03e8ddb659</code>	Donald	Duck	123 Main St., Nashua, NH	987-654-3210	• Cello • Piano
<code>6226b4c06f047803e8ae0d57</code>	Minnie	Mouse	345 Main St., Disneyland, CA	987-654-3210	• Piano

The screenshot shows a browser developer tools interface with multiple network tabs. One tab is active, showing a POST request to `crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/users`. The request body contains the following JSON data:

```
[{"_id": "621eef656780a03e8ddb659", "firstname": "Donald", "lastname": "Duck", "address": "123 Main St.", "city": "Nashua", "state": "NH", "phone": "987-654-3210", "instruments": [{"_id": "62144e8d56780a03e8ddae90", "name": "Cello", "section": "Strings"}]}, {"_id": "6226b4c06f047803e8ae0d57", "firstname": "Minnie", "lastname": "Mouse", "address": "345 Main St.", "city": "Disneyland", "state": "CA", "phone": "987-654-3210", "instruments": [{"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}]}]
```



PROMINEO TECH

Gigs Page (Read):

The screenshot shows a web browser window titled "Find-A-Gig-App" at "localhost:3000/gigs". The page has a header with "Find-A-Gig" and navigation links for Home, Users, Gigs, and Instruments. Below the header is a purple box containing the text "Available Gigs". Two gigs are listed in a grid:

- 1. New Year's Celebration**
Id: 621eb2b856780a03e8ddb638
On: 2022-12-31
Located at:
345 Market St., .
Description: Ring in the New Year
Contact Info: 345-678-9012
Instruments Requested:
 - Piano

[Remove](#)

[Delete Gig](#)
- 2. Memorial Day Parade**
Id: 621f8f1f56780a03e8ddb681
On: 2022-05-28
Located at:
Union Square, .
Description: Play Taps at Each War Memorial
Contact Info: 123-456-7890
Instruments Requested:
 - Trumpet

[Remove](#)

[Delete Gig](#)

At the bottom of the page is a form titled "Enter a New Gig" with fields for Name of Gig, Date, Duration, Address, City, State, Phone, Description, and an "Add Gig" button.

Data stored associated with this screenshot:

The screenshot shows a browser window with multiple tabs open. The active tab is "crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs". The page displays a JSON array of gig data:

```
[{"_id": "621eb2b856780a03e8ddb638", "name": "New Year's Celebration", "date": "2022-12-31", "address": "345 Market St.", "phone": "345-678-9012", "description": "Ring in the New Year", "instruments": [{"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}]}, {"_id": "621f8f1f56780a03e8ddb681", "name": "Memorial Day Parade", "date": "2022-05-28", "address": "Union Square", "phone": "123-456-7890", "description": "Play Taps at Each War Memorial", "instruments": [{"_id": "6226c0db6f047803e8ae0d5b", "name": "Trumpet", "section": "Brass"}]}]
```



Instruments Page (Read):

The screenshot shows a web application interface titled "Find-A-Gig" with a purple header bar. The main content area displays a grid of three instrument categories: "Cello, Strings", "Piano, Percussion", and "Trumpet, Brass". Each category has a form with fields for "User Id" and "Gig Id", and buttons for "Add User Instrument", "Add Gig Instrument", and "Delete Instrument". Below this grid is a modal window titled "Enter a New Instrument" with fields for "Instrument Name" and "Instrument Section", and a "Add Instrument" button.

The screenshot shows a browser developer tools Network tab with several requests listed. One request, "Get /api/instruments", has a response body containing the following JSON data:

```
[{"_id": "62144e8d56780a03e8ddae90", "name": "Cello", "section": "Strings"}, {"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}, {"_id": "6226c0db6f047803e8ae0d5b", "name": "Trumpet", "section": "Brass"}]
```



PROMINEO TECH

Enter a New Musician
(Create):

The screenshot shows a web browser window titled "Find-A-Gig-App" at "localhost:3000/users". The main content area is titled "Available Musicians" and displays a table of musicians:

ID	First Name	Last Name	Address	Phone	Instruments
621eeef656780a03eddb59	Donald	Duck	123 Main St., Nashua, NH	987-654-3210	• Cello Delete User Remove
622d14c00f04780a03eddb57	Minnie	Mouse	545 Main St., Disneyland, CA	987-654-3210	• Piano Delete User Remove

Below the table is a modal dialog titled "Enter a New Musician" containing fields for First Name (Jiminy), Last Name (Cricket), Address (987 Poplar Lane), City (Boston), State (MA), and Phone (715-908-7654). A "Add Musician" button is at the bottom of the form.

