````
# Jobify

#### Track Your Job Search

Project in Action - [Jobify](https://www.jobify.live/)

#### Support

Find the App Useful? [You can always buy me a coffee](https://www.buymeacoffee.com/johnsmilga)

#### Run The App Locally

```sh
npm run install-dependencies
```

- rename .env.temp to .env
- setup values for - MONGO_URL, JWT_SECRET, JWT_LIFETIME

```sh
npm start
```

- visit url http://localhost:3000/

#### Setup React App

- create <b>client</b> folder
- open terminal

```sh
cd client
```

```sh
npx create-react-app .
```

```sh
npm start
```

- set editor/browser side by side
- copy/paste assets from complete project

#### Spring Cleaning

- in src remove
- App.css
- App.test.js
- logo.svg
- reportWebVitals.js
- setupTests.js
- fix App.js and index.js

#### Title and Favicon

- change title in public/index.html
- replace favicon.ico in public
- resource [Generate Favicons](https://favicon.io/)
````

````
#### Normalize.css and Global Styles

- CSS in JS (styled-components)
- saves times on the setup
- less lines of css
- speeds up the development
- normalize.css
- small CSS file that provides cross-browser consistency in the default styling of HTML elements.
- [normalize docs](https://necolas.github.io/normalize.css/)

```sh
npm install normalize.css
```

- import 'normalize.css' in index.js
- SET BEFORE 'index.css'
- replace contents of index.css
- if any questions about normalize or specific styles
- Coding Addict - [Default Starter Video](https://youtu.be/UDdyGNlQK5w)
- Repo - [Default Starter Repo](https://github.com/john-smilga/default-starter)

#### Landing Page

- zoom level 175%
- markdown preview extension
- get something on the screen
- react router and styled components right after
- create pages directory in the source
- for now Landing.js
- create component (snippets extension)
- setup basic return

```js
<h4>Landing Page<h4>
```

- import logo.svg and main.svg
- import Landing in App.js and render

      #### Styled Components

      - CSS in JS
      - Styled Components
      - have logic and styles in component
      - no name collisions
      - apply javascript logic
      - [Styled Components Docs](https://styled-components.com/)
      - [Styled Components Course](https://www.udemy.com/course/styled-components-tutorial-and-project-course/?referralCode=9DABB172FCB2625B663F)

      ```sh
      npm install styled-components
      ```

      ```js
      import styled from "styled-components";

      const El = styled.el`
        // styles go here
      `;
````
````

```
```

- no name collisions, since unique class
- vscode-styled-components extension
- colors and bugs
- style entire react component

```js
const Wrapper = styled.el``;

const Component = () => {
  return (
    <Wrapper>
      <h1> Component</h1>
    </Wrapper>
  );
};
```

- only responsible for styling
- wrappers folder in assets

#### Logo and Images

- logo built in Figma
- [Cool Images](https://undraw.co/)

#### Logo

- create components folder in source
- create Logo.js
- move import and image logic
- export as default
- utilize index.js

#### React Router

- Version 6
- [React Router Docs](https://reactrouter.com/docs/en/v6)

```sh
npm install history@5 react-router-dom@6
```

- import four components

```js
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
```

- Connect to browser's URL with BrowserRouter
- Routes instead of Switch

```js

<BrowserRouter>
    <Routes>
      <Route path="/" element={<div>Dashboard</div>} />
      <Route path="/register" element={<div>Register</div>} />
      <Route path="/landing" element={<Landing />} />
      <Route path="*" element={<div>Error</div>}>
```

```
          </Routes>
      </BrowserRouter>

      ```

      ```js
      <nav>
        <Link to="/">Dashboard</Link>
        <Link to="/register">Register</Link>
        <Link to="/landing">Home</Link>
      </nav>
      ```

      - go to Landing.js

      ```js
      import { Link } from "react-router-dom";

      return (
        <Link to="/register" className="btn btn-hero">
          Login / Register
        </Link>
      );
      ```

