

```
1. # Jobify
2.
3. ##### Track Your Job Search
4.
5. Project in Action - [Jobify](https://www.jobify.live/)
6.
7. ##### Support
8.
9. Find the App Useful? [You can always buy me a coffee](https://www.buymeacoffee.com/johnsmilga)
10.
11. ##### Run The App Locally
12.
13. ```sh
14. npm run install-dependencies
15. ```
16.
17. - rename .env.temp to .env
18. - setup values for - MONGO_URL, JWT_SECRET, JWT_LIFETIME
19.
20. ```sh
21. npm start
22. ```
23.
24. - visit url http://localhost:3000/
25.
26. ##### Setup React App
27.
28. - create <b>client</b> folder
29. - open terminal
30.
31. ```sh
32. cd client
33. ```
34.
35. ```sh
36. npx create-react-app .
37. ```
38.
39. ```sh
40. npm start
41. ```
42.
43. - set editor/browser side by side
44. - copy/paste assets from complete project
45.
46. ##### Spring Cleaning
47.
48. - in src remove
49. - App.css
50. - App.test.js
51. - logo.svg
52. - reportWebVitals.js
53. - setupTests.js
54. - fix App.js and index.js
55.
56. ##### Title and Favicon
57.
58. - change title in public/index.html
59. - replace favicon.ico in public
60. - resource [Generate Favicons](https://favicon.io/)
61.
```

```

62. ##### Normalize.css and Global Styles
63.
64. - CSS in JS (styled-components)
65. - saves times on the setup
66. - less lines of css
67. - speeds up the development
68. - normalize.css
69. - small CSS file that provides cross-browser consistency in the default styling of HTML
    elements.
70. - [normalize docs](https://necolas.github.io/normalize.css/)
71.
72. ```sh
73. npm install normalize.css
74. ```
75.
76. - import 'normalize.css' in index.js
77. - SET BEFORE 'index.css'
78. - replace contents of index.css
79. - if any questions about normalize or specific styles
80. - Coding Addict - [Default Starter Video](https://youtu.be/UDdyGNlQK5w)
81. - Repo - [Default Starter Repo](https://github.com/john-smilga/default-starter)
82.
83. ##### Landing Page
84.
85. - zoom level 175%
86. - markdown preview extension
87. - get something on the screen
88. - react router and styled components right after
89. - create pages directory in the source
90. - for now Landing.js
91. - create component (snippets extension)
92. - setup basic return
93.
94. ```js
95. <h4>Landing Page</h4>
96. ```
97.
98. - import logo.svg and main.svg
99. - import Landing in App.js and render
100.
101. ##### Styled Components
102.
103. - CSS in JS
104. - Styled Components
105. - have logic and styles in component
106. - no name collisions
107. - apply javascript logic
108. - [Styled Components Docs](https://styled-components.com/)
109. - [Styled Components Course](https://www.udemy.com/course/styled-components-tutorial-
    and-project-course/?referralCode=9DABB172FCB2625B663F)
110.
111. ```sh
112. npm install styled-components
113. ```
114.
115. ```js
116. import styled from "styled-components";
117.
118. const El = styled.el`
119.   // styles go here
120. `;

```

```

121.     ```
122.
123.     - no name collisions, since unique class
124.     - vscode-styled-components extension
125.     - colors and bugs
126.     - style entire react component
127.
128.     ```js
129.     const Wrapper = styled.el``;
130.
131.     const Component = () => {
132.         return (
133.             <Wrapper>
134.                 <h1> Component</h1>
135.             </Wrapper>
136.         );
137.     };
138.     ```
139.
140.     - only responsible for styling
141.     - wrappers folder in assets
142.
143.     ##### Logo and Images
144.
145.     - logo built in Figma
146.     - [Cool Images](https://undraw.co/)
147.
148.     ##### Logo
149.
150.     - create <b>components</b> folder in source
151.     - create Logo.js
152.     - move import and image logic
153.     - export as default
154.     - utilize index.js
155.
156.     ##### React Router
157.
158.     - Version 6
159.     - [React Router Docs](https://reactrouter.com/docs/en/v6)
160.
161.     ```sh
162.     npm install history@5 react-router-dom@6
163.     ```
164.
165.     - import four components
166.
167.     ```js
168.     import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
169.     ```
170.
171.     - Connect to browser's URL with BrowserRouter
172.     - Routes instead of Switch
173.
174.     ```js
175.
176.     <BrowserRouter>
177.         <Routes>
178.             <Route path="/" element={<div>Dashboard</div>} />
179.             <Route path="/register" element={<div>Register</div>} />
180.             <Route path="/landing" element={<Landing />} />
181.             <Route path="*" element={<div>Error</div>}>

```

```

182.         </Routes>
183.     </BrowserRouter>
184.
185.     ```
186.
187.     ```js
188.     <nav>
189.         <Link to="/">Dashboard</Link>
190.         <Link to="/register">Register</Link>
191.         <Link to="/landing">Home</Link>
192.     </nav>
193.     ```
194.
195.     - go to Landing.js
196.
197.     ```js
198.     import { Link } from "react-router-dom";
199.
200.     return (
201.         <Link to="/register" className="btn btn-hero">
202.             Login / Register
203.         </Link>
204.     );
205.     ```
206.
207.     ##### Setup Pages
208.
209.     - create Error, Register, Dashboard pages
210.     - basic return
211.     - create index.js
212.     - import all the pages
213.     - export one by one
214.     - basically the same, as in components
215.     - import App.js
216.     - add to element={}
217.     - remove temp navbar
218.
219.     ##### Error Page
220.
221.     ```js
222.     import { Link } from "react-router-dom";
223.     import img from "../assets/images/not-found.svg";
224.     import Wrapper from "../assets/wrappers/ErrorMessage";
225.
226.     return (
227.         <Wrapper className="full-page">
228.             <div>
229.                 <img src={img} alt="not found" />
230.                 <h3>text</h3>
231.                 <p>text</p>
232.                 <Link to="/">back home</Link>
233.             </div>
234.         </Wrapper>
235.     );
236.     ```
237.
238.     ##### Auto Imports
239.
240.     - use while developing
241.     - only sparingly while recording
242.     - better picture

```

```

243.     - messes with flow
244.     - just my preference
245.     - still use them, just not all the time
246.
247.     ##### Register Page - Setup
248.
249.     - show preview in Browser and themes
250.
251.     ```js
252.     import { useState, useEffect } from "react";
253.     import { Logo } from "../components";
254.     import Wrapper from "../assets/wrappers/RegisterPage";
255.     // global context and useNavigate later
256.
257.     const initialState = {
258.       name: "",
259.       email: "",
260.       password: "",
261.       isMember: true,
262.     };
263.     // if possible prefer local state
264.     // global state
265.
266.     function Register() {
267.       const [values, setValues] = useState(initialState);
268.
269.       // global context and useNavigate later
270.
271.       const handleChange = (e) => {
272.         console.log(e.target);
273.       };
274.
275.       const onSubmit = (e) => {
276.         e.preventDefault();
277.         console.log(e.target);
278.       };
279.       return (
280.         <Wrapper className="full-page">
281.           <form className="form" onSubmit={onSubmit}>
282.             <Logo />
283.             <h3>Login</h3>
284.
285.             {/* name field */}
286.             <div className="form-row">
287.               <label htmlFor="name" className="form-label">
288.                 name
289.               </label>
290.
291.               <input
292.                 type="text"
293.                 value={values.name}
294.                 name="name"
295.                 onChange={handleChange}
296.                 className="form-input"
297.               />
298.             </div>
299.
300.             <button type="submit" className="btn btn-block">
301.               submit
302.             </button>
303.           </form>

```

```

304.         </Wrapper>
305.     );
306. }
307. ...
308.
309. ##### FormRow Component
310.
311. - create FormRow.js in <b>components</b>
312. - setup import/export
313. - setup one for email and password
314. - hint "type,name,value"
315.
316. ```js
317. const FormRow = ({ type, name, value, handleChange, labelText }) => {
318.     return (
319.         <div className="form-row">
320.             <label htmlFor={name} className="form-label">
321.                 {labelText || name}
322.             </label>
323.
324.             <input
325.                 type={type}
326.                 value={value}
327.                 name={name}
328.                 onChange={handleChange}
329.                 className="form-input"
330.             />
331.         </div>
332.     );
333. };
334.
335. export default FormRow;
336. ...
337.
338. ##### Alert Component
339.
340. - right away setup as component
341. - create Alert.js in <b>components</b>
342.
343. ```js
344. const Alert = () => {
345.     return <div className="alert alert-danger">alert goes here</div>;
346. };
347.
348. export default Alert;
349. ...
350.
351. - setup import/export
352. - alert-danger or alert-success
353. - eventually setup in global context
354. - showAlert in initialState (true || false)
355. - right after h3 login
356.
357. ```js
358. values.showAlert && <Alert />;
359. ...
360.
361. ##### Toggle Member
362.
363. ```js
364. const toggleMember = () => {

```

```

365.     setValues({ ...values, isMember: !values.isMember });
366. };
367.
368. return (
369.     <Wrapper>
370.         {/* control h3 */}
371.
372.         <h3>{values.isMember ? "Login" : "Register"}</h3>
373.
374.         {/* toggle name */}
375.
376.         {!values.isMember && (
377.             <FormRow
378.                 type="text"
379.                 name="name"
380.                 value={values.name}
381.                 handleChange={handleChange}
382.             />
383.         )}
384.
385.         {/* right after submit btn */}
386.         {/* toggle button */}
387.
388.         <p>
389.             {values.isMember ? "Not a member yet?" : "Already a member?"}
390.
391.             <button type="button" onClick={toggleMember} className="member-btn">
392.                 {values.isMember ? "Register" : "Login"}
393.             </button>
394.         </p>
395.     </Wrapper>
396. );
397. ...
398.
399. ##### Global Context
400.
401. - in src create <b>context</b> directory
402. - actions.js
403. - reducer.js
404. - appContext.js
405.
406. ```js
407. import React, { useState, useReducer, useContext } from "react";
408.
409. export const initialState = {
410.     isLoading: false,
411.     showAlert: false,
412.     alertText: "",
413.     alertType: "",
414. };
415. const AppContext = React.createContext();
416. const AppProvider = ({ children }) => {
417.     const [state, setState] = useState(initialState);
418.
419.     return (
420.         <AppContext.Provider
421.             value={{
422.                 ...state,
423.             }}
424.         >
425.             {children}

```

```

426.         </AppContext.Provider>
427.     );
428. };
429. // make sure use
430. export const useAppContext = () => {
431.     return useContext(AppContext);
432. };
433.
434. export { AppProvider };
435. ```
436.
437. - index.js
438.
439. ```js
440. import { AppProvider } from "../context/appContext";
441.
442. ReactDOM.render(
443.     <React.StrictMode>
444.         <AppProvider>
445.             <App />
446.         </AppProvider>
447.     </React.StrictMode>,
448.     document.getElementById("root")
449. );
450. ```
451.
452. - Register.js
453.
454. ```js
455. import { useAppContext } from "../context/appContext";
456.
457. const { isLoading, showAlert } = useAppContext();
458. ```
459.
460. - switch to global showAlert
461.
462. ##### useReducer
463.
464. - [React Tutorial](https://youtu.be/iZhV0bILFb0)
465. - useReducer vs Redux
466. - multiple reducers vs one
467.
468. ##### Wire Up Reducer
469.
470. ```js
471. reducer.js;
472.
473. const reducer = (state, action) => {
474.     throw new Error(`no such action :${action.type}`);
475. };
476. export default reducer;
477. ```
478.
479. ```js
480. appContext.js;
481.
482. import reducer from "../reducer";
483.
484. const [state, dispatch] = useReducer(reducer, initialState);
485. ```
486.

```



```

487.     #### Display Alert
488.
489.     ```js
490.     actions.js;
491.
492.     export const DISPLAY_ALERT = "SHOW_ALERT";
493.     ```
494.
495.     - setup imports (reducer and appContext)
496.
497.     ```js
498.     appContext.js
499.
500.     const displayAlert() =>{
501.         dispatch({type:DISPLAY_ALERT})
502.     }
503.
504.     ```
505.
506.     ```js
507.     reducer.js;
508.
509.     if (action.type === DISPLAY_ALERT) {
510.         return {
511.             ...state,
512.             showAlert: true,
513.             alertType: "danger",
514.             alertText: "Please provide all values!",
515.         };
516.     }
517.     ```
518.
519.     ```js
520.     Alert.js in Components;
521.
522.     import { useAppContext } from "../context/appContext";
523.
524.     const Alert = () => {
525.         const { alertType, alertText } = useAppContext();
526.         return <div className={`alert alert-${alertType}`}>{alertText}</div>;
527.     };
528.     ```
529.
530.     #### Display Alert
531.
532.     - [JS Nuggets - Dynamic Object Keys](https://youtu.be/_qxCYtWm0tw)
533.
534.     ```js
535.     appContext.js;
536.
537.     const handleChange = (e) => {
538.         setValues({ ...values, [e.target.name]: e.target.value });
539.     };
540.     ```
541.
542.     - get displayAlert function
543.
544.     ```js
545.     appContext.js;
546.
547.     const onSubmit = (e) => {

```

```

548.     e.preventDefault();
549.     const { name, email, password, isMember } = values;
550.     if (!email || !password || (!isMember && !name)) {
551.         displayAlert();
552.         return;
553.     }
554.     console.log(values);
555. };
556. ```
557.
558. ##### Clear Alert
559.
560. - technically optional
561.
562. ```js
563. actions.js;
564.
565. export const CLEAR_ALERT = "CLEAR_ALERT";
566. ```
567.
568. - setup imports (reducer and appContext)
569.
570. ```js
571. reducer.js;
572.
573. if (action.type === CLEAR_ALERT) {
574.     return {
575.         ...state,
576.         showAlert: false,
577.         alertType: "",
578.         alertText: "",
579.     };
580. }
581. ```
582.
583. ```js
584. appContext.js;
585.
586. const displayAlert = () => {
587.     dispatch({
588.         type: DISPLAY_ALERT,
589.     });
590.     clearAlert();
591. };
592.
593. const clearAlert = () => {
594.     setTimeout(() => {
595.         dispatch({
596.             type: CLEAR_ALERT,
597.         });
598.     }, 3000);
599. };
600. ```
601.
602. ##### Setup Server
603.
604. - stop the dev server in client
605. - cd ..
606. - start setting up our server
607. - setup package.json
608.

```

```
609.     ``sh
610.     npm init -y
611.     ``
612.
613.     - create server.js
614.     - console.log('server running...')
615.
616.     ``sh
617.     node server
618.     ``
619.
620.     #### ES6 vs CommonJS
621.
622.     ``js
623.     CommonJS;
624.
625.     const express = require("express");
626.     const app = express();
627.     ``
628.
629.     ``js
630.     ES6;
631.
632.     import express from "express";
633.     const app = express();
634.     ``
635.
636.     - file extension .mjs
637.
638.     ``js
639.     package.json
640.
641.     "type":"module"
642.     ``
643.
644.     #### Nodemon and Basic Express Server
645.
646.     ``sh
647.     npm install nodemon --save-dev
648.     ``
649.
650.     ``js
651.     package.json
652.
653.     "start":"nodemon server"
654.     ``
655.
656.
657.     ``sh
658.     npm install express
659.     ``
660.
661.     ``js
662.     import express from "express";
663.     const app = express();
664.
665.     app.get("/", (req, res) => {
666.         res.send("Welcome!");
667.     });
668.
669.     const port = process.env.PORT || 5000;
```

```
670.
671.     app.listen(port, () => console.log(`Server is listening on port ${port}...`));
672.     ```
673.
674.     # Jobify
675.
676.     ##### Track Your Job Search
677.
678.     Project in Action - [Jobify](https://www.jobify.live/)
679.
680.     ##### Support
681.
682.     Find the App Useful? [You can always buy me a
683.     coffee](https://www.buymeacoffee.com/johnsmilga)
684.
685.     ##### Run The App Locally
686.     ```sh
687.     npm run install-dependencies
688.     ```
689.
690.     - rename .env.temp to .env
691.     - setup values for - MONGO_URL, JWT_SECRET, JWT_LIFETIME
692.
693.     ```sh
694.     npm start
695.     ```
696.
697.     - visit url http://localhost:3000/
698.
699.     ##### Setup React App
700.
701.     - create <b>client</b> folder
702.     - open terminal
703.
704.     ```sh
705.     cd client
706.     ```
707.
708.     ```sh
709.     npx create-react-app .
710.     ```
711.
712.     ```sh
713.     npm start
714.     ```
715.
716.     - set editor/browser side by side
717.     - copy/paste assets from complete project
718.
719.     ##### Spring Cleaning
720.
721.     - in src remove
722.     - App.css
723.     - App.test.js
724.     - logo.svg
725.     - reportWebVitals.js
726.     - setupTests.js
727.     - fix App.js and index.js
728.
729.     ##### Title and Favicon
```

```
730.
731.     - change title in public/index.html
732.     - replace favicon.ico in public
733.     - resource [Generate Favicons](https://favicon.io/)
734.
735.     #### Normalize.css and Global Styles
736.
737.     - CSS in JS (styled-components)
738.     - saves times on the setup
739.     - less lines of css
740.     - speeds up the development
741.     - normalize.css
742.     - small CSS file that provides cross-browser consistency in the default styling of HTML
    elements.
743.     - [normalize docs](https://necolas.github.io/normalize.css/)
744.
745.     ```sh
746.     npm install normalize.css
747.     ```
748.
749.     - import 'normalize.css' in index.js
750.     - SET BEFORE 'index.css'
751.     - replace contents of index.css
752.     - if any questions about normalize or specific styles
753.     - Coding Addict - [Default Starter Video](https://youtu.be/UDdyGN1QK5w)
754.     - Repo - [Default Starter Repo](https://github.com/john-smilga/default-starter)
755.
756.     #### Landing Page
757.
758.     - zoom level 175%
759.     - markdown preview extension
760.     - get something on the screen
761.     - react router and styled components right after
762.     - create pages directory in the source
763.     - for now Landing.js
764.     - create component (snippets extension)
765.     - setup basic return
766.
767.     ```js
768.     <h4>Landing Page</h4>
769.     ```
770.
771.     - import logo.svg and main.svg
772.     - import Landing in App.js and render
773.
774.     #### Styled Components
775.
776.     - CSS in JS
777.     - Styled Components
778.     - have logic and styles in component
779.     - no name collisions
780.     - apply javascript logic
781.     - [Styled Components Docs](https://styled-components.com/)
782.     - [Styled Components Course](https://www.udemy.com/course/styled-components-tutorial-
    and-project-course/?referralCode=9DABB172FCB2625B663F)
783.
784.     ```sh
785.     npm install styled-components
786.     ```
787.
788.     ```js
```

```

789.     import styled from "styled-components";
790.
791.     const El = styled.el`
792.         // styles go here
793.     `;
794.
795.
796.     - no name collisions, since unique class
797.     - vscode-styled-components extension
798.     - colors and bugs
799.     - style entire react component
800.
801.     ```js
802.     const Wrapper = styled.el``;
803.
804.     const Component = () => {
805.         return (
806.             <Wrapper>
807.                 <h1> Component</h1>
808.             </Wrapper>
809.         );
810.     };
811.
812.
813.     - only responsible for styling
814.     - wrappers folder in assets
815.
816.     ##### Logo and Images
817.
818.     - logo built in Figma
819.     - [Cool Images](https://undraw.co/)
820.
821.     ##### Logo
822.
823.     - create <b>components</b> folder in source
824.     - create Logo.js
825.     - move import and image logic
826.     - export as default
827.     - utilize index.js
828.
829.     ##### React Router
830.
831.     - Version 6
832.     - [React Router Docs](https://reactrouter.com/docs/en/v6)
833.
834.     ```sh
835.     npm install history@5 react-router-dom@6
836.
837.
838.     - import four components
839.
840.     ```js
841.     import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
842.
843.
844.     - Connect to browser's URL with BrowserRouter
845.     - Routes instead of Switch
846.
847.     ```js
848.
849.     <BrowserRouter>

```

```

850.         <Routes>
851.             <Route path="/" element={<div>Dashboard</div>} />
852.             <Route path="/register" element={<div>Register</div>} />
853.             <Route path="/landing" element={<Landing />} />
854.             <Route path="*" element={<div>Error</div>}>
855.         </Routes>
856.     </BrowserRouter>
857.
858.     ...
859.
860.     ```js
861.     <nav>
862.         <Link to="/">Dashboard</Link>
863.         <Link to="/register">Register</Link>
864.         <Link to="/landing">Home</Link>
865.     </nav>
866.     ...
867.
868.     - go to Landing.js
869.
870.     ```js
871.     import { Link } from "react-router-dom";
872.
873.     return (
874.         <Link to="/register" className="btn btn-hero">
875.             Login / Register
876.         </Link>
877.     );
878.     ...
879.
880.     ##### Setup Pages
881.
882.     - create Error, Register, Dashboard pages
883.     - basic return
884.     - create index.js
885.     - import all the pages
886.     - export one by one
887.     - basically the same, as in components
888.     - import App.js
889.     - add to element={}
890.     - remove temp navbar
891.
892.     ##### Error Page
893.
894.     ```js
895.     import { Link } from "react-router-dom";
896.     import img from "../assets/images/not-found.svg";
897.     import Wrapper from "../assets/wrappers/ErrorMessage";
898.
899.     return (
900.         <Wrapper className="full-page">
901.             <div>
902.                 <img src={img} alt="not found" />
903.                 <h3>text</h3>
904.                 <p>text</p>
905.                 <Link to="/">back home</Link>
906.             </div>
907.         </Wrapper>
908.     );
909.     ...
910.