After the **Add Musician** Button is pushed:

The screenshot shows the same web browser window at "localhost:3000/users". The "Available Musicians" table now includes the new entry:

ID	First Name	Last Name	Address	Phone	Instruments
621eeef656780a03eddb59	Donald	Duck	123 Main St., Nashua, NH	987-654-3210	• Cello Delete User Remove
622d14c00f04780a03eddb57	Minnie	Mouse	545 Main St., Disneyland, CA	987-654-3210	• Piano Delete User Remove
622d49ca0f04780a03eddb5f	Jiminy	Cricket	987 Poplar Lane, Boston, MA	715-908-7654	Delete User

The "Enter a New Musician" modal is still visible at the bottom of the screen.



PROMINEO TECH

Enter a New Gig
(Create):

The screenshot shows a web browser window titled "Find-A-Gig-App" at "localhost:3000/gigs". The interface has a purple header bar with tabs for "Home", "Users", "Gigs", and "Instruments". Below the header is a large background image of musical sheet music. A purple box highlights the "Available Gigs" section, which lists two gigs:

- 1. New Year's Celebration**
Id: 621eb2b856780a03e8db638
On: 2022-12-31
Located at:
345 Market St., ...
Description: Ring in the New Year
Contact Info: 345-678-9012
Instruments Requested:
 - Piano

[Remove] [Delete Gig]
- 2. Memorial Day Parade**
Id: 621fb1f56780a03e8db6381
On: 2022-05-28
Located at:
Union Square, ...
Description: Play Taps at Each War Memorial
Contact Info: 123-456-7890
Instruments Requested:
 - Trumpet

[Remove] [Delete Gig]

A purple box also highlights the "Enter a New Gig" form at the bottom, which contains fields for Name of Gig (set to "Wedding"), Date (05/21/2022), Duration (1 hour), Address (362 Gettysburg Pike), City (Mechanicsburg), State (PA), Phone (123-456-7890), Description ("Play Christian Music D!"), and an "Add Gig" button.

After Add Gig button
is pushed:

The screenshot shows the same web browser window at "localhost:3000/gigs". The "Available Gigs" section now includes a third gig:

- 1. New Year's Celebration**
Id: 621eb2b856780a03e8db638
On: 2022-12-31
Located at:
345 Market St., ...
Description: Ring in the New Year
Contact Info: 345-678-9012
Instruments Requested:
 - Piano

[Remove] [Delete Gig]
- 2. Memorial Day Parade**
Id: 621fb1f56780a03e8db6381
On: 2022-05-28
Located at:
Union Square, ...
Description: Play Taps at Each War Memorial
Contact Info: 123-456-7890
Instruments Requested:
 - Trumpet

[Remove] [Delete Gig]
- 3. Wedding**
Id: 62264a716f047803e8a11112
On: 2022-05-21
Located at:
362 Gettysburg Pike, Mechanicsburg, PA.
Description: Play Christian Music During the Wedding Ceremony at DayBreak Church
Contact Info: 123-456-7890
Instruments Requested:
[Delete Gig]

The "Enter a New Gig" form is still visible at the bottom of the page.



PROMINEO TECH

Add an instrument to the Wedding Gig (Update):

The screenshot shows two views of the Find-A-Gig-App interface.

Instruments Page: The URL is `localhost:3000/instruments`. It displays a list of available instruments under three categories: Cello, Strings; Piano, Percussion; and Trumpet, Bass. Each category has a form to add a new instrument. A modal window titled "Enter a New Instrument" is open, showing fields for "Instrument Name" and "Instrument Section", with an "Add Instrument" button.

Gigs Page: The URL is `localhost:3000/gigs`. It lists three gigs: 1. New Year's Celebration, 2. Memorial Day Parade, and 3. Wedding. Each gig card includes details like date, address, contact info, and instruments requested. A modal window titled "Enter a New Gig" is open, showing fields for "Name of Gig", "Address", "Phone", and "Description", with an "Add Gig" button.

curl -X POST http://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs

```
[{"_id": "621eb2b856780a03e8ddb638", "name": "New Year's Celebration", "date": "2022-12-31", "address": "345 Market St", "phone": "345-678-9012", "description": "Ring in the New Year", "instruments": [{"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}]}, {"_id": "621f8f1f56780a03e8ddb681", "name": "Memorial Day Parade", "date": "2022-05-28", "address": "Union Square", "phone": "123-456-7890", "description": "Play Taps at Each War Memorial", "instruments": [{"_id": "6226c0db6f047803e8ae0d5b", "name": "Trumpet", "section": "Brass"}]}, {"_id": "622d4a716f047803e8ae1112", "name": "Wedding", "date": "2022-05-21", "address": "362 Gettysburg Pike", "phone": "123-456-7890", "description": "Play Christian Music During the Wedding Ceremony at DayBreak Church", "instruments": [{"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}]}]
```



PROMINEO TECH

Add an instrument to User (**Update**): Jiminy Cricket

The screenshot shows the 'Instruments' section of the application. It displays three categories of instruments: Cello/Strings, Piano/Percussion, and Trumpet/Brass. Each category has a form to add a new instrument. A modal dialog at the bottom allows entering a new instrument name and section.