      #### Setup Pages

      - create Error, Register, Dashboard pages
      - basic return
      - create index.js
      - import all the pages
      - export one by one
      - basically the same, as in components
      - import App.js
      - add to element={}
      - remove temp navbar

      #### Error Page

      ```js
      import { Link } from "react-router-dom";
      import img from "../assets/images/not-found.svg";
      import Wrapper from "../assets/wrappers/ErrorPage";

      return (
        <Wrapper className="full-page">
          <div>
            <img src={img} alt="not found" />
            <h3>text</h3>
            <p>text</p>
            <Link to="/">back home</Link>
          </div>
        </Wrapper>
      );
      ```

      #### Auto Imports

      - use while developing
      - only sparingly while recording
      - better picture
```
```

````
- messes with flow
- just my preference
- still use them, just not all the time

#### Register Page - Setup

- show preview in Browser and themes

```js
import { useState, useEffect } from "react";
import { Logo } from "../components";
import Wrapper from "../assets/wrappers/RegisterPage";
// global context and useNavigate later

const initialState = {
  name: "",
  email: "",
  password: "",
  isMember: true,
};
// if possible prefer local state
// global state

function Register() {
  const [values, setValues] = useState(initialState);

  // global context and useNavigate later

  const handleChange = (e) => {
    console.log(e.target);
  };

  const onSubmit = (e) => {
    e.preventDefault();
    console.log(e.target);
  };
  return (
    <Wrapper className="full-page">
      <form className="form" onSubmit={onSubmit}>
        <Logo />
        <h3>Login</h3>

        {/* name field */}
        <div className="form-row">
          <label htmlFor="name" className="form-label">
            name
          </label>

          <input
            type="text"
            value={values.name}
            name="name"
            onChange={handleChange}
            className="form-input"
          />
        </div>

        <button type="submit" className="btn btn-block">
          submit
        </button>
      </form>
````
````

```
          </Wrapper>
        );
      }
      ```

      #### FormRow Component

      - create FormRow.js in <b>components</b>
      - setup import/export
      - setup one for email and password
      - hint "type,name,value"

      ```js
      const FormRow = ({ type, name, value, handleChange, labelText }) => {
        return (
          <div className="form-row">
            <label htmlFor={name} className="form-label">
              {labelText || name}
            </label>

            <input
              type={type}
              value={value}
              name={name}
              onChange={handleChange}
              className="form-input"
            />
          </div>
        );
      };

      export default FormRow;
      ```

      #### Alert Component

      - right away setup as component
      - create Alert.js in <b>components</b>

      ```js
      const Alert = () => {
        return <div className="alert alert-danger">alert goes here</div>;
      };

      export default Alert;
      ```

      - setup import/export
      - alert-danger or alert-success
      - eventually setup in global context
      - showAlert in initialState (true || false)
      - right after h3 login

      ```js
      values.showAlert && <Alert />;
      ```

      #### Toggle Member

      ```js
      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}
```

```
          </AppContext.Provider>
        );
      };
      // make sure use
      export const useAppContext = () => {
        return useContext(AppContext);
      };

      export { AppProvider };
      ```

      - index.js

      ```js
      import { AppProvider } from "./context/appContext";

      ReactDOM.render(
        <React.StrictMode>
          <AppProvider>
            <App />
          </AppProvider>
        </React.StrictMode>,
        document.getElementById("root")
      );
      ```

      - Register.js

      ```js
      import { useAppContext } from "../context/appContext";

      const { isLoading, showAlert } = useAppContext();
      ```

      - switch to global showAlert

      #### useReducer

      - [React Tutorial](https://youtu.be/iZhV0bILFb0)
      - useReducer vs Redux
      - multiple reducers vs one

      #### Wire Up Reducer

      ```js
      reducer.js;

      const reducer = (state, action) => {
        throw new Error(`no such action :${action.type}`);
      };
      export default reducer;
      ```

      ```js
      appContext.js;

      import reducer from "./reducer";

      const [state, dispatch] = useReducer(reducer, initialState);
      ```
```
```