```

```

911.     #### Auto Imports
912.
913.     - use while developing
914.     - only sparingly while recording
915.     - better picture
916.     - messes with flow
917.     - just my preference
918.     - still use them, just not all the time
919.
920.     #### Register Page - Setup
921.
922.     - show preview in Browser and themes
923.
924.     ```js
925.     import { useState, useEffect } from "react";
926.     import { Logo } from "../components";
927.     import Wrapper from "../assets/wrappers/RegisterPage";
928.     // global context and useNavigate later
929.
930.     const initialState = {
931.       name: "",
932.       email: "",
933.       password: "",
934.       isMember: true,
935.     };
936.     // if possible prefer local state
937.     // global state
938.
939.     function Register() {
940.       const [values, setValues] = useState(initialState);
941.
942.       // global context and useNavigate later
943.
944.       const handleChange = (e) => {
945.         console.log(e.target);
946.       };
947.
948.       const onSubmit = (e) => {
949.         e.preventDefault();
950.         console.log(e.target);
951.       };
952.       return (
953.         <Wrapper className="full-page">
954.           <form className="form" onSubmit={onSubmit}>
955.             <Logo />
956.             <h3>Login</h3>
957.
958.             {/* name field */}
959.             <div className="form-row">
960.               <label htmlFor="name" className="form-label">
961.                 name
962.               </label>
963.
964.               <input
965.                 type="text"
966.                 value={values.name}
967.                 name="name"
968.                 onChange={handleChange}
969.                 className="form-input"
970.               />
971.             </div>

```



```

972.
973.         <button type="submit" className="btn btn-block">
974.             submit
975.         </button>
976.     </form>
977. </Wrapper>
978. );
979. }
980. ```
981.
982. ##### FormRow Component
983.
984. - create FormRow.js in <b>components</b>
985. - setup import/export
986. - setup one for email and password
987. - hint "type,name,value"
988.
989. ```js
990. const FormRow = ({ type, name, value, handleChange, labelText }) => {
991.     return (
992.         <div className="form-row">
993.             <label htmlFor={name} className="form-label">
994.                 {labelText || name}
995.             </label>
996.
997.             <input
998.                 type={type}
999.                 value={value}
1000.                 name={name}
1001.                 onChange={handleChange}
1002.                 className="form-input"
1003.             />
1004.         </div>
1005.     );
1006. };
1007.
1008. export default FormRow;
1009. ```
1010.
1011. ##### Alert Component
1012.
1013. - right away setup as component
1014. - create Alert.js in <b>components</b>
1015.
1016. ```js
1017. const Alert = () => {
1018.     return <div className="alert alert-danger">alert goes here</div>;
1019. };
1020.
1021. export default Alert;
1022. ```
1023.
1024. - setup import/export
1025. - alert-danger or alert-success
1026. - eventually setup in global context
1027. - showAlert in initialState (true || false)
1028. - right after h3 login
1029.
1030. ```js
1031. values.showAlert && <Alert />;
1032. ```

```

```

1033.
1034.     ##### Toggle Member
1035.
1036.     ```js
1037.     const toggleMember = () => {
1038.       setValues({ ...values, isMember: !values.isMember });
1039.     };
1040.
1041.     return (
1042.       <Wrapper>
1043.         {/* control h3 */}
1044.
1045.         <h3>{values.isMember ? "Login" : "Register"}</h3>
1046.
1047.         {/* toggle name */}
1048.
1049.         {!values.isMember && (
1050.           <FormRow
1051.             type="text"
1052.             name="name"
1053.             value={values.name}
1054.             handleChange={handleChange}
1055.           />
1056.         )}
1057.
1058.         {/* right after submit btn */}
1059.         {/* toggle button */}
1060.
1061.         <p>
1062.           {values.isMember ? "Not a member yet?" : "Already a member?"}
1063.
1064.           <button type="button" onClick={toggleMember} className="member-btn">
1065.             {values.isMember ? "Register" : "Login"}
1066.           </button>
1067.         </p>
1068.       </Wrapper>
1069.     );
1070.     ```
1071.
1072.     ##### Global Context
1073.
1074.     - in src create <b>context</b> directory
1075.     - actions.js
1076.     - reducer.js
1077.     - appContext.js
1078.
1079.     ```js
1080.     import React, { useState, useReducer, useContext } from "react";
1081.
1082.     export const initialState = {
1083.       isLoading: false,
1084.       showAlert: false,
1085.       alertText: "",
1086.       alertType: "",
1087.     };
1088.     const AppContext = React.createContext();
1089.     const AppProvider = ({ children }) => {
1090.       const [state, setState] = useState(initialState);
1091.
1092.       return (
1093.         <AppContext.Provider

```

```

1094.         value={{
1095.             ...state,
1096.         }}
1097.     >
1098.         {children}
1099.     </AppContext.Provider>
1100. );
1101. };
1102. // make sure use
1103. export const useAppContext = () => {
1104.     return useContext(AppContext);
1105. };
1106.
1107. export { AppProvider };
1108. ```
1109.
1110. - index.js
1111.
1112. ```js
1113. import { AppProvider } from "../context/appContext";
1114.
1115. ReactDOM.render(
1116.     <React.StrictMode>
1117.         <AppProvider>
1118.             <App />
1119.         </AppProvider>
1120.     </React.StrictMode>,
1121.     document.getElementById("root")
1122. );
1123. ```
1124.
1125. - Register.js
1126.
1127. ```js
1128. import { useAppContext } from "../context/appContext";
1129.
1130. const { isLoading, showAlert } = useAppContext();
1131. ```
1132.
1133. - switch to global showAlert
1134.
1135. ##### useReducer
1136.
1137. - [React Tutorial](https://youtu.be/iZhV0bILFb0)
1138. - useReducer vs Redux
1139. - multiple reducers vs one
1140.
1141. ##### Wire Up Reducer
1142.
1143. ```js
1144. reducer.js;
1145.
1146. const reducer = (state, action) => {
1147.     throw new Error(`no such action :${action.type}`);
1148. };
1149. export default reducer;
1150. ```
1151.
1152. ```js
1153. appContext.js;
1154.

```

```

1155.     import reducer from "./reducer";
1156.
1157.     const [state, dispatch] = useReducer(reducer, initialState);
1158.     ```
1159.
1160.     ##### Display Alert
1161.
1162.     ```js
1163.     actions.js;
1164.
1165.     export const DISPLAY_ALERT = "SHOW_ALERT";
1166.     ```
1167.
1168.     - setup imports (reducer and appContext)
1169.
1170.     ```js
1171.     appContext.js
1172.
1173.     const displayAlert() =>{
1174.         dispatch({type:DISPLAY_ALERT})
1175.     }
1176.
1177.     ```
1178.
1179.     ```js
1180.     reducer.js;
1181.
1182.     if (action.type === DISPLAY_ALERT) {
1183.         return {
1184.             ...state,
1185.             showAlert: true,
1186.             alertType: "danger",
1187.             alertText: "Please provide all values!",
1188.         };
1189.     }
1190.     ```
1191.
1192.     ```js
1193.     Alert.js in Components;
1194.
1195.     import { useAppContext } from "../context/appContext";
1196.
1197.     const Alert = () => {
1198.         const { alertType, alertText } = useAppContext();
1199.         return <div className={`alert alert-${alertType}`}>{alertText}</div>;
1200.     };
1201.     ```
1202.
1203.     ##### Display Alert
1204.
1205.     - [JS Nuggets - Dynamic Object Keys](https://youtu.be/_qxCYtWm0tw)
1206.
1207.     ```js
1208.     appContext.js;
1209.
1210.     const handleChange = (e) => {
1211.         setValues({ ...values, [e.target.name]: e.target.value });
1212.     };
1213.     ```
1214.
1215.     - get displayAlert function

```

```

1216.
1217.   ```js
1218.   appContext.js;
1219.
1220.   const onSubmit = (e) => {
1221.     e.preventDefault();
1222.     const { name, email, password, isMember } = values;
1223.     if (!email || !password || (!isMember && !name)) {
1224.       displayAlert();
1225.       return;
1226.     }
1227.     console.log(values);
1228.   };
1229.   ```
1230.
1231.   ##### Clear Alert
1232.
1233.   - technically optional
1234.
1235.   ```js
1236.   actions.js;
1237.
1238.   export const CLEAR_ALERT = "CLEAR_ALERT";
1239.   ```
1240.
1241.   - setup imports (reducer and appContext)
1242.
1243.   ```js
1244.   reducer.js;
1245.
1246.   if (action.type === CLEAR_ALERT) {
1247.     return {
1248.       ...state,
1249.       showAlert: false,
1250.       alertType: "",
1251.       alertText: "",
1252.     };
1253.   }
1254.   ```
1255.
1256.   ```js
1257.   appContext.js;
1258.
1259.   const displayAlert = () => {
1260.     dispatch({
1261.       type: DISPLAY_ALERT,
1262.     });
1263.     clearAlert();
1264.   };
1265.
1266.   const clearAlert = () => {
1267.     setTimeout(() => {
1268.       dispatch({
1269.         type: CLEAR_ALERT,
1270.       });
1271.     }, 3000);
1272.   };
1273.   ```
1274.
1275.   ##### Setup Server
1276.

```

```
1277.     - stop the dev server in client
1278.     - cd ..
1279.     - start setting up our server
1280.     - setup package.json
1281.
1282.     ```sh
1283.     npm init -y
1284.     ```
1285.
1286.     - create server.js
1287.     - console.log('server running...')
1288.
1289.     ```sh
1290.     node server
1291.     ```
1292.
1293.     ##### ES6 vs CommonJS
1294.
1295.     ```js
1296.     CommonJS;
1297.
1298.     const express = require("express");
1299.     const app = express();
1300.     ```
1301.
1302.     ```js
1303.     ES6;
1304.
1305.     import express from "express";
1306.     const app = express();
1307.     ```
1308.
1309.     - file extension .mjs
1310.
1311.     ```js
1312.     package.json
1313.
1314.     "type":"module"
1315.     ```
1316.
1317.     ##### Nodemon and Basic Express Server
1318.
1319.     ```sh
1320.     npm install nodemon --save-dev
1321.     ```
1322.
1323.     ```js
1324.     package.json
1325.
1326.     "start":"nodemon server"
1327.
1328.     ```
1329.
1330.     ```sh
1331.     npm install express
1332.     ```
1333.
1334.     ```js
1335.     import express from "express";
1336.     const app = express();
1337.
```

```

1338.   app.get("/", (req, res) => {
1339.       res.send("Welcome!");
1340.   });
1341.
1342.   const port = process.env.PORT || 5000;
1343.
1344.   app.listen(port, () => console.log(`Server is listening on port ${port}...`));
1345.   ```
1346.
1347.   ##### Not Found Middleware
1348.
1349.   - in the root create <b>middleware</b> folder
1350.   - not-found.js
1351.   - setup function
1352.   - return 404 with message 'Route does not exist'
1353.   - import in server.js
1354.   - make sure to use .js extension
1355.   - place after home route
1356.
1357.   ##### Error Middleware
1358.
1359.   - in the middleware create error-handler.js
1360.   - setup function
1361.   - accept 4 parameters, first one error
1362.   - log error
1363.   - return 500
1364.   - json({msg: 'there was an error'})
1365.   - import in the server.js
1366.   - make sure to use .js extension
1367.   - place it last
1368.   - eventually handle Mongoose Errors, just like in the node-express
1369.   - showcase with async errors
1370.
1371.   ##### ENV Variables
1372.
1373.   ```sh
1374.   npm install dotenv
1375.   ```
1376.
1377.   - import dotenv from 'dotenv'
1378.   - dotenv.config()
1379.
1380.   - create .env
1381.   - PORT=4000
1382.   - .gitignore
1383.   - /node_modules
1384.   - .env
1385.
1386.   ##### Connect to MongoDB
1387.
1388.   - switched back to PORT=5000
1389.   - remove Error from '/'
1390.
1391.   - existing MongoDB Atlas Account
1392.
1393.   ```sh
1394.   npm install mongoose
1395.   ```
1396.
1397.   - create <b>db</b> folder
1398.   - create connect.js

```

```

1399. - setup connectDB(url)
1400. - in server.js create start() function
1401. - get connection string
1402. - setup as MONGO_URL in .env
1403. - provide credentials and DB Name
1404.
1405. ##### Auth Controller and Route Structure
1406.
1407. - create <b>controllers</b>
1408. - authController.js
1409. - create async functions
1410.
1411. ```js
1412. export { register, login, updateUser };
1413. ```
1414.
1415. - return res.send('function name')
1416. - create <b>routes</b> folder
1417. - authRoutes.js
1418. - setup express router
1419. - import functions from authController.js
1420.
1421. ```js
1422. router.route("/register").post(register);
1423. router.route("/login").post(login);
1424. router.route("/updateUser").patch(updateUser);
1425.
1426. export default router;
1427. ```
1428.
1429. - import authRouter in server.js
1430.
1431. ```js
1432. app.use("/api/v1/auth", authRouter);
1433. ```
1434.
1435. ##### Jobs Controller and Route Structure
1436.
1437. - jobsController.js
1438. - create async functions
1439.
1440. ```js
1441. export { createJob, deleteJob, getAllJobs, updateJob, showStats };
1442. ```
1443.
1444. - return res.send('function name')
1445.
1446. - jobsRoutes.js
1447. - setup express router
1448. - import functions from jobsController.js
1449.
1450. ```js
1451. router.route("/").post(createJob).get(getAllJobs);
1452. // place before :id
1453. router.route("/stats").get(showStats);
1454. router.route("/:id").delete(deleteJob).patch(updateJob);
1455.
1456. export default router;
1457. ```
1458.
1459. - in server.js jobsRouter

```



```
1460.
1461.   ```js
1462.   app.use("/api/v1/jobs", jobsRouter);
1463.   ```
1464.
1465.   ##### Postman
1466.
1467.   - URL global var
1468.   - JOBIFY Collection
1469.   - auth and jobs folders
1470.   - setup routes
1471.
1472.   ##### User Model
1473.
1474.   - <b>models</b> folder
1475.   - User.js
1476.   - setup schema
1477.   - name, email, password, lastName, location
1478.   - all {type:String}
1479.
1480.   ##### Validate Email
1481.
1482.   ```js
1483.   validate:{
1484.     validator:(field)=> {return 2 > 1},
1485.     message:'Please provide valid email'
1486.   }
1487.   ```
1488.
1489.   - [Validator Package](https://www.npmjs.com/package/validator)
1490.
1491.   ```sh
1492.   npm install validator
1493.   ```
1494.
1495.   - import in User.js
1496.   - validator.isEmail
1497.
1498.   ##### Register User - Initial Setup
1499.
1500.   - authController
1501.   - import User model
1502.   - setup temporary try/catch
1503.   - await User.create(req.body)
1504.   - if success 201 with json({user}) (temp)
1505.   - if error 500 with json({msg:'there was an error'})
1506.
1507.   ##### Pass Error to Error Handler
1508.
1509.   - next(error)
1510.
1511.   ##### Express-Async-Errors Package
1512.
1513.   - remove try/catch
1514.   - [Express-Async-Errors](https://www.npmjs.com/package/express-async-errors)
1515.
1516.   ```sh
1517.   npm install express-async-errors
1518.
1519.   ```
1520.
```

```
1521. - in server.js
1522. - import 'express-async-errors'
1523.
1524. - use throw Error('error') instead of next(error)
1525.
1526. ##### Http Status Codes
1527.
1528. - constants for status codes
1529. - personal preference
1530. - provides consistency
1531. - less bugs
1532. - easier to read/manage
1533.
1534. - [Http Status Codes](https://www.npmjs.com/package/http-status-codes)
1535.
1536. ```sh
1537. npm install http-status-codes
1538. ```
1539.
1540. - import/setup in authController and error-handler
1541. - setup defaultError
1542.
1543. ##### Custom Errors
1544.
1545. ##### Refactor Errors
1546.
1547. - create errors folder
1548. - create custom-api, bad-request, not-found, index.js files
1549. - add proper imports
1550. - setup index.js just like in the front-end
1551. - import {BadRequestError} in authController
1552. - gotcha "errors/index.js"
1553.
1554. ##### Hash Passwords
1555.
1556. - one way street, only compare hashed values
1557. - [bcrypt.js](https://www.npmjs.com/package/bcryptjs)
1558.
1559. ```sh
1560. npm install bcryptjs
1561. ```
1562.
1563. - User Model
1564. - import bcrypt from 'bcryptjs'
1565. - await genSalt(10)
1566. - await hash(password , salt)
1567. - await compare(requestPassword , currentPassword)
1568. - [mongoose middleware](https://mongoosejs.com/docs/middleware.html)
1569. - UserSchema.pre('save', async function(){
1570.   "this" points to instance created by UserSchema
1571. })
1572.
1573. ##### Mongoose - Custom Instance Methods
1574.
1575. [Custom Instance Methods](https://mongoosejs.com/docs/guide.html#methods)
1576.
1577. - UserSchema.methods.createJWT = function(){console.log(this)}
1578. - register controller
1579. - right after User.create()
1580. - invoke user.createJWT()
1581.
```

```

1582. ##### JWT
1583.
1584. - token
1585. - [jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken)
1586.
1587. ```sh
1588. npm install jsonwebtoken
1589. ```
1590.
1591. - User Model
1592. - import jwt from 'jsonwebtoken'
1593. - jwt.sign(payload,secret,options)
1594. - createJWT
1595.
1596. ```js
1597. return jwt.sign({ userId: this._id }, "jwtSecret", { expiresIn: "1d" });
1598. ```
1599.
1600. ```js
1601. return jwt.sign({ userId: this._id }, process.env.JWT_SECRET, {
1602.   expiresIn: process.env.JWT_LIFETIME,
1603. });
1604. ```
1605.
1606. ##### JWT_SECRET and JWT_LIFETIME
1607.
1608. - [Keys Generator](https://www.allkeysgenerator.com/)
1609. - RESTART SERVER!!!!
1610.
1611. ##### Complete Register
1612.
1613. - password : {select:false}
1614. - complete response
1615.
1616. ##### Concurrently
1617.
1618. - front-end and backend (server)
1619. - run separate terminals
1620. - [concurrently](https://www.npmjs.com/package/concurrently)
1621.
1622. ```sh
1623. npm install concurrently --save-dev
1624. ```
1625.
1626.
1627. - package.json
1628.
1629. ```js
1630. // --kill-others switch, all commands are killed if one dies
1631. // --prefix client - folder
1632. // cd client && npm start
1633. // escape quotes
1634.
1635. "scripts": {
1636.   "server": "nodemon server --ignore client",
1637.   "client": "npm start --prefix client",
1638.   "start": "concurrently --kill-others-on-fail \"npm run server\" \"npm run
client\""}
1639. },
1640. ```
1641.