The screenshot shows the 'Users' section of the application. It displays a list of available musicians. A modal dialog at the bottom allows entering a new musician's information.

The screenshot shows a browser terminal displaying the JSON response of the user list. The response includes the user's ID, first name, last name, address, phone number, and instruments.

```
[{"_id": "621eeef656780a03e8ddb659", "firstname": "Donald", "lastname": "Duck", "address": "123 Main St.", "city": "Nashua", "state": "NH", "phone": "987-654-3210", "instruments": [{"_id": "62144e8d56780a03e8ddae90", "name": "Cello", "section": "Strings"}]}, {"_id": "6226b4c06f047803e8ae0d57", "firstname": "Minnie ", "lastname": "Mouse", "address": "345 Main St.", "city": "Disneyland", "state": "CA", "phone": "987-654-3210", "instruments": [{"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}]}, {"_id": "622d49caf047803e8ae1111", "firstname": "Jiminy", "lastname": "Cricket", "address": "987 Poplar Lane", "city": "Boston", "state": "MA", "phone": "715-908-7654", "instruments": [{"_id": "6226c0db6f047803e8ae0d5b", "name": "Trumpet", "section": "Brass"}]}]
```



PROMINEO TECH

Push **Delete Gig** on the Wedding Gig:

The screenshot shows a web browser window titled "Find-A-Gig-App" at "localhost:3000/gigs". The page has a header with "Find-A-Gig" and navigation links for Home, Users, Gigs, and Instruments. The main content area is titled "Available Gigs" and lists two gigs:

- 1. New Year's Celebration**
Id: 621eb2b856780a03e8ddb638
On: 2022-12-31
Located at:
345 Market St., ..

Description: Ring in the New Year
Contact Info: 345-678-9012
Instruments Requested:
 - Piano[Remove](#)
[Delete Gig](#)
- 2. Memorial Day Parade**
Id: 621fbff1f56780a03e8ddb681
On: 2022-05-28
Located at:
Union Square, ..

Description: Play Taps at Each War Memorial
Contact Info: 123-456-7890
Instruments Requested:
 - Trumpet[Remove](#)
[Delete Gig](#)

Below the gigs is a form titled "Enter a New Gig" with fields for Name of Gig, Address, City, State, Phone, and Description, along with an "Add Gig" button.

The screenshot shows a browser window with multiple tabs and an open developer tools Network tab. The URL is "crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs". The Network tab displays a JSON response:

```
[{"_id": "621eb2b856780a03e8ddb638", "name": "New Year's Celebration", "date": "2022-12-31", "address": "345 Market St.", "phone": "345-678-9012", "description": "Ring in the New Year", "instruments": [{"_id": "6226bad96f047803e8ae0d58", "name": "Piano", "section": "Percussion"}]}, {"_id": "621fbff1f56780a03e8ddb681", "name": "Memorial Day Parade", "date": "2022-05-28", "address": "Union Square", "phone": "123-456-7890", "description": "Play Taps at Each War Memorial", "instruments": [{"_id": "6226c0db6f047803e8ae0d5b", "name": "Trumpet", "section": "Brass"}]}]
```



Pushed Delete Instrument on New Year's Celebration Gig:

Find-A-Gig App

localhost:3000/gigs

Available Gigs

1. New Year's Celebration
Id: 621eb2b856780a03e8ddb638
On: 2022-12-31
Located at:
345 Market St., ..
Description: Ring in the New Year
Contact Info: 345-678-9012
Instruments Requested:
Delete Gig

2. Memorial Day Parade
Id: 621fbff1f56780a03e8ddb681
On: 2022-05-28
Located at:
Union Square, ..
Description: Play Taps at Each War Memorial
Contact Info: 123-456-7890
Instruments Requested:
• Trumpet
Remove
Delete Gig

Enter a New Gig

Name of Gig mm/dd/yyyy Duration
Address City State
Phone
Description
Add Gig

Crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/gigs

```
[{"_id": "621eb2b856780a03e8ddb638", "name": "New Year's Celebration", "date": "2022-12-31", "address": "345 Market St", "phone": "345-678-9012", "description": "Ring in the New Year", "instruments": []}, {"_id": "621fbff1f56780a03e8ddb681", "name": "Memorial Day Parade", "date": "2022-05-28", "address": "Union Square", "phone": "123-456-7890", "description": "Play Taps at Each War Memorial", "instruments": []}, {"_id": "6226c0db6f047803e8ae0d5b", "name": "Trumpet", "section": "Brass"}]
```