````
#### Display Alert

```js
actions.js;

export const DISPLAY_ALERT = "SHOW_ALERT";
```

- setup imports (reducer and appContext)

```js
appContext.js

const displayAlert() =>{
  dispatch({type:DISPLAY_ALERT})
}

```

```js
reducer.js;

if (action.type === DISPLAY_ALERT) {
  return {
    ...state,
    showAlert: true,
    alertType: "danger",
    alertText: "Please provide all values!",
  };
}
```

```js
Alert.js in Components;

import { useAppContext } from "../context/appContext";

const Alert = () => {
  const { alertType, alertText } = useAppContext();
  return <div className={`alert alert-${alertType}`}>{alertText}</div>;
};
```

#### Display Alert

- [JS Nuggets - Dynamic Object Keys](https://youtu.be/_qxCYtWm0tw)

```js
appContext.js;

const handleChange = (e) => {
  setValues({ ...values, [e.target.name]: e.target.value });
};
```

- get displayAlert function

```js
appContext.js;

const onSubmit = (e) => {
````
````

```
    e.preventDefault();
    const { name, email, password, isMember } = values;
    if (!email || !password || (!isMember && !name)) {
      displayAlert();
      return;
    }
    console.log(values);
  };
  ```

  #### Clear Alert

  - technically optional

  ```js
  actions.js;

  export const CLEAR_ALERT = "CLEAR_ALERT";
  ```

  - setup imports (reducer and appContext)

  ```js
  reducer.js;

  if (action.type === CLEAR_ALERT) {
    return {
      ...state,
      showAlert: false,
      alertType: "",
      alertText: "",
    };
  }
  ```

  ```js
  appContext.js;

  const displayAlert = () => {
    dispatch({
      type: DISPLAY_ALERT,
    });
    clearAlert();
  };

  const clearAlert = () => {
    setTimeout(() => {
      dispatch({
        type: CLEAR_ALERT,
      });
    }, 3000);
  };
  ```

  #### Setup Server

  - stop the dev server in client
  - cd ..
  - start setting up our server
  - setup package.json
```
```

````
```sh
npm init -y
```

- create server.js
- console.log('server running...')

```sh
node server
```

#### ES6 vs CommonJS

```js
CommonJS;

const express = require("express");
const app = express();
```

```js
ES6;

import express from "express";
const app = express();
```

- file extension .mjs

```js
package.json

"type":"module"
```

#### Nodemon and Basic Express Server

```sh
npm install nodemon --save-dev
```

```js
package.json

"start":"nodemon server"

```

```sh
npm install express
```

```js
import express from "express";
const app = express();

app.get("/", (req, res) => {
  res.send("Welcome!");
});

const port = process.env.PORT || 5000;
````
````

````
app.listen(port, () => console.log(`Server is listening on port ${port}...`));
```

# Jobify

#### Track Your Job Search

Project in Action - [Jobify](https://www.jobify.live/)

#### Support

Find the App Useful? [You can always buy me a coffee](https://www.buymeacoffee.com/johnsmilga)

#### Run The App Locally

```sh
npm run install-dependencies
```

- rename .env.temp to .env
- setup values for - MONGO_URL, JWT_SECRET, JWT_LIFETIME

```sh
npm start
```

- visit url http://localhost:3000/

#### Setup React App

- create <b>client</b> folder
- open terminal