```

```
1642. ##### Cors Error
1643.
1644. [Cors Error](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS)
1645.
1646. - two fixes (cors package and proxy)
1647.
1648. ##### Cors Package
1649.
1650. [cors package](https://www.npmjs.com/package/cors)
1651.
1652. ```sh
1653. npm install cors
1654. ```
1655.
1656. ```js
1657. import cors from "cors";
1658.
1659. app.use(cors());
1660. ```
1661.
1662. ##### Proxy
1663.
1664. - access from anywhere
1665. - don't want to use full url
1666.
1667. [cra proxy](https://create-react-app.dev/docs/proxying-api-requests-in-development/)
1668.
1669. ```js
1670. "proxy": "http://localhost:5000"
1671. ```
1672.
1673. - my preference to remove trailing slash /
1674. - restart app
1675.
1676. ##### Register User - Setup
1677.
1678. ```js
1679. appContext.js;
1680.
1681. const initialState = {
1682.   user: null,
1683.   token: null,
1684.   userLocation: "",
1685. };
1686. ```
1687.
1688. - actions.js REGISTER_USER_BEGIN, SUCCESS, ERROR
1689. - import reducer, appContext
1690.
1691. ```js
1692. appContext.js;
1693. const registerUser = async (currentUser) => {
1694.   console.log(currentUser);
1695. };
1696. ```
1697.
1698. - import in Register.js
1699.
1700. ```js
1701. Register.js;
1702.
```

```

1703.     const currentUser = { name, email, password };
1704.     if (isMember) {
1705.         console.log("already a member");
1706.     } else {
1707.         registerUser(currentUser);
1708.     }
1709.
1710.     return (
1711.         <button type="submit" className="btn btn-block" disabled={isLoading}>
1712.             submit
1713.         </button>
1714.     );
1715.     ````
1716.
1717.     #### Axios
1718.
1719.     - [axios docs](https://axios-http.com/docs/intro)
1720.     - stop app
1721.     - cd client
1722.
1723.     ```sh
1724.     npm install axios
1725.     ```
1726.
1727.     - cd ..
1728.     - restart app
1729.
1730.     #### Register User - Complete
1731.
1732.     ```js
1733.     appContext.js;
1734.
1735.     import axios from "axios";
1736.
1737.     const registerUser = async (currentUser) => {
1738.         dispatch({ type: REGISTER_USER_BEGIN });
1739.         try {
1740.             const response = await axios.post("/api/v1/auth/register", currentUser);
1741.             console.log(response);
1742.             const { user, token, location } = response.data;
1743.             dispatch({
1744.                 type: REGISTER_USER_SUCCESS,
1745.                 payload: {
1746.                     user,
1747.                     token,
1748.                     location,
1749.                 },
1750.             });
1751.
1752.             // will add later
1753.             // addUserToLocalStorage({
1754.             //     user,
1755.             //     token,
1756.             //     location,
1757.             // })
1758.         } catch (error) {
1759.             console.log(error.response);
1760.             dispatch({
1761.                 type: REGISTER_USER_ERROR,
1762.                 payload: { msg: error.response.data.msg },
1763.             });

```

```

1764.     }
1765.     clearAlert();
1766. };
1767. ...
1768.
1769. ```js
1770. reducer.js;
1771. if (action.type === REGISTER_USER_BEGIN) {
1772.     return { ...state, isLoading: true };
1773. }
1774. if (action.type === REGISTER_USER_SUCCESS) {
1775.     return {
1776.         ...state,
1777.         user: action.payload.user,
1778.         token: action.payload.token,
1779.         userLocation: action.payload.location,
1780.         jobLocation: action.payload.location,
1781.         isLoading: false,
1782.         showAlert: true,
1783.         alertType: "success",
1784.         alertText: "User Created! Redirecting...",
1785.     };
1786. }
1787. if (action.type === REGISTER_USER_ERROR) {
1788.     return {
1789.         ...state,
1790.         isLoading: false,
1791.         showAlert: true,
1792.         alertType: "danger",
1793.         alertText: action.payload.msg,
1794.     };
1795. }
1796. ...
1797.
1798. ##### Navigate To Dashboard
1799.
1800. ```js
1801. Register.js;
1802. import { useEffect } from "react";
1803. import { useNavigate } from "react-router-dom";
1804.
1805. const Register = () => {
1806.     const { user } = useAppContext();
1807.     const navigate = useNavigate();
1808.
1809.     useEffect(() => {
1810.         if (user) {
1811.             setTimeout(() => {
1812.                 navigate("/");
1813.             }, 3000);
1814.         }
1815.     }, [user, navigate]);
1816. };
1817. ...
1818.
1819. ##### Local Storage
1820.
1821. ```js
1822. appContext.js;
1823. const addUserToLocalStorage = ({ user, token, location }) => {
1824.     localStorage.setItem("user", JSON.stringify(user));

```

```

1825.     localStorage.setItem("token", token);
1826.     localStorage.setItem("location", location);
1827. };
1828.
1829. const removeUserFromLocalStorage = () => {
1830.     localStorage.removeItem("token");
1831.     localStorage.removeItem("user");
1832.     localStorage.removeItem("location");
1833. };
1834.
1835. const registerUser = async (currentUser) => {
1836.     // in try block
1837.     addUserToLocalStorage({
1838.         user,
1839.         token,
1840.         location,
1841.     });
1842. };
1843.
1844. // set as default
1845. const token = localStorage.getItem("token");
1846. const user = localStorage.getItem("user");
1847. const userLocation = localStorage.getItem("location");
1848.
1849. const initialState = {
1850.     user: user ? JSON.parse(user) : null,
1851.     token: token,
1852.     userLocation: userLocation || "",
1853.     jobLocation: userLocation || "",
1854. };
1855.
1856.
1857. ##### Morgan Package
1858.
1859. - http logger middleware for node.js
1860. - [morgan docs](https://www.npmjs.com/package/morgan)
1861.
1862. ```sh
1863. npm install morgan
1864. ```
1865.
1866. ```js
1867. import morgan from "morgan";
1868.
1869. if (process.env.NODE_ENV !== "production") {
1870.     app.use(morgan("dev"));
1871. }
1872. ```
1873.
1874. ##### UnauthenticatedError
1875.
1876. - unauthenticated.js in errors
1877. - import/export
1878.
1879. ```js
1880. import { StatusCodes } from "http-status-codes";
1881. import CustomAPIError from "../custom-api.js";
1882.
1883. class UnauthenticatedError extends CustomAPIError {
1884.     constructor(message) {
1885.         super(message);

```

```

1886.         this.statusCode = StatusCodes.UNAUTHORIZED;
1887.     }
1888. }
1889. ...
1890.
1891. ##### Compare Password
1892.
1893. ```js
1894. User.js in models;
1895.
1896. UserSchema.methods.comparePassword = async function (candidatePassword) {
1897.     const isMatch = await bcrypt.compare(candidatePassword, this.password);
1898.     return isMatch;
1899. };
1900. ...
1901.
1902. ```js
1903. authController.js;
1904. const login = async (req, res) => {
1905.     const { email, password } = req.body;
1906.     if (!email || !password) {
1907.         throw new BadRequestError("Please provide all values");
1908.     }
1909.     const user = await User.findOne({ email }).select("+password");
1910.
1911.     if (!user) {
1912.         throw new UnauthenticatedError("Invalid Credentials");
1913.     }
1914.     const isPasswordCorrect = await user.comparePassword(password);
1915.     if (!isPasswordCorrect) {
1916.         throw new UnauthenticatedError("Invalid Credentials");
1917.     }
1918.     const token = user.createJWT();
1919.     user.password = undefined;
1920.     res.status(StatusCodes.OK).json({ user, token, location: user.location });
1921. };
1922. ...
1923.
1924. - test in Postman
1925.
1926. ##### Login User - Setup
1927.
1928. - actions.js LOGIN_USER_BEGIN,SUCCESS,ERROR
1929. - import reducer,appContext
1930.
1931. ```js
1932. appContext.js;
1933. const loginUser = async (currentUser) => {
1934.     console.log(currentUser);
1935. };
1936. ...
1937.
1938. - import in Register.js
1939.
1940. ```js
1941. Register.js;
1942.
1943. if (isMember) {
1944.     loginUser(currentUser);
1945. } else {
1946.     registerUser(currentUser);

```



```

1947.     }
1948.     ...
1949.
1950.     ##### Login User - Complete
1951.
1952.     ```js
1953.     appContext.js;
1954.     const loginUser = async (currentUser) => {
1955.         dispatch({ type: LOGIN_USER_BEGIN });
1956.         try {
1957.             const { data } = await axios.post("/api/v1/auth/login", currentUser);
1958.             const { user, token, location } = data;
1959.
1960.             dispatch({
1961.                 type: LOGIN_USER_SUCCESS,
1962.                 payload: { user, token, location },
1963.             });
1964.
1965.             addUserToLocalStorage({ user, token, location });
1966.         } catch (error) {
1967.             dispatch({
1968.                 type: LOGIN_USER_ERROR,
1969.                 payload: { msg: error.response.data.msg },
1970.             });
1971.         }
1972.         clearAlert();
1973.     };
1974.     ...
1975.
1976.     ```js
1977.     reducer.js;
1978.
1979.     if (action.type === LOGIN_USER_BEGIN) {
1980.         return {
1981.             ...state,
1982.             isLoading: true,
1983.         };
1984.     }
1985.     if (action.type === LOGIN_USER_SUCCESS) {
1986.         return {
1987.             ...state,
1988.             isLoading: false,
1989.             user: action.payload.user,
1990.             token: action.payload.token,
1991.             userLocation: action.payload.location,
1992.             jobLocation: action.payload.location,
1993.             showAlert: true,
1994.             alertType: "success",
1995.             alertText: "Login Successful! Redirecting...",
1996.         };
1997.     }
1998.     if (action.type === LOGIN_USER_ERROR) {
1999.         return {
2000.             ...state,
2001.             isLoading: false,
2002.             showAlert: true,
2003.             alertType: "danger",
2004.             alertText: action.payload.msg,
2005.         };
2006.     }
2007.     ...

```

```

2008.
2009.     ##### Refactor
2010.
2011.     ```js
2012.     actions.js;
2013.     export const SETUP_USER_BEGIN = "SETUP_USER_BEGIN";
2014.     export const SETUP_USER_SUCCESS = "SETUP_USER_SUCCESS";
2015.     export const SETUP_USER_ERROR = "SETUP_USER_ERROR";
2016.     ```
2017.
2018.     ```js
2019.     appContext.js;
2020.
2021.     const setupUser = async ({ currentUser, endPoint, alertText }) => {
2022.         dispatch({ type: SETUP_USER_BEGIN });
2023.         try {
2024.             const { data } = await axios.post(`/api/v1/auth/${endPoint}`, currentUser);
2025.
2026.             const { user, token, location } = data;
2027.             dispatch({
2028.                 type: SETUP_USER_SUCCESS,
2029.                 payload: { user, token, location, alertText },
2030.             });
2031.             addUserToLocalStorage({ user, token, location });
2032.         } catch (error) {
2033.             dispatch({
2034.                 type: SETUP_USER_ERROR,
2035.                 payload: { msg: error.response.data.msg },
2036.             });
2037.         }
2038.         clearAlert();
2039.     };
2040.     ```
2041.
2042.     ```js
2043.     reducer.js;
2044.     if (action.type === SETUP_USER_BEGIN) {
2045.         return { ...state, isLoading: true };
2046.     }
2047.     if (action.type === SETUP_USER_SUCCESS) {
2048.         return {
2049.             ...state,
2050.             isLoading: false,
2051.             token: action.payload.token,
2052.             user: action.payload.user,
2053.             userLocation: action.payload.location,
2054.             jobLocation: action.payload.location,
2055.             showAlert: true,
2056.             alertType: "success",
2057.             alertText: action.payload.alertText,
2058.         };
2059.     }
2060.     if (action.type === SETUP_USER_ERROR) {
2061.         return {
2062.             ...state,
2063.             isLoading: false,
2064.             showAlert: true,
2065.             alertType: "danger",
2066.             alertText: action.payload.msg,
2067.         };
2068.     }

```

```

2069.   ```
2070.
2071.   - import/export
2072.
2073.   ```js
2074.   Register.js;
2075.
2076.   const onSubmit = (e) => {
2077.       e.preventDefault();
2078.       const { name, email, password, isMember } = values;
2079.       if (!email || !password || (!isMember && !name)) {
2080.           displayAlert();
2081.           return;
2082.       }
2083.       const currentUser = { name, email, password };
2084.       if (isMember) {
2085.           setupUser({
2086.               currentUser,
2087.               endPoint: "login",
2088.               alertText: "Login Successful! Redirecting...",
2089.           });
2090.       } else {
2091.           setupUser({
2092.               currentUser,
2093.               endPoint: "register",
2094.               alertText: "User Created! Redirecting...",
2095.           });
2096.       }
2097.   };
2098.   ```
2099.
2100.   ##### Nested Pages in React Router 6
2101.
2102.   ##### Dashboard pages
2103.
2104.   - delete Dashboard.js
2105.   - fix imports/exports
2106.   - replace in home route
2107.
2108.   ```js
2109.   <Route path="/" element={<div>dashboard</div>} />
2110.   ```
2111.
2112.   - create <b>dashboard</b> directory in pages
2113.   - create AddJob, AllJobs, Profile, Stats, SharedLayout, index.js
2114.   - setup basic returns
2115.
2116.   ```js
2117.   return <h1>Add Job Page</h1>;
2118.   ```
2119.
2120.   - export all with index.js (just like components)
2121.   - import all pages in App.js
2122.
2123.   ##### Nested Structure
2124.
2125.   ```js
2126.   App.js
2127.
2128.   <Route path="/" >
2129.       <Route path="stats" element={<Stats />} />

```

```

2130.     <Route path='all-jobs' element={<AllJobs />}></Route>
2131.     <Route path='add-job' element={<AddJob />}></Route>
2132.     <Route path='profile' element={<Profile />}></Route>
2133. </Route>
2134. ```
2135.
2136. ##### Shared Layout
2137.
2138. ```js
2139. App.js
2140.
2141. <Route path='/' element={<SharedLayout/>} >
2142. ```
2143.
2144. ```js
2145. SharedLayout.js;
2146.
2147. import { Outlet, Link } from "react-router-dom";
2148. import Wrapper from "../../assets/wrappers/SharedLayout";
2149.
2150. const SharedLayout = () => {
2151.   return (
2152.     <Wrapper>
2153.       <nav>
2154.         <Link to="all-jobs">all jobs</Link>
2155.         <Link to="add-job">all jobs</Link>
2156.       </nav>
2157.       <Outlet />
2158.     </Wrapper>
2159.   );
2160. };
2161.
2162. export default SharedLayout;
2163. ```
2164.
2165. ```js
2166. App.js
2167.
2168. <Route index element={<Stats/>} >
2169. ```
2170.
2171. ##### Protected Route
2172.
2173. - create ProtectedRoute.js in pages
2174. - import/export
2175. - wrap SharedLayout in App.js
2176.
2177. ```js
2178. <Route
2179.   path="/"
2180.   element={
2181.     <ProtectedRoute>
2182.       <SharedLayout />
2183.     </ProtectedRoute>
2184.   }
2185. />
2186. ```
2187.
2188. ```js
2189. ProtectedRoute.js;
2190.

```

```

2191. import { Navigate } from "react-router-dom";
2192. import { useAppContext } from "../context/appContext";
2193.
2194. const ProtectedRoute = ({ children }) => {
2195.   const { user } = useAppContext();
2196.   if (!user) {
2197.     return <Navigate to="/landing" />;
2198.   }
2199.   return children;
2200. };
2201.
2202.
2203. ##### Navbar, SmallSidebar, BigSidebar
2204.
2205. - create Navbar, SmallSidebar, BigSidebar in components
2206. - import Wrappers from assets/wrappers
2207. - simple return
2208. - import/export
2209.
2210. ```js
2211. SharedLayout.js;
2212.
2213. import { Outlet } from "react-router-dom";
2214. import { Navbar, SmallSidebar, BigSidebar } from "../../components";
2215. import Wrapper from "../../assets/wrappers/SharedLayout";
2216.
2217. const SharedLayout = () => {
2218.   const { user } = useAppContext();
2219.   return (
2220.     <>
2221.       <Wrapper>
2222.         <main className="dashboard">
2223.           <SmallSidebar />
2224.           <BigSidebar />
2225.           <div>
2226.             <Navbar />
2227.             <div className="dashboard-page">
2228.               <Outlet />
2229.             </div>
2230.           </div>
2231.         </main>
2232.       </Wrapper>
2233.     </>
2234.   );
2235. };
2236.
2237. export default SharedLayout;
2238.
2239.
2240. ##### React Icons
2241.
2242. [React Icons](https://react-icons.github.io/react-icons/)
2243.
2244. ```sh
2245. npm install react-icons
2246.
2247.
2248. ```js
2249. Navbar.js
2250.
2251. import Wrapper from '../assets/wrappers/Navbar'

```

```

2252.     import {FaHome} from 'react-icons/fa'
2253.     const Navbar = () => {
2254.         return (
2255.             <Wrapper>
2256.                 <h4>navbar</h4>
2257.                 <FaHome>
2258.             </Wrapper>
2259.         )
2260.     }
2261.
2262.     export default Navbar
2263.
2264.     ```
2265.
2266.     ##### Navbar Setup
2267.
2268.     ```js
2269.     Navbar.js;
2270.
2271.     import { useState } from "react";
2272.     import { FaAlignLeft, FaUserCircle, FaCaretDown } from "react-icons/fa";
2273.     import { useContext } from "../context/appContext";
2274.     import Logo from "../Logo";
2275.     import Wrapper from "../assets/wrappers/Navbar";
2276.     const Navbar = () => {
2277.         return (
2278.             <Wrapper>
2279.                 <div className="nav-center">
2280.                     <button
2281.                         className="toggle-btn"
2282.                         onClick={() => console.log("toggle sidebar")}
2283.                     >
2284.                         <FaAlignLeft />
2285.                     </button>
2286.
2287.                     <div>
2288.                         <Logo />
2289.                         <h3 className="logo-text">dashboard</h3>
2290.                     </div>
2291.
2292.                     <div className="btn-container">
2293.                         <button className="btn" onClick={() => console.log("show logout")}>
2294.                             <FaUserCircle />
2295.                             john
2296.                             <FaCaretDown />
2297.                         </button>
2298.                         <div className="dropdown show-dropdown">
2299.                             <button
2300.                                 onClick={() => console.log("logout user")}
2301.                                 className="dropdown-btn"
2302.                             >
2303.                                 logout
2304.                             </button>
2305.                         </div>
2306.                     </div>
2307.                 </div>
2308.             </Wrapper>
2309.         );
2310.     };
2311.
2312.     export default Navbar;

```

```

2313.     ```
2314.
2315.     ##### Toggle Sidebar
2316.
2317.     ```js
2318.     actions.js;
2319.
2320.     export const TOGGLE_SIDEBAR = "TOGGLE_SIDEBAR";
2321.     ```
2322.
2323.     - import/export
2324.
2325.     ```js
2326.     appContext.js;
2327.
2328.     const initialState = {
2329.       showSidebar: false,
2330.     };
2331.
2332.     const toggleSidebar = () => {
2333.       dispatch({ type: TOGGLE_SIDEBAR });
2334.     };
2335.     ```
2336.
2337.     ```js
2338.     reducer.js;
2339.
2340.     if (action.type === TOGGLE_SIDEBAR) {
2341.       return { ...state, showSidebar: !state.showSidebar };
2342.     }
2343.     ```
2344.
2345.     ```js
2346.     Navbar.js;
2347.
2348.     const { toggleSidebar } = useAppContext();
2349.
2350.     return (
2351.       <button className="toggle-btn" onClick={toggleSidebar}>
2352.         <FaAlignLeft />
2353.       </button>
2354.     );
2355.     ```
2356.
2357.     ##### Toggle Dropdown
2358.
2359.     ```js
2360.     Navbar.js
2361.
2362.     const [showLogout, setShowLogout] = useState(false)
2363.
2364.     <div className='btn-container'>
2365.       <button className='btn' onClick={() => setShowLogout(!showLogout)}>
2366.         <FaUserCircle />
2367.         {user.name}
2368.         <FaCaretDown />
2369.       </button>
2370.       <div className={showLogout ? 'dropdown show-dropdown' : 'dropdown'}>
2371.         <button onClick={() => logoutUser()} className='dropdown-btn'>
2372.           logout
2373.         </button>

```

```

2374.     </div>
2375. </div>
2376.
2377.   ```
2378.
2379.   ##### Logout User
2380.
2381.   ```js
2382.   actions.js;
2383.
2384.   export const LOGOUT_USER = "LOGOUT_USER";
2385.   ```
2386.
2387.   - import/export
2388.
2389.   ```js
2390.   appContext.js
2391.
2392.   const logoutUser = () => {
2393.     dispatch({ type: LOGOUT_USER })
2394.     removeUserFromLocalStorage()
2395.   }
2396.
2397.   value={{logoutUser}}
2398.   ```
2399.
2400.   ```js
2401.   reducer.js;
2402.
2403.   import { initialState } from "../appContext";
2404.
2405.   if (action.type === LOGOUT_USER) {
2406.     return {
2407.       ...initialState,
2408.       user: null,
2409.       token: null,
2410.       userLocation: "",
2411.       jobLocation: "",
2412.     };
2413.   }
2414.   ```
2415.
2416.   ```js
2417.   Navbar.js;
2418.
2419.   const { user, logoutUser, toggleSidebar } = useAppContext();
2420.
2421.   return (
2422.     <div className="btn-container">
2423.       <button className="btn" onClick={() => setShowLogout(!showLogout)}>
2424.         <FaUserCircle />
2425.         {user.name}
2426.         {user && user.name}
2427.         {user?.name} // optional chaining
2428.         <FaCaretDown />
2429.       </button>
2430.       <div className={showLogout ? "dropdown show-dropdown" : "dropdown"}>
2431.         <button onClick={logoutUser} className="dropdown-btn">
2432.           logout
2433.         </button>
2434.       </div>

```



```

2435.     </div>
2436.   );
2437.   ```
2438.
2439.   ##### Setup Links
2440.
2441.   - create <b>utils</b>in the <b>src</b>
2442.   - setup links.js
2443.
2444.   ```js
2445.   import { IoBarChartSharp } from "react-icons/io5";
2446.   import { MdQueryStats } from "react-icons/md";
2447.   import { FaWpforms } from "react-icons/fa";
2448.   import { ImProfile } from "react-icons/im";
2449.
2450.   const links = [
2451.     {
2452.       id: 1,
2453.       text: "stats",
2454.       path: "/",
2455.       icon: <IoBarChartSharp />,
2456.     },
2457.     {
2458.       id: 2,
2459.       text: "all jobs",
2460.       path: "all-jobs",
2461.       icon: <MdQueryStats />,
2462.     },
2463.     {
2464.       id: 3,
2465.       text: "add job",
2466.       path: "add-job",
2467.       icon: <FaWpforms />,
2468.     },
2469.     {
2470.       id: 4,
2471.       text: "profile",
2472.       path: "profile",
2473.       icon: <ImProfile />,
2474.     },
2475.   ];
2476.
2477.   export default links;
2478.   ```
2479.
2480.   ##### Small Sidebar - Setup
2481.
2482.   ```js
2483.   SmallSidebar.js;
2484.
2485.   import Wrapper from "../assets/wrappers/SmallSidebar";
2486.   import { FaTimes } from "react-icons/fa";
2487.   import { useAppContext } from "../context/appContext";
2488.   import links from "../utils/links";
2489.   import { NavLink } from "react-router-dom";
2490.   import Logo from "../Logo";
2491.
2492.   export const SmallSidebar = () => {
2493.     return (
2494.       <Wrapper>
2495.         <div className="sidebar-container show-sidebar">

```

```

2496.         <div className="content">
2497.             <button className="close-btn" onClick={() => console.log("toggle")}>
2498.                 <FaTimes />
2499.             </button>
2500.             <header>
2501.                 <Logo />
2502.             </header>
2503.             <div className="nav-links">nav links</div>
2504.         </div>
2505.     </div>
2506. </Wrapper>
2507. );
2508. };
2509.
2510. export default SmallSidebar;
2511. ```
2512.
2513. ##### Small Sidebar - Toggle
2514.
2515. ```js
2516. SmallSidebar.js;
2517.
2518. const { showSidebar, toggleSidebar } = useAppContext();
2519. ```
2520.
2521. ```js
2522. SmallSidebar.js;
2523.
2524. return (
2525.     <div
2526.         className={
2527.             showSidebar ? "sidebar-container show-sidebar" : "sidebar-container"
2528.         }
2529.     ></div>
2530. );
2531. ```
2532.
2533. ```js
2534. SmallSidebar.js;
2535.
2536. return (
2537.     <button className="close-btn" onClick={toggleSidebar}>
2538.         <FaTimes />
2539.     </button>
2540. );
2541. ```
2542.
2543. ##### Small Sidebar - Nav Links
2544.
2545. ```js
2546. SmallSidebar.js;
2547.
2548. import { NavLink } from "react-router-dom";
2549.
2550. return (
2551.     <div className="nav-links">
2552.         {links.map((link) => {
2553.             const { text, path, id, icon } = link;
2554.
2555.             return (
2556.                 <NavLink

```

```

2557.         to={path}
2558.         className={({ isActive }) =>
2559.             isActive ? "nav-link active" : "nav-link"
2560.         }
2561.         key={id}
2562.         onClick={toggleSidebar}
2563.     >
2564.         <span className="icon">{icon}</span>
2565.         {text}
2566.     </NavLink>
2567. );
2568. }}}
2569. </div>
2570. );
2571. ...
2572.
2573. ##### Nav Links Component
2574.
2575. - in <b>components</b> create NavLinks.js
2576. - styles still set from Wrapper
2577. - also can setup in links.js, preference
2578.
2579. ```js
2580. import { NavLink } from "react-router-dom";
2581. import links from "../utils/links";
2582.
2583. const NavLinks = ({ toggleSidebar }) => {
2584.     return (
2585.         <div className="nav-links">
2586.             {links.map((link) => {
2587.                 const { text, path, id, icon } = link;
2588.
2589.                 return (
2590.                     <NavLink
2591.                         to={path}
2592.                         key={id}
2593.                         onClick={toggleSidebar}
2594.                         className={({ isActive }) =>
2595.                             isActive ? "nav-link active" : "nav-link"
2596.                         }
2597.                     >
2598.                         <span className="icon">{icon}</span>
2599.                         {text}
2600.                     </NavLink>
2601.                 );
2602.             })}
2603.         </div>
2604.     );
2605. };
2606.
2607. export default NavLinks;
2608. ```
2609.
2610. ```js
2611. SmallSidebar.js
2612.
2613. import NavLinks from './NavLinks'
2614.
2615. return <NavLinks toggleSidebar={toggleSidebar}>
2616. ```
2617.

```

```

2618.     #### Big Sidebar
2619.
2620.     ```js
2621.     import { useAppContext } from "../context/appContext";
2622.     import NavLinks from "../NavLinks";
2623.     import Logo from "../components/Logo";
2624.     import Wrapper from "../assets/wrappers/BigSidebar";
2625.
2626.     const BigSidebar = () => {
2627.         const { showSidebar } = useAppContext();
2628.         return (
2629.             <Wrapper>
2630.                 <div
2631.                     className={
2632.                         showSidebar ? "sidebar-container " : "sidebar-container show-sidebar"
2633.                     }
2634.                 >
2635.                     <div className="content">
2636.                         <header>
2637.                             <Logo />
2638.                         </header>
2639.                         <NavLinks />
2640.                     </div>
2641.                 </div>
2642.             </Wrapper>
2643.         );
2644.     };
2645.
2646.     export default BigSidebar;
2647.     ```
2648.
2649.     #### REACT ROUTER UPDATE !!!
2650.
2651.     - [Stack Overflow](https://stackoverflow.com/questions/70644361/react-router-dom-v6-shows-active-for-index-as-well-as-other-subroutes)
2652.
2653.     ```js
2654.     <NavLink
2655.         to={path}
2656.         key={id}
2657.         onClick={toggleSidebar}
2658.         className={({ isActive }) =>
2659.             isActive ? 'nav-link active' : 'nav-link'}
2660.     >
2661.     end
2662.     >
2663.     ```
2664.
2665.
2666.     #### Authenticate User Setup
2667.
2668.     - create auth.js in <b>middleware</b>
2669.
2670.     ```js
2671.     const auth = async (req, res, next) => {
2672.         console.log("authenticate user");
2673.         next();
2674.     };
2675.
2676.     export default auth;
2677.     ```

```

```

2678.
2679.   ```js
2680.   authRoutes.js;
2681.
2682.   import authenticateUser from "../middleware/auth.js";
2683.
2684.   router.route("/updateUser").patch(authenticateUser, updateUser);
2685.   ```
2686.
2687.   - two options
2688.
2689.   ```js
2690.   server.js;
2691.
2692.   import authenticateUser from "../middleware/auth.js";
2693.   app.use("/api/v1/jobs", authenticateUser, jobsRouter);
2694.   ```
2695.
2696.   ```js
2697.   jobsRoutes.js;
2698.
2699.   import authenticateUser from "../middleware/auth.js";
2700.
2701.   // all routes !!!!
2702.
2703.   router.route("/stats").get(authenticateUser, showStats);
2704.   ```
2705.
2706.   ##### Auth - Bearer Schema
2707.
2708.   ```js
2709.   Postman
2710.
2711.   Headers
2712.
2713.   Authorization: Bearer <token>
2714.   ```
2715.
2716.
2717.   ```js
2718.   auth.js;
2719.
2720.   const auth = async (req, res, next) => {
2721.     const headers = req.headers;
2722.     const authHeader = req.headers.authorization;
2723.     console.log(headers);
2724.     console.log(authHeader);
2725.     next();
2726.   };
2727.   ```
2728.
2729.   ##### Postman - Set Token Programmatically
2730.
2731.   - register and login routes
2732.   - Tests
2733.
2734.   ```js
2735.   const jsonData = pm.response.json();
2736.   pm.globals.set("token", jsonData.token);
2737.
2738.   Type: Bearer;

```

```

2739.
2740.   Token: {
2741.     {
2742.       token;
2743.     }
2744.   }
2745.   ...
2746.
2747.   ##### Unauthenticated Error
2748.
2749.   ```js
2750.   auth.js;
2751.
2752.   import { UnAuthenticatedError } from "../errors/index.js";
2753.
2754.   const auth = async (req, res, next) => {
2755.     const authHeader = req.headers.authorization;
2756.
2757.     if (!authHeader) {
2758.       // why, well is it 400 or 404?
2759.       // actually 401
2760.       throw new UnAuthenticatedError("Authentication Invalid");
2761.     }
2762.
2763.     next();
2764.   };
2765.   ...
2766.
2767.   ##### Auth Middleware
2768.
2769.   ```js
2770.   import jwt from "jsonwebtoken";
2771.   import { UnAuthenticatedError } from "../errors/index.js";
2772.
2773.   const auth = async (req, res, next) => {
2774.     // check header
2775.     const authHeader = req.headers.authorization;
2776.     if (!authHeader || !authHeader.startsWith("Bearer ")) {
2777.       throw new UnauthenticatedError("Authentication invalid");
2778.     }
2779.     const token = authHeader.split(" ")[1];
2780.
2781.     try {
2782.       const payload = jwt.verify(token, process.env.JWT_SECRET);
2783.       // console.log(payload)
2784.       // attach the user request object
2785.       // req.user = payload
2786.       req.user = { userId: payload.userId };
2787.       next();
2788.     } catch (error) {
2789.       throw new UnauthenticatedError("Authentication invalid");
2790.     }
2791.   };
2792.
2793.   export default auth;
2794.   ...
2795.
2796.   ##### Update User
2797.
2798.   ```js
2799.   const updateUser = async (req, res) => {

```

```

2800.     const { email, name, lastName, location } = req.body;
2801.     if (!email || !name || !lastName || !location) {
2802.         throw new BadRequestError("Please provide all values");
2803.     }
2804.
2805.     const user = await User.findOne({ _id: req.user.userId });
2806.
2807.     user.email = email;
2808.     user.name = name;
2809.     user.lastName = lastName;
2810.     user.location = location;
2811.
2812.     await user.save();
2813.
2814.     // various setups
2815.     // in this case only id
2816.     // if other properties included, must re-generate
2817.
2818.     const token = user.createJWT();
2819.     res.status(StatusCodes.OK).json({
2820.         user,
2821.         token,
2822.         location: user.location,
2823.     });
2824. };
2825. ...
2826.
2827. ##### Modified Paths
2828.
2829. - user.save() vs User.findOneAndUpdate
2830.
2831. ```js
2832. User.js;
2833.
2834. UserSchema.pre("save", async function () {
2835.     console.log(this.modifiedPaths());
2836.     console.log(this.isModified("name"));
2837.
2838.     // if (!this.isModified('password')) return
2839.     // const salt = await bcrypt.genSalt(10)
2840.     // this.password = await bcrypt.hash(this.password, salt)
2841. });
2842. ...
2843.
2844. ##### Profile Page
2845.
2846. ```js
2847. appContext.js
2848.
2849. const updateUser = async (currentUser) => {
2850.     console.log(currentUser)
2851. }
2852.
2853. value={{updateUser}}
2854. ...
2855.
2856. ```js
2857. Profile.js;
2858.
2859. import { useState } from "react";
2860. import { FormRow, Alert } from "../components";

```

```

2861. import { useAppContext } from "../../context/appContext";
2862. import Wrapper from "../../assets/wrappers/DashboardFormPage";
2863.
2864. const Profile = () => {
2865.   const { user, showAlert, displayAlert, updateUser, isLoading } =
2866.     useAppContext();
2867.   const [name, setName] = useState(user?.name);
2868.   const [email, setEmail] = useState(user?.email);
2869.   const [lastName, setLastName] = useState(user?.lastName);
2870.   const [location, setLocation] = useState(user?.location);
2871.
2872.   const handleSubmit = (e) => {
2873.     e.preventDefault();
2874.     if (!name || !email || !lastName || !location) {
2875.       // test and remove temporary
2876.       displayAlert();
2877.       return;
2878.     }
2879.
2880.     updateUser({ name, email, lastName, location });
2881.   };
2882.   return (
2883.     <Wrapper>
2884.       <form className="form" onSubmit={handleSubmit}>
2885.         <h3>profile </h3>
2886.         {showAlert && <Alert />}
2887.
2888.         {/* name */}
2889.         <div className="form-center">
2890.           <FormRow
2891.             type="text"
2892.             name="name"
2893.             value={name}
2894.             handleChange={(e) => setName(e.target.value)}
2895.           />
2896.           <FormRow
2897.             labelText="last name"
2898.             type="text"
2899.             name="lastName"
2900.             value={lastName}
2901.             handleChange={(e) => setLastName(e.target.value)}
2902.           />
2903.           <FormRow
2904.             type="email"
2905.             name="email"
2906.             value={email}
2907.             handleChange={(e) => setEmail(e.target.value)}
2908.           />
2909.
2910.           <FormRow
2911.             type="text"
2912.             name="location"
2913.             value={location}
2914.             handleChange={(e) => setLocation(e.target.value)}
2915.           />
2916.           <button className="btn btn-block" type="submit" disabled={isLoading}>
2917.             {isLoading ? "Please Wait..." : "save changes"}
2918.           </button>
2919.         </div>
2920.       </form>
2921.     </Wrapper>

```



```

2922.     );
2923. };
2924.
2925. export default Profile;
2926. ```
2927.
2928. ##### Bearer Token - Manual Approach
2929.
2930. ```js
2931. appContext.js;
2932.
2933. const updaterUser = async (currentUser) => {
2934.   try {
2935.     const { data } = await axios.patch("/api/v1/auth/updateUser", currentUser, {
2936.       headers: {
2937.         Authorization: `Bearer ${state.token}`,
2938.       },
2939.     });
2940.     console.log(data);
2941.   } catch (error) {
2942.     console.log(error.response);
2943.   }
2944. };
2945. ```
2946.
2947. ##### Axios - Global Setup
2948.
2949. <!-- IMPORTANT -->
2950.
2951. In current axios version,
2952. common property returns undefined,
2953. so we don't use it anymore!!!
2954.
2955. ```js
2956. appContext.js;
2957.
2958. axios.defaults.headers["Authorization"] = `Bearer ${state.token}`;
2959. ```
2960.
2961. ##### Axios - Setup Instance
2962.
2963. ```js
2964. AppContext.js;
2965.
2966. const authFetch = axios.create({
2967.   baseURL: "/api/v1",
2968.   headers: {
2969.     Authorization: `Bearer ${state.token}`,
2970.   },
2971. });
2972.
2973. const updaterUser = async (currentUser) => {
2974.   try {
2975.     const { data } = await authFetch.patch("/auth/updateUser", currentUser);
2976.   } catch (error) {
2977.     console.log(error.response);
2978.   }
2979. };
2980. ```
2981.
2982. ##### Axios - Interceptors

```

```

2983.
2984.   - will use instance, but can use axios instead
2985.
2986.   <!-- IMPORTANT -->
2987.
2988.   In current axios version,
2989.   common property returns undefined,
2990.   so we don't use it anymore!!!
2991.
2992.   ```js
2993.   appContext.js;
2994.
2995.   // response interceptor
2996.   authFetch.interceptors.request.use(
2997.     (config) => {
2998.       config.headers["Authorization"] = `Bearer ${state.token}`;
2999.       return config;
3000.     },
3001.     (error) => {
3002.       return Promise.reject(error);
3003.     }
3004.   );
3005.   // response interceptor
3006.   authFetch.interceptors.response.use(
3007.     (response) => {
3008.       return response;
3009.     },
3010.     (error) => {
3011.       console.log(error.response);
3012.       if (error.response.status === 401) {
3013.         console.log("AUTH ERROR");
3014.       }
3015.       return Promise.reject(error);
3016.     }
3017.   );
3018.   ```
3019.
3020.   #### Update User
3021.
3022.   ```js
3023.   actions.js;
3024.   export const UPDATE_USER_BEGIN = "UPDATE_USER_BEGIN";
3025.   export const UPDATE_USER_SUCCESS = "UPDATE_USER_SUCCESS";
3026.   export const UPDATE_USER_ERROR = "UPDATE_USER_ERROR";
3027.   ```
3028.
3029.   ```js
3030.   appContext.js;
3031.
3032.   const updateUser = async (currentUser) => {
3033.     dispatch({ type: UPDATE_USER_BEGIN });
3034.     try {
3035.       const { data } = await authFetch.patch("/auth/updateUser", currentUser);
3036.
3037.       // no token
3038.       const { user, location, token } = data;
3039.
3040.       dispatch({
3041.         type: UPDATE_USER_SUCCESS,
3042.         payload: { user, location, token },
3043.       });

```

```

3044.
3045.     addUserToLocalStorage({ user, location, token });
3046.   } catch (error) {
3047.     dispatch({
3048.       type: UPDATE_USER_ERROR,
3049.       payload: { msg: error.response.data.msg },
3050.     });
3051.   }
3052.   clearAlert();
3053. };
3054. ...
3055.
3056. ```js
3057. reducer.js
3058. if (action.type === UPDATE_USER_BEGIN) {
3059.   return { ...state, isLoading: true }
3060. }
3061.
3062. if (action.type === UPDATE_USER_SUCCESS) {
3063.   return {
3064.     ...state,
3065.     isLoading: false,
3066.     token: action.payload.token
3067.     user: action.payload.user,
3068.     userLocation: action.payload.location,
3069.     jobLocation: action.payload.location,
3070.     showAlert: true,
3071.     alertType: 'success',
3072.     alertText: 'User Profile Updated!',
3073.   }
3074. }
3075. if (action.type === UPDATE_USER_ERROR) {
3076.   return {
3077.     ...state,
3078.     isLoading: false,
3079.     showAlert: true,
3080.     alertType: 'danger',
3081.     alertText: action.payload.msg,
3082.   }
3083. }
3084. ...
3085.
3086. ##### 401 Error - Logout User
3087.
3088. ```js
3089. appContext.js;
3090. // response interceptor
3091. authFetch.interceptors.response.use(
3092.   (response) => {
3093.     return response;
3094.   },
3095.   (error) => {
3096.     if (error.response.status === 401) {
3097.       logoutUser();
3098.     }
3099.     return Promise.reject(error);
3100.   }
3101. );
3102.
3103. const updateUser = async (currentUser) => {
3104.   dispatch({ type: UPDATE_USER_BEGIN });