```sh
cd client
```

```sh
npx create-react-app .
```

```sh
npm start
```

- set editor/browser side by side
- copy/paste assets from complete project

#### Spring Cleaning

- in src remove
- App.css
- App.test.js
- logo.svg
- reportWebVitals.js
- setupTests.js
- fix App.js and index.js

#### 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/UDdyGNlQK5w)
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
```

````
      import styled from "styled-components";

      const El = styled.el`
        // styles go here
      `;
      ```

      - no name collisions, since unique class
      - vscode-styled-components extension
      - colors and bugs
      - style entire react component

      ```js
      const Wrapper = styled.el``;

      const Component = () => {
        return (
          <Wrapper>
            <h1> Component</h1>
          </Wrapper>
        );
      };
      ```

      - only responsible for styling
      - wrappers folder in assets

      #### Logo and Images

      - logo built in Figma
      - [Cool Images](https://undraw.co/)

      #### Logo

      - create <b>components</b> folder in source
      - create Logo.js
      - move import and image logic
      - export as default
      - utilize index.js

      #### React Router

      - Version 6
      - [React Router Docs](https://reactrouter.com/docs/en/v6)

      ```sh
      npm install history@5 react-router-dom@6
      ```

      - import four components

      ```js
      import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
      ```

      - Connect to browser's URL with BrowserRouter
      - Routes instead of Switch

      ```js

      <BrowserRouter>
````

```
          <Routes>
            <Route path="/" element={<div>Dashboard</div>} />
            <Route path="/register" element={<div>Register</div>} />
            <Route path="/landing" element={<Landing />} />
            <Route path="*" element={<div>Error</div>}>
          </Routes>
      </BrowserRouter>

      ```

      ```js
      <nav>
        <Link to="/">Dashboard</Link>
        <Link to="/register">Register</Link>
        <Link to="/landing">Home</Link>
      </nav>
      ```

      - go to Landing.js

      ```js
      import { Link } from "react-router-dom";

      return (
        <Link to="/register" className="btn btn-hero">
          Login / Register
        </Link>
      );
      ```

      #### Setup Pages

      - create Error, Register, Dashboard pages
      - basic return
      - create index.js
      - import all the pages
      - export one by one
      - basically the same, as in components
      - import App.js
      - add to element={}
      - remove temp navbar

      #### Error Page

      ```js
      import { Link } from "react-router-dom";
      import img from "../assets/images/not-found.svg";
      import Wrapper from "../assets/wrappers/ErrorPage";

      return (
        <Wrapper className="full-page">
          <div>
            <img src={img} alt="not found" />
            <h3>text</h3>
            <p>text</p>
            <Link to="/">back home</Link>
          </div>
        </Wrapper>
      );
      ```
```
```

````
#### Auto Imports

- use while developing
- only sparingly while recording
- better picture
- messes with flow
- just my preference
- still use them, just not all the time

#### Register Page - Setup

- show preview in Browser and themes

```js
import { useState, useEffect } from "react";
import { Logo } from "../components";
import Wrapper from "../assets/wrappers/RegisterPage";
// global context and useNavigate later

const initialState = {
  name: "",
  email: "",
  password: "",
  isMember: true,
};
// if possible prefer local state
// global state

function Register() {
  const [values, setValues] = useState(initialState);

  // global context and useNavigate later

  const handleChange = (e) => {
    console.log(e.target);
  };

  const onSubmit = (e) => {
    e.preventDefault();
    console.log(e.target);
  };
  return (
    <Wrapper className="full-page">
      <form className="form" onSubmit={onSubmit}>
        <Logo />
        <h3>Login</h3>

        {/* name field */}
        <div className="form-row">
          <label htmlFor="name" className="form-label">
            name
          </label>

          <input
            type="text"
            value={values.name}
            name="name"
            onChange={handleChange}
            className="form-input"
          />
        </div>
````
````

````
                <button type="submit" className="btn btn-block">
                  submit
                </button>
              </form>
            </Wrapper>
          );
        }
        ```

        #### FormRow Component

        - create FormRow.js in <b>components</b>
        - setup import/export
        - setup one for email and password
        - hint "type,name,value"