```

```

3105.     try {
3106.         const { data } = await authFetch.patch("/auth/updateUser", currentUser);
3107.
3108.         // no token
3109.         const { user, location } = data;
3110.
3111.         dispatch({
3112.             type: UPDATE_USER_SUCCESS,
3113.             payload: { user, location, token },
3114.         });
3115.
3116.         addUserToLocalStorage({ user, location, token: initialState.token });
3117.     } catch (error) {
3118.         if (error.response.status !== 401) {
3119.             dispatch({
3120.                 type: UPDATE_USER_ERROR,
3121.                 payload: { msg: error.response.data.msg },
3122.             });
3123.         }
3124.     }
3125.     clearAlert();
3126. };
3127. ```
3128.
3129. ##### Job Model
3130.
3131. - Job Model
3132.
3133. ```js
3134. Job.js;
3135.
3136. import mongoose from "mongoose";
3137.
3138. const JobSchema = new mongoose.Schema(
3139.     {
3140.         company: {
3141.             type: String,
3142.             required: [true, "Please provide company name"],
3143.             maxlength: 50,
3144.         },
3145.         position: {
3146.             type: String,
3147.             required: [true, "Please provide position"],
3148.             maxlength: 100,
3149.         },
3150.         status: {
3151.             type: String,
3152.             enum: ["interview", "declined", "pending"],
3153.             default: "pending",
3154.         },
3155.
3156.         jobType: {
3157.             type: String,
3158.             enum: ["full-time", "part-time", "remote", "internship"],
3159.             default: "full-time",
3160.         },
3161.         jobLocation: {
3162.             type: String,
3163.             default: "my city",
3164.             required: true,
3165.         },

```

```

3166.         createdBy: {
3167.             type: mongoose.Types.ObjectId,
3168.             ref: "User",
3169.             required: [true, "Please provide user"],
3170.         },
3171.     },
3172.     { timestamps: true }
3173. );
3174.
3175. export default mongoose.model("Job", JobSchema);
3176. ```
3177.
3178. ##### Create Job
3179.
3180. ```js
3181. jobsController.js;
3182.
3183. import Job from "../models/Job.js";
3184. import { StatusCodes } from "http-status-codes";
3185. import { BadRequestError, NotFoundError } from "../errors/index.js";
3186.
3187. const createJob = async (req, res) => {
3188.     const { position, company } = req.body;
3189.
3190.     if (!position || !company) {
3191.         throw new BadRequestError("Please Provide All Values");
3192.     }
3193.
3194.     req.body.createdBy = req.user.userId;
3195.
3196.     const job = await Job.create(req.body);
3197.     res.status(StatusCodes.CREATED).json({ job });
3198. };
3199. ```
3200.
3201. ##### Job State Values
3202.
3203. ```js
3204. appContext.js;
3205. const initialState = {
3206.     isEditing: false,
3207.     editJobId: "",
3208.     position: "",
3209.     company: "",
3210.     // jobLocation
3211.     jobTypeOptions: ["full-time", "part-time", "remote", "internship"],
3212.     jobType: "full-time",
3213.     statusOptions: ["pending", "interview", "declined"],
3214.     status: "pending",
3215. };
3216. ```
3217.
3218. ##### AddJob Page - Setup
3219.
3220. ```js
3221. import { FormRow, Alert } from "../../components";
3222. import { useAppContext } from "../../context/appContext";
3223. import Wrapper from "../../assets/wrappers/DashboardFormPage";
3224. const AddJob = () => {
3225.     const {
3226.         isEditing,

```

```

3227.     showAlert,
3228.     displayAlert,
3229.     position,
3230.     company,
3231.     jobLocation,
3232.     jobType,
3233.     jobTypeOptions,
3234.     status,
3235.     statusOptions,
3236.   } = useAppContext();
3237.
3238.   const handleSubmit = (e) => {
3239.     e.preventDefault();
3240.
3241.     if (!position || !company || !jobLocation) {
3242.       displayAlert();
3243.       return;
3244.     }
3245.     console.log("create job");
3246.   };
3247.
3248.   const handleJobInput = (e) => {
3249.     const name = e.target.name;
3250.     const value = e.target.value;
3251.     console.log(`${name}:${value}`);
3252.   };
3253.
3254.   return (
3255.     <Wrapper>
3256.       <form className="form">
3257.         <h3>{isEditing ? "edit job" : "add job"} </h3>
3258.         {showAlert && <Alert />}
3259.
3260.         {/* position */}
3261.         <div className="form-center">
3262.           <FormRow
3263.             type="text"
3264.             name="position"
3265.             value={position}
3266.             handleChange={handleJobInput}
3267.           />
3268.           {/* company */}
3269.           <FormRow
3270.             type="text"
3271.             name="company"
3272.             value={company}
3273.             handleChange={handleJobInput}
3274.           />
3275.           {/* location */}
3276.           <FormRow
3277.             type="text"
3278.             labelText="location"
3279.             name="jobLocation"
3280.             value={jobLocation}
3281.             handleChange={handleJobInput}
3282.           />
3283.           {/* job type */}
3284.
3285.           {/* job status */}
3286.
3287.           <div className="btn-container">

```

```

3288.         <button
3289.             className="btn btn-block submit-btn"
3290.             type="submit"
3291.             onClick={handleSubmit}
3292.         >
3293.             submit
3294.         </button>
3295.     </div>
3296. </div>
3297. </form>
3298. </Wrapper>
3299. );
3300. };
3301.
3302. export default AddJob;
3303. ```
3304.
3305. ##### Select Input
3306.
3307. ```js
3308. return (
3309.     // job type
3310.     <div className="form-row">
3311.         <label htmlFor="jobType" className="form-label">
3312.             job type
3313.         </label>
3314.
3315.         <select
3316.             name="jobType"
3317.             value={jobType}
3318.             onChange={handleJobInput}
3319.             className="form-select"
3320.         >
3321.             {jobTypeOptions.map((itemValue, index) => {
3322.                 return (
3323.                     <option key={index} value={itemValue}>
3324.                         {itemValue}
3325.                     </option>
3326.                 );
3327.             })}
3328.         </select>
3329.     </div>
3330. );
3331. ```
3332.
3333. ##### FormRowSelect
3334.
3335. - create FormRowSelect in components
3336. - setup import/export
3337.
3338. ```js
3339. const FormRowSelect = ({ labelText, name, value, handleChange, list }) => {
3340.     return (
3341.         <div className="form-row">
3342.             <label htmlFor={name} className="form-label">
3343.                 {labelText || name}
3344.             </label>
3345.
3346.             <select
3347.                 name={name}
3348.                 value={value}

```

```

3349.         onChange={handleChange}
3350.         className="form-select"
3351.     >
3352.         {list.map((itemValue, index) => {
3353.             return (
3354.                 <option key={index} value={itemValue}>
3355.                     {itemValue}
3356.                 </option>
3357.             );
3358.         })}
3359.     </select>
3360. </div>
3361. );
3362. };
3363.
3364. export default FormRowSelect;
3365. ```
3366.
3367. ```js
3368. AddJob.js;
3369.
3370. return (
3371.     <>
3372.         {/* job status */}
3373.
3374.         <FormRowSelect
3375.             name="status"
3376.             value={status}
3377.             handleChange={handleJobInput}
3378.             list={statusOptions}
3379.         />
3380.
3381.         {/* job type */}
3382.         <FormRowSelect
3383.             labelText="type"
3384.             name="jobType"
3385.             value={jobType}
3386.             handleChange={handleJobInput}
3387.             list={jobTypeOptions}
3388.         />
3389.     </>
3390. );
3391. ```
3392.
3393. ##### Change State Values With Handle Change
3394.
3395. - [JS Nuggets Dynamic Object Keys](https://youtu.be/_qxCYtWm0tw)
3396.
3397. ```js
3398. actions.js;
3399.
3400. export const HANDLE_CHANGE = "HANDLE_CHANGE";
3401. ```
3402.
3403. ```js
3404. appContext.js
3405.
3406. const handleChange = ({ name, value }) => {
3407.     dispatch({
3408.         type: HANDLE_CHANGE,
3409.         payload: { name, value },

```



```

3410.     })
3411.   }
3412.
3413.   value={{handleChange}}
3414.   ```
3415.
3416.   ```js
3417.   reducer.js;
3418.
3419.   if (action.type === HANDLE_CHANGE) {
3420.     return { ...state, [action.payload.name]: action.payload.value };
3421.   }
3422.   ```
3423.
3424.   ```js
3425.   AddJob.js;
3426.
3427.   const { handleChange } = useAppContext();
3428.
3429.   const handleJobInput = (e) => {
3430.     handleChange({ name: e.target.name, value: e.target.value });
3431.   };
3432.   ```
3433.
3434.   ##### Clear Values
3435.
3436.   ```js
3437.   actions.js;
3438.
3439.   export const CLEAR_VALUES = "CLEAR_VALUES";
3440.   ```
3441.
3442.   ```js
3443.   appContext.js
3444.
3445.   const clearValues = () => {
3446.     dispatch({ type: CLEAR_VALUES })
3447.   }
3448.
3449.   value={{clearValues}}
3450.   ```
3451.
3452.   ```js
3453.   reducer.js;
3454.
3455.   if (action.type === CLEAR_VALUES) {
3456.     const initialState = {
3457.       isEditing: false,
3458.       editJobId: "",
3459.       position: "",
3460.       company: "",
3461.       jobLocation: state.userLocation,
3462.       jobType: "full-time",
3463.       status: "pending",
3464.     };
3465.     return { ...state, ...initialState };
3466.   }
3467.   ```
3468.
3469.   ```js
3470.   AddJob.js;

```

```

3471.
3472.     const { clearValues } = useAppContext();
3473.
3474.     return (
3475.         <div className="btn-container">
3476.             {/* submit button */}
3477.
3478.             <button
3479.                 className="btn btn-block clear-btn"
3480.                 onClick={(e) => {
3481.                     e.preventDefault();
3482.                     clearValues();
3483.                 }}
3484.             >
3485.                 clear
3486.             </button>
3487.         </div>
3488.     );
3489.     ...
3490.
3491.     #### Create Job
3492.
3493.     ```js
3494.     actions.js;
3495.
3496.     export const CREATE_JOB_BEGIN = "CREATE_JOB_BEGIN";
3497.     export const CREATE_JOB_SUCCESS = "CREATE_JOB_SUCCESS";
3498.     export const CREATE_JOB_ERROR = "CREATE_JOB_ERROR";
3499.     ...
3500.
3501.     ```js
3502.     appContext.js;
3503.
3504.     const createJob = async () => {
3505.         dispatch({ type: CREATE_JOB_BEGIN });
3506.         try {
3507.             const { position, company, jobLocation, jobType, status } = state;
3508.
3509.             await authFetch.post("/jobs", {
3510.                 company,
3511.                 position,
3512.                 jobLocation,
3513.                 jobType,
3514.                 status,
3515.             });
3516.             dispatch({
3517.                 type: CREATE_JOB_SUCCESS,
3518.             });
3519.             // call function instead clearValues()
3520.             dispatch({ type: CLEAR_VALUES });
3521.         } catch (error) {
3522.             if (error.response.status === 401) return;
3523.             dispatch({
3524.                 type: CREATE_JOB_ERROR,
3525.                 payload: { msg: error.response.data.msg },
3526.             });
3527.         }
3528.         clearAlert();
3529.     };
3530.     ...
3531.

```

```

3532.     ````js
3533.     AddJob.js;
3534.
3535.     const { createJob } = useAppContext();
3536.
3537.     const handleSubmit = (e) => {
3538.         e.preventDefault();
3539.         // while testing
3540.
3541.         // if (!position || !company || !jobLocation) {
3542.         //     displayAlert()
3543.         //     return
3544.         // }
3545.         if (isEditing) {
3546.             // eventually editJob()
3547.             return;
3548.         }
3549.         createJob();
3550.     };
3551.     ````
3552.
3553.     ````js
3554.     reducer.js;
3555.
3556.     if (action.type === CREATE_JOB_BEGIN) {
3557.         return { ...state, isLoading: true };
3558.     }
3559.     if (action.type === CREATE_JOB_SUCCESS) {
3560.         return {
3561.             ...state,
3562.             isLoading: false,
3563.             showAlert: true,
3564.             alertType: "success",
3565.             alertText: "New Job Created!",
3566.         };
3567.     }
3568.     if (action.type === CREATE_JOB_ERROR) {
3569.         return {
3570.             ...state,
3571.             isLoading: false,
3572.             showAlert: true,
3573.             alertType: "danger",
3574.             alertText: action.payload.msg,
3575.         };
3576.     }
3577.     ````
3578.
3579.     ##### Get All Jobs
3580.
3581.     ````js
3582.     jobsController.js;
3583.
3584.     const getAllJobs = async (req, res) => {
3585.         const jobs = await Job.find({ createdBy: req.user.userId });
3586.
3587.         res
3588.             .status(StatusCodes.OK)
3589.             .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
3590.     };
3591.     ````
3592.

```

```

3593.     ##### Jobs State Values
3594.
3595.     ```js
3596.     appContext.js;
3597.
3598.     const initialState = {
3599.         jobs: [],
3600.         totalJobs: 0,
3601.         numOfPages: 1,
3602.         page: 1,
3603.     };
3604.     ```
3605.
3606.     ##### Get All Jobs Request
3607.
3608.     ```js
3609.     actions.js;
3610.     export const GET_JOBS_BEGIN = "GET_JOBS_BEGIN";
3611.     export const GET_JOBS_SUCCESS = "GET_JOBS_SUCCESS";
3612.     ```
3613.
3614.     ```js
3615.     appContext.js
3616.
3617.     import React, { useReducer, useContext, useEffect } from 'react'
3618.
3619.     const getJobs = async () => {
3620.         let url = `/jobs`
3621.
3622.         dispatch({ type: GET_JOBS_BEGIN })
3623.         try {
3624.             const { data } = await authFetch(url)
3625.             const { jobs, totalJobs, numOfPages } = data
3626.             dispatch({
3627.                 type: GET_JOBS_SUCCESS,
3628.                 payload: {
3629.                     jobs,
3630.                     totalJobs,
3631.                     numOfPages,
3632.                 },
3633.             })
3634.         } catch (error) {
3635.             console.log(error.response)
3636.             logoutUser()
3637.         }
3638.         clearAlert()
3639.     }
3640.
3641.     useEffect(() => {
3642.         getJobs()
3643.     }, [])
3644.
3645.     value={{getJobs}}
3646.     ```
3647.
3648.
3649.     ```js
3650.     reducer.js;
3651.
3652.     if (action.type === GET_JOBS_BEGIN) {
3653.         return { ...state, isLoading: true, showAlert: false };

```

```

3654.     }
3655.     if (action.type === GET_JOBS_SUCCESS) {
3656.         return {
3657.             ...state,
3658.             isLoading: false,
3659.             jobs: action.payload.jobs,
3660.             totalJobs: action.payload.totalJobs,
3661.             numOfPages: action.payload.numOfPages,
3662.         };
3663.     }
3664.     ...
3665.
3666.     ##### AllJobs Page Setup
3667.
3668.     - create
3669.     - SearchContainer export
3670.     - JobsContainer export
3671.     - Job
3672.     - JobInfo
3673.
3674.     ```js
3675.     AllJobs.js;
3676.
3677.     import { JobsContainer, SearchContainer } from ".././components";
3678.     const AllJobs = () => {
3679.         return (
3680.             <>
3681.                 <SearchContainer />
3682.                 <JobsContainer />
3683.             </>
3684.         );
3685.     };
3686.
3687.     export default AllJobs;
3688.     ...
3689.
3690.     ```js
3691.     JobsContainer.js;
3692.     import { useAppContext } from "../context/appContext";
3693.     import { useEffect } from "react";
3694.     import Loading from "../Loading";
3695.     import Job from "../Job";
3696.     import Wrapper from "../assets/wrappers/JobsContainer";
3697.
3698.     const JobsContainer = () => {
3699.         const { getJobs, jobs, isLoading, page, totalJobs } = useAppContext();
3700.         useEffect(() => {
3701.             getJobs();
3702.         }, []);
3703.
3704.         if (isLoading) {
3705.             return <Loading center />;
3706.         }
3707.         if (jobs.length === 0) {
3708.             return (
3709.                 <Wrapper>
3710.                     <h2>No jobs to display...</h2>
3711.                 </Wrapper>
3712.             );
3713.         }
3714.         return (

```

```

3715.         <Wrapper>
3716.             <h5>
3717.                 {totalJobs} job{jobs.length > 1 && "s"} found
3718.             </h5>
3719.             <div className="jobs">
3720.                 {jobs.map((job) => {
3721.                     return <Job key={job._id} {...job} />;
3722.                 })}
3723.             </div>
3724.         </Wrapper>
3725.     );
3726. };
3727.
3728. export default JobsContainer;
3729. ```
3730.
3731. ```js
3732. Job.js;
3733.
3734. import moment from "moment";
3735.
3736. const Job = ({ company }) => {
3737.     return <h5>{company}</h5>;
3738. };
3739.
3740. export default Job;
3741. ```
3742.
3743. ##### Moment.js
3744.
3745. - Format Dates
3746. - [moment.js](https://momentjs.com/)
3747.
3748. - stop server
3749. - cd client
3750.
3751. ```sh
3752. npm install moment
3753.
3754. ```
3755.
3756. ```js
3757. Job.js;
3758.
3759. import moment from "moment";
3760.
3761. const Job = ({ company, createdAt }) => {
3762.     let date = moment(createdAt);
3763.     date = date.format("MMM Do, YYYY");
3764.     return (
3765.         <div>
3766.             <h5>{company}</h5>
3767.             <h5>{date}</h5>
3768.         </div>
3769.     );
3770. };
3771.
3772. export default Job;
3773. ```
3774.
3775. ##### Job Component - Setup

```

```

3776.
3777.   ```js
3778.   appContext.js
3779.
3780.   const setEditJob = (id) => {
3781.       console.log(`set edit job : ${id}`)
3782.   }
3783.   const deleteJob = (id) =>{
3784.       console.log(`delete : ${id}`)
3785.   }
3786.   value={{setEditJob,deleteJob}}
3787.   ```
3788.
3789.   ```js
3790.   Job.js;
3791.
3792.   import { FaLocationArrow, FaBriefcase, FaCalendarAlt } from "react-icons/fa";
3793.   import { Link } from "react-router-dom";
3794.   import { useAppContext } from "../context/appContext";
3795.   import Wrapper from "../assets/wrappers/Job";
3796.   import JobInfo from "./JobInfo";
3797.
3798.   const Job = ({
3799.       _id,
3800.       position,
3801.       company,
3802.       jobLocation,
3803.       jobType,
3804.       createdAt,
3805.       status,
3806.   }) => {
3807.       const { setEditJob, deleteJob } = useAppContext();
3808.
3809.       let date = moment(createdAt);
3810.       date = date.format("MMM Do, YYYY");
3811.
3812.       return (
3813.           <Wrapper>
3814.               <header>
3815.                   <div className="main-icon">{company.charAt(0)}</div>
3816.                   <div className="info">
3817.                       <h5>{position}</h5>
3818.                       <p>{company}</p>
3819.                   </div>
3820.               </header>
3821.               <div className="content">
3822.                   {/* content center later */}
3823.               <footer>
3824.                   <div className="actions">
3825.                       <Link
3826.                           to="/add-job"
3827.                           onClick={() => setEditJob(_id)}
3828.                           className="btn edit-btn"
3829.                       >
3830.                           Edit
3831.                       </Link>
3832.                       <button
3833.                           type="button"
3834.                           className="btn delete-btn"
3835.                           onClick={() => deleteJob(_id)}
3836.                       >

```

```

3837.             Delete
3838.             </button>
3839.         </div>
3840.     </footer>
3841. </div>
3842. </Wrapper>
3843. );
3844. };
3845.
3846. export default Job;
3847. ```
3848.
3849. ##### JobInfo
3850.
3851. ```js
3852. JobInfo.js;
3853.
3854. import Wrapper from "../assets/wrappers/JobInfo";
3855.
3856. const JobInfo = ({ icon, text }) => {
3857.     return (
3858.         <Wrapper>
3859.             <span className="icon">{icon}</span>
3860.             <span className="text">{text}</span>
3861.         </Wrapper>
3862.     );
3863. };
3864.
3865. export default JobInfo;
3866. ```
3867.
3868. ```js
3869. Job.js;
3870. return (
3871.     <div className="content">
3872.         <div className="content-center">
3873.             <JobInfo icon={<FaLocationArrow />} text={jobLocation} />
3874.             <JobInfo icon={<FaCalendarAlt />} text={date} />
3875.             <JobInfo icon={<FaBriefcase />} text={jobType} />
3876.             <div className={`status ${status}`}>{status}</div>
3877.         </div>
3878.         {/* footer content */}
3879.     </div>
3880. );
3881. ```
3882.
3883. ##### SetEditJob
3884.
3885. ```js
3886. actions.js;
3887. export const SET_EDIT_JOB = "SET_EDIT_JOB";
3888. ```
3889.
3890. ```js
3891. appContext.js
3892.
3893. const setEditJob = (id) => {
3894.     dispatch({ type: SET_EDIT_JOB, payload: { id } })
3895. }
3896. const editJob = () => {
3897.     console.log('edit job')

```



```
3898.     }
3899.     value={{editJob}}
3900.     ```
3901.
3902.     ```js
3903.     reducer.js;
3904.
3905.     if (action.type === SET_EDIT_JOB) {
3906.         const job = state.jobs.find((job) => job._id === action.payload.id);
3907.         const { _id, position, company, jobLocation, jobType, status } = job;
3908.         return {
3909.             ...state,
3910.             isEditing: true,
3911.             editJobId: _id,
3912.             position,
3913.             company,
3914.             jobLocation,
3915.             jobType,
3916.             status,
3917.         };
3918.     }
3919.     ```
3920.
3921.     ```js
3922.     AddJob.js;
3923.     const { isEditing, editJob } = useAppContext();
3924.     const handleSubmit = (e) => {
3925.         e.preventDefault();
3926.
3927.         if (!position || !company || !jobLocation) {
3928.             displayAlert();
3929.             return;
3930.         }
3931.         if (isEditing) {
3932.             editJob();
3933.             return;
3934.         }
3935.         createJob();
3936.     };
3937.     ```
3938.
3939.     ##### Edit Job - Server
3940.
3941.     ```js
3942.     jobsController.js;
3943.
3944.     const updateJob = async (req, res) => {
3945.         const { id: jobId } = req.params;
3946.
3947.         const { company, position } = req.body;
3948.
3949.         if (!company || !position) {
3950.             throw new BadRequestError("Please Provide All Values");
3951.         }
3952.
3953.         const job = await Job.findOne({ _id: jobId });
3954.
3955.         if (!job) {
3956.             throw new NotFoundError(`No job with id ${jobId}`);
3957.         }
3958.
```

```
3959.      // check permissions
3960.
3961.      const updatedJob = await Job.findOneAndUpdate({ _id: jobId }, req.body, {
3962.          new: true,
3963.          runValidators: true,
3964.      });
3965.
3966.      res.status(StatusCodes.OK).json({ updatedJob });
3967.  };
3968.  ```
```

```
3970.  ##### Alternative Approach
```

```
3971.
3972.  - optional
3973.  - multiple approaches
3974.  - different setups
3975.  - course Q&A
```

```
3976.
3977.  ```js
3978.  jobsController.js;
3979.  const updateJob = async (req, res) => {
3980.      const { id: jobId } = req.params;
3981.      const { company, position, jobLocation } = req.body;
3982.
3983.      if (!position || !company) {
3984.          throw new BadRequestError("Please provide all values");
3985.      }
3986.      const job = await Job.findOne({ _id: jobId });
3987.
3988.      if (!job) {
3989.          throw new NotFoundError(`No job with id :${jobId}`);
3990.      }
3991.
3992.      // check permissions
3993.
3994.      // alternative approach
3995.
3996.      job.position = position;
3997.      job.company = company;
3998.      job.jobLocation = jobLocation;
3999.
4000.      await job.save();
4001.      res.status(StatusCodes.OK).json({ job });
4002.  };
4003.  ```
```

```
4004.
4005.  ##### Check Permissions
```

```
4006.
4007.  ```js
4008.  jobsController.js;
4009.
4010.  const updateJob = async (req, res) => {
4011.      const { id: jobId } = req.params;
4012.      const { company, position, status } = req.body;
4013.
4014.      if (!position || !company) {
4015.          throw new BadRequestError("Please provide all values");
4016.      }
4017.      const job = await Job.findOne({ _id: jobId });
4018.
4019.      if (!job) {
```

```

4020.         throw new NotFoundError(`No job with id :${jobId}`);
4021.     }
4022.
4023.     // check permissions
4024.     // req.user.userId (string) === job.createdBy(object)
4025.     // throw new UnAuthenticatedError('Not authorized to access this route')
4026.
4027.     // console.log(typeof req.user.userId)
4028.     // console.log(typeof job.createdBy)
4029.
4030.     checkPermissions(req.user, job.createdBy);
4031.
4032.     const updatedJob = await Job.findOneAndUpdate({ _id: jobId }, req.body, {
4033.         new: true,
4034.         runValidators: true,
4035.     });
4036.
4037.     res.status(StatusCodes.OK).json({ updatedJob });
4038. };
4039. ```
4040.
4041. - utils folder
4042. - checkPermissions.js
4043. - import in jobsController.js
4044.
4045. ```js
4046. checkPermissions.js;
4047.
4048. import { UnAuthenticatedError } from "../errors/index.js";
4049.
4050. const checkPermissions = (requestUser, resourceUserId) => {
4051.     // if (requestUser.role === 'admin') return
4052.     if (requestUser.userId === resourceUserId.toString()) return;
4053.     throw new UnauthorizedError("Not authorized to access this route");
4054. };
4055.
4056. export default checkPermissions;
4057. ```
4058.
4059. ##### Remove/Delete Job
4060.
4061. ```js
4062. jobsController.js;
4063.
4064. const deleteJob = async (req, res) => {
4065.     const { id: jobId } = req.params;
4066.
4067.     const job = await Job.findOne({ _id: jobId });
4068.
4069.     if (!job) {
4070.         throw new NotFoundError(`No job with id : ${jobId}`);
4071.     }
4072.
4073.     checkPermissions(req.user, job.createdBy);
4074.
4075.     await job.remove();
4076.     res.status(StatusCodes.OK).json({ msg: "Success! Job removed" });
4077. };
4078. ```
4079.
4080. ##### Delete Job - Front-End

```

```

4081.
4082.   ```js
4083.   actions.js;
4084.
4085.   export const DELETE_JOB_BEGIN = "DELETE_JOB_BEGIN";
4086.   ```
4087.
4088.   ```js
4089.   appContext.js;
4090.
4091.   const deleteJob = async (jobId) => {
4092.     dispatch({ type: DELETE_JOB_BEGIN });
4093.     try {
4094.       await authFetch.delete(`/jobs/${jobId}`);
4095.       getJobs();
4096.     } catch (error) {
4097.       logoutUser();
4098.     }
4099.   };
4100.   ```
4101.
4102.   ```js
4103.   reducer.js;
4104.
4105.   if (action.type === DELETE_JOB_BEGIN) {
4106.     return { ...state, isLoading: true };
4107.   }
4108.   ```
4109.
4110.   #### Edit Job - Front-End
4111.
4112.   ```js
4113.   actions.js;
4114.   export const EDIT_JOB_BEGIN = "EDIT_JOB_BEGIN";
4115.   export const EDIT_JOB_SUCCESS = "EDIT_JOB_SUCCESS";
4116.   export const EDIT_JOB_ERROR = "EDIT_JOB_ERROR";
4117.   ```
4118.
4119.   ```js
4120.   appContext.js;
4121.   const editJob = async () => {
4122.     dispatch({ type: EDIT_JOB_BEGIN });
4123.     try {
4124.       const { position, company, jobLocation, jobType, status } = state;
4125.
4126.       await authFetch.patch(`/jobs/${state.editJobId}`, {
4127.         company,
4128.         position,
4129.         jobLocation,
4130.         jobType,
4131.         status,
4132.       });
4133.       dispatch({
4134.         type: EDIT_JOB_SUCCESS,
4135.       });
4136.       dispatch({ type: CLEAR_VALUES });
4137.     } catch (error) {
4138.       if (error.response.status === 401) return;
4139.       dispatch({
4140.         type: EDIT_JOB_ERROR,
4141.         payload: { msg: error.response.data.msg },

```

```

4142.     });
4143.   }
4144.   clearAlert();
4145. };
4146. ```
4147.
4148. ```js
4149. reducer.js;
4150.
4151. if (action.type === EDIT_JOB_BEGIN) {
4152.   return { ...state, isLoading: true };
4153. }
4154. if (action.type === EDIT_JOB_SUCCESS) {
4155.   return {
4156.     ...state,
4157.     isLoading: false,
4158.     showAlert: true,
4159.     alertType: "success",
4160.     alertText: "Job Updated!",
4161.   };
4162. }
4163. if (action.type === EDIT_JOB_ERROR) {
4164.   return {
4165.     ...state,
4166.     isLoading: false,
4167.     showAlert: true,
4168.     alertType: "danger",
4169.     alertText: action.payload.msg,
4170.   };
4171. }
4172. ```
4173.
4174. ##### Create More Jobs
4175.
4176. - [Mockaroo](https://www.mockaroo.com/)
4177. - setup mock-data.json in the root
4178.
4179. ##### Populate Database
4180.
4181. - create populate.js in the root
4182.
4183. ```js
4184. populate.js;
4185.
4186. import { readFile } from "fs/promises";
4187.
4188. import dotenv from "dotenv";
4189. dotenv.config();
4190.
4191. import connectDB from "../db/connect.js";
4192. import Job from "../models/Job.js";
4193.
4194. const start = async () => {
4195.   try {
4196.     await connectDB(process.env.MONGO_URL);
4197.     await Job.deleteMany();
4198.
4199.     const jsonProducts = JSON.parse(
4200.       await readFile(new URL("../mock-data.json", import.meta.url))
4201.     );
4202.     await Job.create(jsonProducts);

```

```

4203.         console.log("Success!!!!");
4204.         process.exit(0);
4205.     } catch (error) {
4206.         console.log(error);
4207.         process.exit(1);
4208.     }
4209. };
4210.
4211. start();
4212. ```
4213.
4214. ##### Show Stats - Structure
4215.
4216. - aggregation pipeline
4217. - step by step
4218. - [Aggregation Pipeline](https://docs.mongodb.com/manual/core/aggregation-pipeline/)
4219.
4220. ```js
4221. jobsController.js;
4222.
4223. import mongoose from "mongoose";
4224.
4225. const showStats = async (req, res) => {
4226.     let stats = await Job.aggregate([
4227.         { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
4228.         { $group: { _id: "$status", count: { $sum: 1 } } },
4229.     ]);
4230.
4231.     res.status(StatusCodes.OK).json({ stats });
4232. };
4233. ```
4234.
4235. ##### Show Stats - Object Setup
4236.
4237. - [Reduce Basics](https://youtu.be/3Wkw9nrS2mw)
4238. - [Reduce Object Example ](https://youtu.be/5BFkp8JjLEY)
4239.
4240. ```js
4241. jobsController.js;
4242.
4243. const showStats = async (req, res) => {
4244.     let stats = await Job.aggregate([
4245.         { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
4246.         { $group: { _id: "$status", count: { $sum: 1 } } },
4247.     ]);
4248.
4249.     stats = stats.reduce((acc, curr) => {
4250.         const { _id: title, count } = curr;
4251.         acc[title] = count;
4252.         return acc;
4253.     }, {});
4254.
4255.     res.status(StatusCodes.OK).json({ stats });
4256. };
4257. ```
4258.
4259. ##### Show Stats - Default Stats
4260.
4261. ```js
4262. jobsController.js;
4263.

```

```

4264.     const showStats = async (req, res) => {
4265.         let stats = await Job.aggregate([
4266.             { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
4267.             { $group: { _id: "$status", count: { $sum: 1 } } },
4268.         ]);
4269.         stats = stats.reduce((acc, curr) => {
4270.             const { _id: title, count } = curr;
4271.             acc[title] = count;
4272.             return acc;
4273.         }, {});
4274.
4275.         const defaultStats = {
4276.             pending: stats.pending || 0,
4277.             interview: stats.interview || 0,
4278.             declined: stats.declined || 0,
4279.         };
4280.         let monthlyApplications = [];
4281.         res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications });
4282.     };
4283.     ...
4284.
4285.     ##### Show Stats - Function Setup
4286.
4287.     ```js
4288.     actions.js;
4289.
4290.     export const SHOW_STATS_BEGIN = "SHOW_STATS_BEGIN";
4291.     export const SHOW_STATS_SUCCESS = "SHOW_STATS_SUCCESS";
4292.     ...
4293.
4294.     ```js
4295.     appContext.js
4296.
4297.     const initialState = {
4298.         stats: {},
4299.         monthlyApplications: []
4300.     }
4301.
4302.
4303.     const showStats = async () => {
4304.         dispatch({ type: SHOW_STATS_BEGIN })
4305.         try {
4306.             const { data } = await authFetch('/jobs/stats')
4307.             dispatch({
4308.                 type: SHOW_STATS_SUCCESS,
4309.                 payload: {
4310.                     stats: data.defaultStats,
4311.                     monthlyApplications: data.monthlyApplications,
4312.                 },
4313.             })
4314.         } catch (error) {
4315.             console.log(error.response)
4316.             // logoutUser()
4317.         }
4318.
4319.         clearAlert()
4320.     }
4321.     value={{showStats}}
4322.     ...
4323.
4324.     ```js

```

```

4325.     reducers.js;
4326.     if (action.type === SHOW_STATS_BEGIN) {
4327.         return { ...state, isLoading: true, showAlert: false };
4328.     }
4329.     if (action.type === SHOW_STATS_SUCCESS) {
4330.         return {
4331.             ...state,
4332.             isLoading: false,
4333.             stats: action.payload.stats,
4334.             monthlyApplications: action.payload.monthlyApplications,
4335.         };
4336.     }
4337.     ...
4338.
4339.     ##### Stats Page - Structure
4340.
4341.     - components
4342.     - StatsContainer.js
4343.     - ChartsContainer.js
4344.     - StatsItem.js
4345.     - simple return
4346.     - import/export index.js
4347.
4348.     ```js
4349.     Stats.js;
4350.
4351.     import { useEffect } from "react";
4352.     import { useAppContext } from "../context/appContext";
4353.     import { StatsContainer, Loading, ChartsContainer } from "../components";
4354.
4355.     const Stats = () => {
4356.         const { showStats, isLoading, monthlyApplications } = useAppContext();
4357.         useEffect(() => {
4358.             showStats();
4359.         }, []);
4360.
4361.         if (isLoading) {
4362.             return <Loading center />;
4363.         }
4364.
4365.         return (
4366.             <>
4367.                 <StatsContainer />
4368.                 {monthlyApplications.length > 0 && <ChartsContainer />}
4369.             </>
4370.         );
4371.     };
4372.
4373.     export default Stats;
4374.     ...
4375.
4376.     ##### StatsContainer
4377.
4378.     ```js
4379.     StatsContainer.js;
4380.
4381.     import { useAppContext } from "../context/appContext";
4382.     import StatItem from "../StatItem";
4383.     import { FaSuitcaseRolling, FaCalendarCheck, FaBug } from "react-icons/fa";
4384.     import Wrapper from "../assets/wrappers/StatsContainer";
4385.     const StatsContainer = () => {

```



```

4386.     const { stats } = useAppContext();
4387.     const defaultStats = [
4388.       {
4389.         title: "pending applications",
4390.         count: stats.pending || 0,
4391.         icon: <FaSuitcaseRolling />,
4392.         color: "#e9b949",
4393.         bcg: "#fcefc7",
4394.       },
4395.       {
4396.         title: "interviews scheduled",
4397.         count: stats.interview || 0,
4398.         icon: <FaCalendarCheck />,
4399.         color: "#647acb",
4400.         bcg: "#e0e8f9",
4401.       },
4402.       {
4403.         title: "jobs declined",
4404.         count: stats.declined || 0,
4405.         icon: <FaBug />,
4406.         color: "#d66a6a",
4407.         bcg: "#ffeeee",
4408.       },
4409.     ];
4410.
4411.     return (
4412.       <Wrapper>
4413.         {defaultStats.map((item, index) => {
4414.           return <StatItem key={index} {...item} />;
4415.         })}
4416.       </Wrapper>
4417.     );
4418.   };
4419.
4420.   export default StatsContainer;
4421.   ```
4422.
4423.   ##### StatItem
4424.
4425.   ```js
4426.   StatItem.js;
4427.
4428.   import Wrapper from "../assets/wrappers/StatItem";
4429.
4430.   function StatItem({ count, title, icon, color, bcg }) {
4431.     return (
4432.       <Wrapper color={color} bcg={bcg}>
4433.         <header>
4434.           <span className="count">{count}</span>
4435.           <div className="icon">{icon}</div>
4436.         </header>
4437.         <h5 className="title">{title}</h5>
4438.       </Wrapper>
4439.     );
4440.   }
4441.
4442.   export default StatItem;
4443.   ```
4444.
4445.   ##### Aggregate Jobs Based on Year and Month
4446.

```

```

4447.   ```js
4448.   jobsController.js;
4449.
4450.   let monthlyApplications = await Job.aggregate([
4451.     { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
4452.     {
4453.       $group: {
4454.         _id: {
4455.           year: {
4456.             $year: "$createdAt",
4457.           },
4458.           month: {
4459.             $month: "$createdAt",
4460.           },
4461.         },
4462.         count: { $sum: 1 },
4463.       },
4464.     },
4465.     { $sort: { "_id.year": -1, "_id.month": -1 } },
4466.     { $limit: 6 },
4467.   ]);
4468.   ```
4469.
4470.   ##### Refactor Data
4471.
4472.   - install moment.js on the SERVER
4473.
4474.   ```sh
4475.   npm install moment
4476.
4477.   ```
4478.
4479.   ```js
4480.   jobsController.js;
4481.
4482.   import moment from "moment";
4483.
4484.   monthlyApplications = monthlyApplications
4485.     .map((item) => {
4486.       const {
4487.         _id: { year, month },
4488.         count,
4489.       } = item;
4490.       // accepts 0-11
4491.       const date = moment()
4492.         .month(month - 1)
4493.         .year(year)
4494.         .format("MMM Y");
4495.       return { date, count };
4496.     })
4497.     .reverse();
4498.   ```
4499.
4500.   ##### Charts Container
4501.
4502.   - BarChart.js
4503.   - AreaChart.js
4504.
4505.   ```js
4506.   ChartsContainer.js;
4507.   import React, { useState } from "react";

```

```

4508.
4509.   import BarChart from "./BarChart";
4510.   import AreaChart from "./AreaChart";
4511.   import { useAppContext } from "../context/appContext";
4512.   import Wrapper from "../assets/wrappers/ChartsContainer";
4513.
4514.   export default function ChartsContainer() {
4515.     const [barChart, setBarChart] = useState(true);
4516.     const { monthlyApplications: data } = useAppContext();
4517.
4518.     return (
4519.       <Wrapper>
4520.         <h4>Monthly Applications</h4>
4521.
4522.         <button type="button" onClick={() => setBarChart(!barChart)}>
4523.           {barChart ? "AreaChart" : "BarChart"}
4524.         </button>
4525.         {barChart ? <BarChart data={data} /> : <AreaChart data={data} />}
4526.       </Wrapper>
4527.     );
4528.   }
4529.   ...
4530.
4531.   ##### Recharts Library
4532.
4533.   - install in the Client!!!
4534.
4535.   [Recharts](https://recharts.org)
4536.
4537.   ```sh
4538.   npm install recharts
4539.   ```
4540.
4541.   ##### Bar Chart
4542.
4543.   ```js
4544.   BarChart.js;
4545.
4546.   import {
4547.     BarChart,
4548.     Bar,
4549.     XAxis,
4550.     YAxis,
4551.     CartesianGrid,
4552.     Tooltip,
4553.     ResponsiveContainer,
4554.   } from "recharts";
4555.
4556.   const BarChartComponent = ({ data }) => {
4557.     return (
4558.       <ResponsiveContainer width="100%" height={300}>
4559.         <BarChart
4560.           data={data}
4561.           margin={{
4562.             top: 50,
4563.           }}
4564.         >
4565.           <CartesianGrid strokeDasharray="3 3" />
4566.           <XAxis dataKey="date" />
4567.           <YAxis allowDecimals={false} />
4568.           <Tooltip />

```

```

4569.         <Bar dataKey="count" fill="#2cb1bc" barSize={75} />
4570.     </BarChart>
4571. </ResponsiveContainer>
4572. );
4573. };
4574. ...
4575.
4576. ##### Area Chart
4577.
4578. ```js
4579. import {
4580.     ResponsiveContainer,
4581.     AreaChart,
4582.     Area,
4583.     XAxis,
4584.     YAxis,
4585.     CartesianGrid,
4586.     Tooltip,
4587. } from "recharts";
4588.
4589. const AreaChartComponent = ({ data }) => {
4590.     return (
4591.         <ResponsiveContainer width="100%" height={300}>
4592.             <AreaChart
4593.                 data={data}
4594.                 margin={{
4595.                     top: 50,
4596.                 }}
4597.             >
4598.                 <CartesianGrid strokeDasharray="3 3" />
4599.                 <XAxis dataKey="date" />
4600.                 <YAxis allowDecimals={false} />
4601.                 <Tooltip />
4602.                 <Area type="monotone" dataKey="count" stroke="#2cb1bc" fill="#bef8fd" />
4603.             </AreaChart>
4604.         </ResponsiveContainer>
4605.     );
4606. };
4607. ...
4608.
4609. ##### Filter
4610.
4611. ##### Get All Jobs - Initial Setup
4612.
4613. ```js
4614. jobsController.js;
4615.
4616. const getAllJobs = async (req, res) => {
4617.     const { search, status, jobType, sort } = req.query;
4618.
4619.     const queryObject = {
4620.         createdBy: req.user.userId,
4621.     };
4622.
4623.     // NO AWAIT
4624.     let result = Job.find(queryObject);
4625.
4626.     // chain sort conditions
4627.
4628.     const jobs = await result;
4629.