        ```js
        const FormRow = ({ type, name, value, handleChange, labelText }) => {
          return (
            <div className="form-row">
              <label htmlFor={name} className="form-label">
                {labelText || name}
              </label>

              <input
                type={type}
                value={value}
                name={name}
                onChange={handleChange}
                className="form-input"
              />
            </div>
          );
        };

        export default FormRow;
        ```

        #### Alert Component

        - right away setup as component
        - create Alert.js in <b>components</b>

        ```js
        const Alert = () => {
          return <div className="alert alert-danger">alert goes here</div>;
        };

        export default Alert;
        ```

        - setup import/export
        - alert-danger or alert-success
        - eventually setup in global context
        - showAlert in initialState (true || false)
        - right after h3 login

        ```js
        values.showAlert && <Alert />;
        ```
````
````

```

#### Toggle Member

```js
const toggleMember = () => {
  setValues({ ...values, isMember: !values.isMember });
};

return (
  <Wrapper>
    {/* control h3 */}

    <h3>{values.isMember ? "Login" : "Register"}</h3>

    {/* toggle name */}

    {!values.isMember && (
      <FormRow
        type="text"
        name="name"
        value={values.name}
        handleChange={handleChange}
      />
    )}

    {/* right after submit btn */}
    {/* toggle button */}

    <p>
      {values.isMember ? "Not a member yet?" : "Already a member?"}

      <button type="button" onClick={toggleMember} className="member-btn">
        {values.isMember ? "Register" : "Login"}
      </button>
    </p>
  </Wrapper>
);
```

#### Global Context

- in src create <b>context</b> directory
- actions.js
- reducer.js
- appContext.js

```js
import React, { useState, useReducer, useContext } from "react";

export const initialState = {
  isLoading: false,
  showAlert: false,
  alertText: "",
  alertType: "",
};
const AppContext = React.createContext();
const AppProvider = ({ children }) => {
  const [state, setState] = useState(initialState);

  return (
    <AppContext.Provider
```
```

```
            value={{
              ...state,
            }}
          >
            {children}
          </AppContext.Provider>
        );
      };
      // make sure use
      export const useAppContext = () => {
        return useContext(AppContext);
      };

      export { AppProvider };
      ```

      - index.js

      ```js
      import { AppProvider } from "./context/appContext";

      ReactDOM.render(
        <React.StrictMode>
          <AppProvider>
            <App />
          </AppProvider>
        </React.StrictMode>,
        document.getElementById("root")
      );
      ```

      - Register.js

      ```js
      import { useAppContext } from "../context/appContext";

      const { isLoading, showAlert } = useAppContext();
      ```

      - switch to global showAlert

      #### useReducer

      - [React Tutorial](https://youtu.be/iZhV0bILFb0)
      - useReducer vs Redux
      - multiple reducers vs one

      #### Wire Up Reducer

      ```js
      reducer.js;

      const reducer = (state, action) => {
        throw new Error(`no such action :${action.type}`);
      };
      export default reducer;
      ```

      ```js
      appContext.js;

```

````
    import reducer from "./reducer";

    const [state, dispatch] = useReducer(reducer, initialState);
    ```

    #### Display Alert

    ```js
    actions.js;

    export const DISPLAY_ALERT = "SHOW_ALERT";
    ```

    - setup imports (reducer and appContext)

    ```js
    appContext.js

    const displayAlert() =>{
      dispatch({type:DISPLAY_ALERT})
    }

    ```

    ```js
    reducer.js;

    if (action.type === DISPLAY_ALERT) {
      return {
        ...state,
        showAlert: true,
        alertType: "danger",
        alertText: "Please provide all values!",
      };
    }
    ```

    ```js
    Alert.js in Components;

    import { useAppContext } from "../context/appContext";

    const Alert = () => {
      const { alertType, alertText } = useAppContext();
      return <div className={`alert alert-${alertType}`}>{alertText}</div>;
    };
    ```

    #### Display Alert

    - [JS Nuggets - Dynamic Object Keys](https://youtu.be/_qxCYtWm0tw)

    ```js
    appContext.js;

    const handleChange = (e) => {
      setValues({ ...values, [e.target.name]: e.target.value });
    };
    ```

    - get displayAlert function
````
````