```

```

4630.         res
4631.             .status(StatusCodes.OK)
4632.             .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
4633.     });
4634.     ...
4635.
4636.     ##### Status
4637.
4638.     ```js
4639.     jobsController.js;
4640.
4641.     const getAllJobs = async (req, res) => {
4642.         const { search, status, jobType, sort } = req.query;
4643.
4644.         const queryObject = {
4645.             createdBy: req.user.userId,
4646.         };
4647.
4648.         if (status !== "all") {
4649.             queryObject.status = status;
4650.         }
4651.
4652.         // NO AWAIT
4653.         let result = Job.find(queryObject);
4654.
4655.         // chain sort conditions
4656.
4657.         const jobs = await result;
4658.
4659.         res
4660.             .status(StatusCodes.OK)
4661.             .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
4662.     });
4663.     ...
4664.
4665.     ##### JobType
4666.
4667.     ```js
4668.     jobsController.js;
4669.
4670.     const getAllJobs = async (req, res) => {
4671.         const { search, status, jobType, sort } = req.query;
4672.
4673.         const queryObject = {
4674.             createdBy: req.user.userId,
4675.         };
4676.
4677.         if (status !== "all") {
4678.             queryObject.status = status;
4679.         }
4680.         if (jobType !== "all") {
4681.             queryObject.jobType = jobType;
4682.         }
4683.         // NO AWAIT
4684.         let result = Job.find(queryObject);
4685.
4686.         // chain sort conditions
4687.
4688.         const jobs = await result;
4689.
4690.         res

```

```

4691.         .status(StatusCodes.OK)
4692.         .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
4693.     };
4694.     ...
4695.
4696.     ##### Search
4697.
4698.     ```js
4699.     jobsController.js;
4700.
4701.     const getAllJobs = async (req, res) => {
4702.         const { search, status, jobType, sort } = req.query;
4703.
4704.         const queryObject = {
4705.             createdBy: req.user.userId,
4706.         };
4707.
4708.         if (status !== "all") {
4709.             queryObject.status = status;
4710.         }
4711.         if (jobType !== "all") {
4712.             queryObject.jobType = jobType;
4713.         }
4714.         if (search) {
4715.             queryObject.position = { $regex: search, $options: "i" };
4716.         }
4717.         // NO AWAIT
4718.         let result = Job.find(queryObject);
4719.
4720.         // chain sort conditions
4721.         if (sort === "latest") {
4722.             result = result.sort("-createdAt");
4723.         }
4724.         if (sort === "oldest") {
4725.             result = result.sort("createdAt");
4726.         }
4727.         if (sort === "a-z") {
4728.             result = result.sort("position");
4729.         }
4730.         if (sort === "z-a") {
4731.             result = result.sort("-position");
4732.         }
4733.         const jobs = await result;
4734.
4735.         res
4736.             .status(StatusCodes.OK)
4737.             .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
4738.     };
4739.     ...
4740.
4741.     ##### Search Context Setup
4742.
4743.     ```js
4744.     appContext.js
4745.
4746.     const initialState = {
4747.         jobType: 'full-time',
4748.         jobTypeOptions: ['full-time', 'part-time', 'remote', 'internship'],
4749.         status: 'pending',
4750.         statusOptions: ['pending', 'interview', 'declined']
4751.     };

```

```

4752.      //
4753.      //
4754.      search: '',
4755.      searchStatus: 'all',
4756.      searchType: 'all',
4757.      sort: 'latest',
4758.      sortOptions: ['latest', 'oldest', 'a-z', 'z-a'],
4759.    }
4760.
4761.    const clearFilters = () =>{
4762.      console.log('clear filters')
4763.    }
4764.
4765.    value={{clearFilters}}
4766.
4767.    // remember this function :)
4768.    const handleChange = ({ name, value }) => {
4769.      dispatch({
4770.        type: HANDLE_CHANGE,
4771.        payload: { name, value },
4772.      })
4773.    }
4774.
4775.    ```
4776.
4777.    ##### Search Container - Setup
4778.
4779.    ```js
4780.    SearchContainer.js;
4781.
4782.    import { FormRow, FormRowSelect } from ".";
4783.    import { useAppContext } from "../context/appContext";
4784.    import Wrapper from "../assets/wrappers/SearchContainer";
4785.    const SearchContainer = () => {
4786.      const {
4787.        isLoading,
4788.        search,
4789.        searchStatus,
4790.        searchType,
4791.        sort,
4792.        sortOptions,
4793.        statusOptions,
4794.        jobTypeOptions,
4795.        handleChange,
4796.        clearFilters,
4797.      } = useAppContext();
4798.
4799.      const handleSearch = (e) => {
4800.        if (isLoading) return;
4801.        handleChange({ name: e.target.name, value: e.target.value });
4802.      };
4803.
4804.      return (
4805.        <Wrapper>
4806.          <form className="form">
4807.            <h4>search form</h4>
4808.            {/* search position */}
4809.            <div className="form-center">
4810.              <FormRow
4811.                type="text"
4812.                name="search"

```

```

4813.         value={search}
4814.         handleChange={handleSearch}
4815.       ></FormRow>
4816.       {/* rest of the inputs */}
4817.     </div>
4818.   </form>
4819. </Wrapper>
4820.   );
4821. };
4822.
4823. export default SearchContainer;
4824. ```
4825.
4826. ##### Search Container - Complete
4827.
4828. ```js
4829. SearchContainer.js;
4830.
4831. import { FormRow, FormRowSelect } from ".";
4832. import { useAppContext } from "../context/appContext";
4833. import Wrapper from "../assets/wrappers/SearchContainer";
4834.
4835. const SearchContainer = () => {
4836.   const {
4837.     isLoading,
4838.     search,
4839.     handleChange,
4840.     searchStatus,
4841.     statusOptions,
4842.     jobTypeOptions,
4843.     searchType,
4844.     clearFilters,
4845.     sort,
4846.     sortOptions,
4847.   } = useAppContext();
4848.
4849.   const handleSearch = (e) => {
4850.     if (isLoading) return;
4851.     handleChange({ name: e.target.name, value: e.target.value });
4852.   };
4853.   const handleSubmit = (e) => {
4854.     e.preventDefault();
4855.     clearFilters();
4856.   };
4857.   return (
4858.     <Wrapper>
4859.       <form className="form">
4860.         <h4>search form</h4>
4861.         {/* search position */}
4862.         <div className="form-center">
4863.           <FormRow
4864.             type="text"
4865.             name="search"
4866.             value={search}
4867.             handleChange={handleSearch}
4868.           ></FormRow>
4869.           {/* search by status */}
4870.           <FormRowSelect
4871.             labelText="job status"
4872.             name="searchStatus"
4873.             value={searchStatus}

```



```

4874.             handleChange={handleSearch}
4875.             list={["all", ...statusOptions]}
4876.         ></FormRowSelect>
4877.         {/* search by type */}
4878.
4879.         <FormRowSelect
4880.             labelText="job type"
4881.             name="searchType"
4882.             value={searchType}
4883.             handleChange={handleSearch}
4884.             list={["all", ...jobTypeOptions]}
4885.         ></FormRowSelect>
4886.         {/* sort */}
4887.
4888.         <FormRowSelect
4889.             name="sort"
4890.             value={sort}
4891.             handleChange={handleSearch}
4892.             list={sortOptions}
4893.         ></FormRowSelect>
4894.         <button
4895.             className="btn btn-block btn-danger"
4896.             disabled={isLoading}
4897.             onClick={handleSubmit}
4898.         >
4899.             clear filters
4900.         </button>
4901.     </div>
4902. </form>
4903. </Wrapper>
4904. );
4905. };
4906.
4907. export default SearchContainer;
4908.
4909.
4910. ##### Clear Filters
4911.
4912. ```js
4913. actions.js;
4914.
4915. export const CLEAR_FILTERS = "CLEAR_FILTERS";
4916.
4917.
4918. ```js
4919. appContext.js;
4920.
4921. const clearFilters = () => {
4922.     dispatch({ type: CLEAR_FILTERS });
4923. };
4924.
4925.
4926. ```js
4927. reducer.js;
4928.
4929. if (action.type === CLEAR_FILTERS) {
4930.     return {
4931.         ...state,
4932.         search: "",
4933.         searchStatus: "all",
4934.         searchType: "all",

```

```

4935.         sort: "latest",
4936.     };
4937. }
4938. ...
4939.
4940. ##### Refactor Get All Jobs
4941.
4942. ```js
4943. const getJobs = async () => {
4944.     // will add page later
4945.     const { search, searchStatus, searchType, sort } = state;
4946.     let url = `/jobs?status=${searchStatus}&jobType=${searchType}&sort=${sort}`;
4947.     if (search) {
4948.         url = url + `&search=${search}`;
4949.     }
4950.     dispatch({ type: GET_JOBS_BEGIN });
4951.     try {
4952.         const { data } = await authFetch(url);
4953.         const { jobs, totalJobs, numOfPages } = data;
4954.         dispatch({
4955.             type: GET_JOBS_SUCCESS,
4956.             payload: {
4957.                 jobs,
4958.                 totalJobs,
4959.                 numOfPages,
4960.             },
4961.         });
4962.     } catch (error) {
4963.         // logoutUser()
4964.     }
4965.     clearAlert();
4966. };
4967. ...
4968.
4969. ```js
4970. JobsContainer.js
4971.
4972. const JobsContainer = () => {
4973.     const {
4974.         getJobs,
4975.         jobs,
4976.         isLoading,
4977.         page,
4978.         totalJobs,
4979.         search,
4980.         searchStatus,
4981.         searchType,
4982.         sort,
4983.     } = useAppContext()
4984.     useEffect(() => {
4985.         getJobs()
4986.     }, [search, searchStatus, searchType, sort])
4987. }
4988. ...
4989.
4990. ##### Limit and Skip
4991.
4992. ```js
4993. jobsController.js;
4994.
4995.

```

```

4996.     const getAllJobs = async (req, res) => {
4997.         const { search, status, jobType, sort } = req.query;
4998.         const queryObject = {
4999.             createdBy: req.user.userId,
5000.         };
5001.         if (search) {
5002.             queryObject.position = { $regex: search, $options: "i" };
5003.         }
5004.         if (status !== "all") {
5005.             queryObject.status = status;
5006.         }
5007.         if (jobType !== "all") {
5008.             queryObject.jobType = jobType;
5009.         }
5010.         let result = Job.find(queryObject);
5011.
5012.         if (sort === "latest") {
5013.             result = result.sort("-createdAt");
5014.         }
5015.         if (sort === "oldest") {
5016.             result = result.sort("createdAt");
5017.         }
5018.         if (sort === "a-z") {
5019.             result = result.sort("position");
5020.         }
5021.         if (sort === "z-a") {
5022.             result = result.sort("-position");
5023.         }
5024.
5025.         const totalJobs = await result;
5026.
5027.         // setup pagination
5028.         const limit = 10;
5029.         const skip = 1;
5030.
5031.         result = result.skip(skip).limit(limit);
5032.         // 23
5033.         // 4 7 7 7 2
5034.         const jobs = await result;
5035.         res
5036.             .status(StatusCodes.OK)
5037.             .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
5038.     };
5039.
5040.
5041.     ##### Page and Limit
5042.
5043.     ```js
5044.     jobsController.js;
5045.
5046.     const getAllJobs = async (req, res) => {
5047.         const { search, status, jobType, sort } = req.query;
5048.         const queryObject = {
5049.             createdBy: req.user.userId,
5050.         };
5051.         if (search) {
5052.             queryObject.position = { $regex: search, $options: "i" };
5053.         }
5054.         if (status !== "all") {
5055.             queryObject.status = status;
5056.         }

```

```

5057.     if (jobType !== "all") {
5058.         queryObject.jobType = jobType;
5059.     }
5060.     let result = Job.find(queryObject);
5061.
5062.     if (sort === "latest") {
5063.         result = result.sort("-createdAt");
5064.     }
5065.     if (sort === "oldest") {
5066.         result = result.sort("createdAt");
5067.     }
5068.     if (sort === "a-z") {
5069.         result = result.sort("position");
5070.     }
5071.     if (sort === "z-a") {
5072.         result = result.sort("-position");
5073.     }
5074.
5075.     // setup pagination
5076.     const page = Number(req.query.page) || 1;
5077.     const limit = Number(req.query.limit) || 10;
5078.     const skip = (page - 1) * limit; //10
5079.     result = result.skip(skip).limit(limit);
5080.     // 75
5081.     // 10 10 10 10 10 10 10 5
5082.     const jobs = await result;
5083.     res
5084.         .status(StatusCodes.OK)
5085.         .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
5086.     };
5087.     ``
5088.
5089.     ##### Total Jobs and Number Of Pages
5090.
5091.     ```js
5092.     jobsController.js;
5093.
5094.     const getAllJobs = async (req, res) => {
5095.         const { search, status, jobType, sort } = req.query;
5096.         const queryObject = {
5097.             createdBy: req.user.userId,
5098.         };
5099.         if (search) {
5100.             queryObject.position = { $regex: search, $options: "i" };
5101.         }
5102.         if (status !== "all") {
5103.             queryObject.status = status;
5104.         }
5105.         if (jobType !== "all") {
5106.             queryObject.jobType = jobType;
5107.         }
5108.         let result = Job.find(queryObject);
5109.
5110.         if (sort === "latest") {
5111.             result = result.sort("-createdAt");
5112.         }
5113.         if (sort === "oldest") {
5114.             result = result.sort("createdAt");
5115.         }
5116.         if (sort === "a-z") {
5117.             result = result.sort("position");

```

```

5118.     }
5119.     if (sort === "z-a") {
5120.         result = result.sort("-position");
5121.     }
5122.
5123.     // setup pagination
5124.     const page = Number(req.query.page) || 1;
5125.     const limit = Number(req.query.limit) || 10;
5126.     const skip = (page - 1) * limit;
5127.
5128.     result = result.skip(skip).limit(limit);
5129.
5130.     const jobs = await result;
5131.
5132.     const totalJobs = await Job.countDocuments(queryObject);
5133.     const numOfPages = Math.ceil(totalJobs / limit);
5134.
5135.     res.status(StatusCodes.OK).json({ jobs, totalJobs, numOfPages });
5136. };
5137. ```
5138.
5139. ##### PageBtnContainer Setup
5140.
5141. - PageBtnContainer.js
5142.
5143. ```js
5144. JobsContainer.js;
5145.
5146. import PageBtnContainer from "../PageBtnContainer";
5147.
5148. const { numOfPages } = useAppContext();
5149.
5150. return (
5151.     <Wrapper>
5152.         <h5>
5153.             {totalJobs} job{jobs.length > 1 && "s"} found
5154.         </h5>
5155.         <div className="jobs">
5156.             {jobs.map((job) => {
5157.                 return <Job key={job._id} {...job} />;
5158.             })}
5159.         </div>
5160.         {numOfPages > 1 && <PageBtnContainer />}
5161.     </Wrapper>
5162. );
5163. ```
5164.
5165. ##### PageBtnContainer - Structure
5166.
5167. ```js
5168. PageBtnContainer.js;
5169.
5170. import { useAppContext } from "../context/appContext";
5171. import { HiChevronDoubleLeft, HiChevronDoubleRight } from "react-icons/hi";
5172. import Wrapper from "../assets/wrappers/PageBtnContainer";
5173.
5174. const PageButtonContainer = () => {
5175.     const { numOfPages, page } = useAppContext();
5176.
5177.     const prevPage = () => {
5178.         console.log("prev page");

```

```

5179.     };
5180.     const nextPage = () => {
5181.         console.log("next page");
5182.     };
5183.
5184.     return (
5185.         <Wrapper>
5186.             <button className="prev-btn" onClick={prevPage}>
5187.                 <HiChevronDoubleLeft />
5188.                 prev
5189.             </button>
5190.
5191.             <div className="btn-container">buttons</div>
5192.
5193.             <button className="next-btn" onClick={nextPage}>
5194.                 next
5195.                 <HiChevronDoubleRight />
5196.             </button>
5197.         </Wrapper>
5198.     );
5199. };
5200.
5201. export default PageButtonContainer;
5202. ```
5203.
5204. ##### Button Container
5205.
5206. - [Array.from] (https://youtu.be/zg1Bv4xubwo)
5207.
5208. ```js
5209. PageBtnContainer.js;
5210.
5211. const pages = Array.from({ length: numOfPages }, (_, index) => {
5212.     return index + 1;
5213. });
5214.
5215. return (
5216.     <div className="btn-container">
5217.         {pages.map((pageNumber) => {
5218.             return (
5219.                 <button
5220.                     type="button"
5221.                     className={pageNumber === page ? "pageBtn active" : "pageBtn"}
5222.                     key={pageNumber}
5223.                     onClick={() => console.log(page)}
5224.                 >
5225.                     {pageNumber}
5226.                 </button>
5227.             );
5228.         })}
5229.     </div>
5230. );
5231. ```
5232.
5233. ##### Change Page
5234.
5235. ```js
5236. actions.js;
5237. export const CHANGE_PAGE = "CHANGE_PAGE";
5238. ```
5239.

```

```

5240.     ````js
5241.     appContext.js
5242.     const changePage = (page) => {
5243.         dispatch({ type: CHANGE_PAGE, payload: { page } })
5244.     }
5245.     value={{changePage}}
5246.     ````
5247.
5248.     ````js
5249.     reducer.js;
5250.
5251.     if (action.type === CHANGE_PAGE) {
5252.         return { ...state, page: action.payload.page };
5253.     }
5254.     ````
5255.
5256.     ````js
5257.     PageBtnContainer.js;
5258.
5259.     const { changePage } = useAppContext();
5260.     return (
5261.         <button
5262.             type="button"
5263.             className={pageNumber === page ? "pageBtn active" : "pageBtn"}
5264.             key={pageNumber}
5265.             onClick={() => changePage(pageNumber)}
5266.         >
5267.             {pageNumber}
5268.         </button>
5269.     );
5270.     ````
5271.
5272.     ##### Prev and Next Buttons
5273.
5274.     ````js
5275.     PageBtnContainer.js;
5276.     const prevPage = () => {
5277.         let newPage = page - 1;
5278.         if (newPage < 1) {
5279.             // newPage = 1
5280.             // alternative
5281.             newPage = numOfPages;
5282.         }
5283.         changePage(newPage);
5284.     };
5285.     const nextPage = () => {
5286.         let newPage = page + 1;
5287.         if (newPage > numOfPages) {
5288.             // newPage = numOfPages
5289.             // alternative
5290.             newPage = 1;
5291.         }
5292.         changePage(newPage);
5293.     };
5294.     ````
5295.
5296.     ##### Trigger New Page
5297.
5298.     ````js
5299.     appContext.js;
5300.

```

```

5301.     const getJobs = async () => {
5302.         const { page, search, searchStatus, searchType, sort } = state;
5303.
5304.         let url =
5305.             `/jobs?page=${page}&status=${searchStatus}&jobType=${searchType}&sort=${sort}`;
5306.         // rest of the code
5307.     };
5308.     ...
5309.     ```js
5310.     JobsContainer.js;
5311.
5312.     const { page } = useAppContext();
5313.     useEffect(() => {
5314.         getJobs();
5315.     }, [page, search, searchStatus, searchType, sort]);
5316.     ...
5317.
5318.     ```js
5319.     reducer.js;
5320.
5321.     if (action.type === HANDLE_CHANGE) {
5322.         // set back to first page
5323.
5324.         return { ...state, page: 1, [action.payload.name]: action.payload.value };
5325.     }
5326.     ...
5327.
5328.     ##### Production Setup - Fix Warnings and logoutUser
5329.
5330.     - getJobs,deleteJob,showStats - invoke logoutUser()
5331.     - fix warnings
5332.
5333.     ```sh
5334.     // eslint-disable-next-line
5335.     ...
5336.
5337.     ##### Production Setup - Build Front-End Application
5338.
5339.     - create front-end production application
5340.
5341.     ```js
5342.     package.json
5343.     "scripts": {
5344.         "build-client": "cd client && npm run build",
5345.         "server": "nodemon server.js --ignore client",
5346.         "client": "cd client && npm run start",
5347.         "start": "concurrently --kill-others-on-fail \"npm run server\" \"npm run client\""
5348.     },
5349.     ...
5350.
5351.     ...
5352.
5353.     ```js
5354.     server.js;
5355.
5356.     import { dirname } from "path";
5357.     import { fileURLToPath } from "url";
5358.     import path from "path";
5359.
5360.     const __dirname = dirname(fileURLToPath(import.meta.url));

```



```

5361.
5362.    // only when ready to deploy
5363.    app.use(express.static(path.resolve(__dirname, "./client/build")));
5364.
5365.    // routes
5366.    app.use("/api/v1/auth", authRouter);
5367.    app.use("/api/v1/jobs", authenticateUser, jobsRouter);
5368.
5369.    // only when ready to deploy
5370.    app.get("*", function (request, response) {
5371.        response.sendFile(path.resolve(__dirname, "./client/build", "index.html"));
5372.    });
5373.    ```
5374.
5375.    ##### Security Packages
5376.
5377.    - remove log in the error-handler
5378.    - [helmet](https://www.npmjs.com/package/helmet)
5379.      Helmet helps you secure your Express apps by setting various HTTP headers.
5380.    - [xss-clean](https://www.npmjs.com/package/xss-clean)
5381.      Node.js Connect middleware to sanitize user input coming from POST body, GET queries,
5382.      and url params.
5383.    - [express-mongo-sanitize](https://www.npmjs.com/package/express-mongo-sanitize)
5384.      Sanitizes user-supplied data to prevent MongoDB Operator Injection.
5385.    - [express-rate-limit](https://www.npmjs.com/package/express-rate-limit)
5386.      Basic rate-limiting middleware for Express.
5387.    ```sh
5388.    npm install helmet xss-clean express-mongo-sanitize express-rate-limit
5389.    ```
5390.
5391.    ```js
5392.    server.js;
5393.
5394.    import helmet from "helmet";
5395.    import xss from "xss-clean";
5396.    import mongoSanitize from "express-mongo-sanitize";
5397.
5398.    app.use(express.json());
5399.    app.use(helmet());
5400.    app.use(xss());
5401.    app.use(mongoSanitize());
5402.    ```
5403.
5404.    ##### Limit Requests
5405.
5406.    ```js
5407.    authRoutes.js;
5408.
5409.    import rateLimiter from "express-rate-limit";
5410.
5411.    const apiLimiter = rateLimiter({
5412.        windowMs: 15 * 60 * 1000, // 15 minutes
5413.        max: 10,
5414.        message: "Too many requests from this IP, please try again after 15 minutes",
5415.    });
5416.
5417.    router.route("/register").post(apiLimiter, register);
5418.    router.route("/login").post(apiLimiter, login);
5419.    ```
5420.

```

```

5421. ##### Alternative Search with Debounce
5422.
5423. client/components/SearchContainer.js
5424.
5425. ```js
5426.
5427. import { useState, useMemo } from 'react';
5428. const SearchContainer = () => {
5429.   const [localSearch, setLocalSearch] = useState('');
5430.   const {
5431.     ....
5432.   } = useAppContext();
5433.   const handleSearch = (e) => {
5434.     handleChange({ name: e.target.name, value: e.target.value });
5435.   };
5436.   const handleSubmit = (e) => {
5437.     e.preventDefault();
5438.     clearFilters();
5439.   };
5440.   const debounce = () => {
5441.     let timeoutID;
5442.     return (e) => {
5443.       setLocalSearch(e.target.value);
5444.       clearTimeout(timeoutID);
5445.       timeoutID = setTimeout(() => {
5446.         handleChange({ name: e.target.name, value: e.target.value });
5447.       }, 1000);
5448.     };
5449.   };
5450.
5451.   const handleSubmit = (e) => {
5452.     e.preventDefault();
5453.     setLocalSearch('');
5454.     clearFilters();
5455.   };
5456.
5457.   const optimizedDebounce = useMemo(() => debounce(), []);
5458.   return (
5459.     <Wrapper>
5460.       <form className='form'>
5461.         <h4>search form</h4>
5462.         <div className='form-center'>
5463.           { /* search position */ }
5464.
5465.           <FormRow
5466.             type='text'
5467.             name='search'
5468.             value={localSearch}
5469.             handleChange={optimizedDebounce}
5470.           />
5471.           .....
5472.         </div>
5473.       </form>
5474.     </Wrapper>
5475.   );
5476. };
5477.
5478. export default SearchContainer;
5479. ```
5480.
5481. ##### Test User - Initial Setup

```

```

5482.
5483.   - create new user (test user)
5484.   - populate DB with jobs
5485.   - create a login button
5486.
5487.   client/pages/Register.js
5488.
5489.   ```js
5490.   const Register = () => {
5491.     return (
5492.       <Wrapper className="full-page">
5493.         <form className="form" onSubmit={onSubmit}>
5494.           <button type="submit" className="btn btn-block" disabled={isLoading}>
5495.             submit
5496.           </button>
5497.           <button
5498.             type="button"
5499.             className="btn btn-block btn-hipster"
5500.             disabled={isLoading}
5501.             onClick={() => {
5502.               setupUser({
5503.                 currentUser: { email: "testUser@test.com", password: "secret" },
5504.                 endPoint: "login",
5505.                 alertText: "Login Successful! Redirecting...",
5506.               });
5507.             }}
5508.           >
5509.             {isLoading ? "loading..." : "demo app"}
5510.           </button>
5511.         </form>
5512.       </Wrapper>
5513.     );
5514.   };
5515.   export default Register;
5516.   ```
5517.
5518.   ##### Test User - Restrict Access (server)
5519.
5520.   - check for test user in auth middleware
5521.   - create new property on user object (true/false)
5522.   - create new middleware (testUser)
5523.   - check for test user, if true send back BadRequest Error
5524.   - add testUser middleware in front of routes you want to restrict access to
5525.
5526.   middleware/auth.js
5527.
5528.   ```js
5529.   import jwt from "jsonwebtoken";
5530.   import { UnauthenticatedError } from "../errors/index.js";
5531.
5532.   UnauthenticatedError;
5533.   const auth = async (req, res, next) => {
5534.     const authHeader = req.headers.authorization;
5535.     if (!authHeader || !authHeader.startsWith("Bearer ")) {
5536.       throw new UnauthenticatedError("Authentication Invalid");
5537.     }
5538.     const token = authHeader.split(" ")[1];
5539.     try {
5540.       const payload = jwt.verify(token, process.env.JWT_SECRET);
5541.       // TEST USER
5542.       const testUser = payload.userId === "testUserID";

```

```

5543.         req.user = { userId: payload.userId, testUser };
5544.         // TEST USER
5545.         next();
5546.     } catch (error) {
5547.         throw new UnauthenticatedError("Authentication Invalid");
5548.     }
5549. };
5550.
5551. export default auth;
5552. ```
5553.
5554. middleware/testUser
5555.
5556. ```js
5557. import { BadRequestError } from "../errors/index.js";
5558.
5559. const testUser = (req, res, next) => {
5560.     if (req.user.testUser) {
5561.         throw new BadRequestError("Test User. Read Only!");
5562.     }
5563.     next();
5564. };
5565.
5566. export default testUser;
5567. ```
5568.
5569. routes/jobsRoutes
5570.
5571. ```js
5572. import express from "express";
5573. const router = express.Router();
5574.
5575. import {
5576.     createJob,
5577.     deleteJob,
5578.     getAllJobs,
5579.     updateJob,
5580.     showStats,
5581. } from "../controllers/jobsController.js";
5582.
5583. import testUser from "../middleware/testUser.js";
5584.
5585. router.route("/").post(testUser, createJob).get(getAllJobs);
5586. // remember about :id
5587. router.route("/stats").get(showStats);
5588. router.route("/:id").delete(testUser, deleteJob).patch(testUser, updateJob);
5589.
5590. export default router;
5591. ```
5592.
5593. routes/authRoutes
5594.
5595. ```js
5596. import express from "express";
5597. const router = express.Router();
5598.
5599. import rateLimiter from "express-rate-limit";
5600. const apiLimiter = rateLimiter({
5601.     windowMs: 15 * 60 * 1000, // 15 minutes
5602.     max: 10,
5603.     message: "Too many requests from this IP, please try again after 15 minutes",

```

```

5604.     });
5605.
5606.     import { register, login, updateUser } from "../controllers/authController.js";
5607.     import authenticateUser from "../middleware/auth.js";
5608.     import testUser from "../middleware/testUser.js";
5609.     router.route("/register").post(apiLimiter, register);
5610.     router.route("/login").post(apiLimiter, login);
5611.     router.route("/updateUser").patch(authenticateUser, testUser, updateUser);
5612.
5613.     export default router;
5614.     ```
5615.
5616.     ##### Store JWT in Cookie
5617.
5618.     - BE PREPARED TO REFACTOR CODE !!!
5619.     - PLEASE DON'T RUSH THROUGH THESES VIDEOS
5620.     - CHECK FEW TIMES BEFORE REMOVING/ADDING CODE
5621.
5622.     ##### Attach Cookies to Login response
5623.
5624.     controllers/authController.js
5625.
5626.     ```js
5627.     // login controller
5628.
5629.     const token = user.createJWT();
5630.
5631.     const oneDay = 1000 * 60 * 60 * 24;
5632.
5633.     res.cookie("token", token, {
5634.         httpOnly: true,
5635.         expires: new Date(Date.now() + oneDay),
5636.         secure: process.env.NODE_ENV === "production",
5637.     });
5638.     ```
5639.
5640.     ##### Setup Function in Utils
5641.
5642.     - create attachCookies.js
5643.
5644.     ```js
5645.     const attachCookie = ({ res, token }) => {
5646.         const oneDay = 1000 * 60 * 60 * 24;
5647.
5648.         res.cookie("token", token, {
5649.             httpOnly: true,
5650.             expires: new Date(Date.now() + oneDay),
5651.             secure: process.env.NODE_ENV === "production",
5652.         });
5653.     };
5654.
5655.     export default attachCookie;
5656.     ```
5657.
5658.     - import in authController.js
5659.     - invoke in register/login/updateUser
5660.
5661.     ```js
5662.     import attachCookie from "../utils/attachCookie.js";
5663.
5664.     attachCookie({ res, token });

```

```
5665.    ```
5666.
5667.    ##### Parse Cookie Coming Back from the Front-End
5668.
5669.    - install cookie-parser (server)
5670.
5671.    ```sh
5672.    npm install cookie-parser
5673.    ```
5674.
5675.    server.js
5676.
5677.    ```js
5678.    import cookieParser from "cookie-parser";
5679.
5680.    app.use(express.json());
5681.    app.use(cookieParser());
5682.    ```
5683.
5684.    middleware/auth.js
5685.
5686.    ```js
5687.    const auth = async (req, res, next) => {
5688.        console.log(req.cookies)
5689.        ....
5690.    }
5691.    ```
5692.
5693.    ##### Refactor Auth Middleware
5694.
5695.    middleware/auth.js
5696.
5697.    ```js
5698.    const auth = async (req, res, next) => {
5699.        const token = req.cookies.token;
5700.        if (!token) {
5701.            throw new UnauthenticatedError("Authentication Invalid");
5702.        }
5703.        // rest of the code
5704.    };
5705.    ```
5706.
5707.    ##### SERVER - Remove Token from JSON Response
5708.
5709.    controllers/authController
5710.
5711.    register/login/updateUser
5712.
5713.    ```js
5714.    res.status(StatusCodes.OK).json({ user, location: user.location });
5715.    ```
5716.
5717.    - test the APP
5718.
5719.    ##### FRONT-END Remove Token from CONTEXT
5720.
5721.    - PLEASE BE CAREFUL WHEN MAKING THESE UPDATES
5722.        client/context/appContext
5723.
5724.    - remove
5725.
```

```

5726.    ``js
5727.    const token = localStorage.getItem("token");
5728.    const user = localStorage.getItem("user");
5729.    const userLocation = localStorage.getItem("location");
5730.    ``
5731.
5732.    - fix initial state
5733.
5734.    ``js
5735.    const initialState = {
5736.        // remove token all together
5737.        user: null,
5738.        userLocation: "",
5739.        jobLocation: "",
5740.    };
5741.    ``
5742.
5743.    - remove request interceptor
5744.
5745.    ``js
5746.    authFetch.interceptors.request.use(
5747.        (config) => {
5748.            config.headers.common["Authorization"] = `Bearer ${state.token}`;
5749.            return config;
5750.        },
5751.        (error) => {
5752.            return Promise.reject(error);
5753.        }
5754.    );
5755.    ``
5756.
5757.    - remove both addToLocalStorage and removeFromLocalStorage functions
5758.    - remove from setupUser and updateUser (token and local storage functions)
5759.    - remove from the reducer token (COMMAND + F)
5760.
5761.    ``js
5762.    const logoutUser = async () => {
5763.        dispatch({ type: LOGOUT_USER });
5764.        // remove local storage code
5765.    };
5766.    ``
5767.
5768.    #### Test Expiration
5769.
5770.    ``js
5771.    expires: new Date(Date.now() + 5000),
5772.    ``
5773.
5774.    #### GET Current User Route
5775.
5776.    controllers/authController.js
5777.
5778.    ``js
5779.    const getCurrentUser = async (req, res) => {
5780.        const user = await User.findOne({ _id: req.user.userId });
5781.        res.status(StatusCodes.OK).json({ user, location: user.location });
5782.    };
5783.
5784.    export { register, login, updateUser, getCurrentUser };
5785.    ``
5786.

```

```

5787. routes/authRoutes.js
5788.
5789. ```js
5790. import {
5791.     register,
5792.     login,
5793.     updateUser,
5794.     getCurrentUser,
5795. } from "../controllers/authController.js";
5796.
5797. router.route("/register").post(apiLimiter, register);
5798. router.route("/login").post(apiLimiter, login);
5799. router.route("/updateUser").patch(authenticateUser, testUser, updateUser);
5800. router.route("/getCurrentUser").get(authenticateUser, getCurrentUser);
5801. ```
5802.
5803. ##### GET Current User - Front-End
5804.
5805. actions.js
5806.
5807. ```js
5808. export const GET_CURRENT_USER_BEGIN = "GET_CURRENT_USER_BEGIN";
5809. export const GET_CURRENT_USER_SUCCESS = "GET_CURRENT_USER_SUCCESS";
5810. ```
5811.
5812. - setup imports (appContext and reducer)
5813.
5814. ##### GET Current User Request
5815.
5816. - first set the state value (default TRUE !!!)
5817. appContext.js
5818.
5819. ```js
5820. const initialState = {
5821.     userLoading: true,
5822. };
5823.
5824. const getCurrentUser = async () => {
5825.     dispatch({ type: GET_CURRENT_USER_BEGIN });
5826.     try {
5827.         const { data } = await authFetch("/auth/getCurrentUser");
5828.         const { user, location } = data;
5829.
5830.         dispatch({
5831.             type: GET_CURRENT_USER_SUCCESS,
5832.             payload: { user, location },
5833.         });
5834.     } catch (error) {
5835.         if (error.response.status === 401) return;
5836.         logoutUser();
5837.     }
5838. };
5839. useEffect(() => {
5840.     getCurrentUser();
5841. }, []);
5842. ```
5843.
5844. reducer.js
5845.
5846. ```js
5847. if (action.type === GET_CURRENT_USER_BEGIN) {

```



```

5848.     return { ...state, userLoading: true, showAlert: false };
5849.   }
5850.   if (action.type === GET_CURRENT_USER_SUCCESS) {
5851.     return {
5852.       ...state,
5853.       userLoading: false,
5854.       user: action.payload.user,
5855.       userLocation: action.payload.location,
5856.       jobLocation: action.payload.location,
5857.     };
5858.   }
5859.   ...
5860.
5861.   ```js
5862.   if (action.type === LOGOUT_USER) {
5863.     return {
5864.       ...initialState,
5865.       userLoading: false,
5866.     };
5867.   }
5868.   ...
5869.
5870.   ##### Protected Route FIX
5871.
5872.   ```js
5873.   import Loading from "../components/Loading";
5874.
5875.   const ProtectedRoute = ({ children }) => {
5876.     const { user, userLoading } = useAppContext();
5877.
5878.     if (userLoading) return <Loading />;
5879.
5880.     if (!user) {
5881.       return <Navigate to="/landing" />;
5882.     }
5883.     return children;
5884.   };
5885.
5886.   export default ProtectedRoute;
5887.   ...
5888.
5889.   ##### Landing Page
5890.
5891.   ```js
5892.   import { Navigate } from "react-router-dom";
5893.   import { useAppContext } from "../context/appContext";
5894.
5895.   const Landing = () => {
5896.     const { user } = useAppContext();
5897.     return (
5898.       <React.Fragment>
5899.         {user && <Navigate to="/" />}
5900.         <Wrapper>// rest of the code.....</Wrapper>
5901.       </React.Fragment>
5902.     );
5903.   };
5904.
5905.   export default Landing;
5906.   ...
5907.
5908.   ##### Logout Route

```

```

5909.
5910.     controllers/authController
5911.
5912.     ```js
5913.     const logout = async (req, res) => {
5914.         res.cookie("token", "logout", {
5915.             httpOnly: true,
5916.             expires: new Date(Date.now() + 1000),
5917.         });
5918.         res.status(StatusCodes.OK).json({ msg: "user logged out!" });
5919.     };
5920.     ```
5921.
5922.     routes/authRoutes
5923.
5924.     ```js
5925.     import {
5926.         register,
5927.         login,
5928.         updateUser,
5929.         getCurrentUser,
5930.         logout,
5931.     } from "../controllers/authController.js";
5932.
5933.     router.route("/register").post(apiLimiter, register);
5934.     router.route("/login").post(apiLimiter, login);
5935.     router.get("/logout", logout);
5936.     // rest of the code ....
5937.     ```
5938.
5939.     ##### Logout - Front-End
5940.
5941.     appContext.js
5942.
5943.     ```js
5944.     const logoutUser = async () => {
5945.         await authFetch.get("/auth/logout");
5946.         dispatch({ type: LOGOUT_USER });
5947.     };
5948.     ```
5949.
5950.     ##### Prepare for Deployment
5951.
5952.     - in client remove build and node_modules
5953.     - in server remove node_modules
5954.
5955.     package.json
5956.
5957.     ```json
5958.
5959.     "scripts":{
5960.         "setup-production":"npm run install-client && npm run build-client && npm install",
5961.         "install-client":"cd client && npm install",
5962.     }
5963.
5964.
5965.
5966.     ```
5967.
5968.     - node server
5969.     - APP NEEDS TO WORK LOCALLY !!!

```

```
5970.  
5971.     #### Github Repo  
5972.  
5973.     - create new repo  
5974.     - remove all existing repos (CHECK CLIENT !!!)  
5975.     - in the root  
5976.     - git init  
5977.     - git add .  
5978.     - git commit -m "first commit"  
5979.     - push up to Github
```