```
```js
appContext.js;

const onSubmit = (e) => {
  e.preventDefault();
  const { name, email, password, isMember } = values;
  if (!email || !password || (!isMember && !name)) {
    displayAlert();
    return;
  }
  console.log(values);
};
```

#### Clear Alert

- technically optional

```js
actions.js;

export const CLEAR_ALERT = "CLEAR_ALERT";
```

- setup imports (reducer and appContext)

```js
reducer.js;

if (action.type === CLEAR_ALERT) {
  return {
    ...state,
    showAlert: false,
    alertType: "",
    alertText: "",
  };
}
```

```js
appContext.js;

const displayAlert = () => {
  dispatch({
    type: DISPLAY_ALERT,
  });
  clearAlert();
};

const clearAlert = () => {
  setTimeout(() => {
    dispatch({
      type: CLEAR_ALERT,
    });
  }, 3000);
};
```

#### Setup Server
```

````
    - stop the dev server in client
    - cd ..
    - start setting up our server
    - setup package.json

    ```sh
    npm init -y
    ```

    - create server.js
    - console.log('server running...')

    ```sh
    node server
    ```

    #### ES6 vs CommonJS

    ```js
    CommonJS;

    const express = require("express");
    const app = express();
    ```

    ```js
    ES6;

    import express from "express";
    const app = express();
    ```

    - file extension .mjs

    ```js
    package.json

    "type":"module"
    ```

    #### Nodemon and Basic Express Server

    ```sh
    npm install nodemon --save-dev
    ```

    ```js
    package.json

    "start":"nodemon server"

    ```

    ```sh
    npm install express
    ```

    ```js
    import express from "express";
    const app = express();

````
````

````
    app.get("/", (req, res) => {
      res.send("Welcome!");
    });

    const port = process.env.PORT || 5000;

    app.listen(port, () => console.log(`Server is listening on port ${port}...`));
    ```

    #### Not Found Middleware

    - in the root create <b>middleware</b> folder
    - not-found.js
    - setup function
    - return 404 with message 'Route does not exist'
    - import in server.js
    - make sure to use .js extension
    - place after home route

    #### Error Middleware

    - in the middleware create error-handler.js
    - setup function
    - accept 4 parameters, first one error
    - log error
    - return 500
    - json({msg:'there was an error'})
    - import in the server.js
    - make sure to use .js extension
    - place it last
    - eventually handle Mongoose Errors, just like in the node-express
    - showcase with async errors

    #### ENV Variables

    ```sh
    npm install dotenv
    ```

    - import dotenv from 'dotenv'
    - dotenv.config()

    - create .env
    - PORT=4000
    - .gitignore
    - /node_modules
    - .env

    #### Connect to MongoDB

    - switched back to PORT=5000
    - remove Error from '/'

    - existing MongoDB Atlas Account

    ```sh
    npm install mongoose
    ```

    - create <b>db</b> folder
    - create connect.js
````
````

````
    - setup connectDB(url)
    - in server.js create start() function
    - get connection string
    - setup as MONGO_URL in .env
    - provide credentials and DB Name

    #### Auth Controller and Route Structure

    - create <b>controllers</b>
    - authController.js
    - create async functions

    ```js
    export { register, login, updateUser };
    ```

    - return res.send('function name')
    - create <b>routes</b> folder
    - authRoutes.js
    - setup express router
    - import functions from authController.js

    ```js
    router.route("/register").post(register);
    router.route("/login").post(login);
    router.route("/updateUser").patch(updateUser);

    export default router;
    ```

    - import authRouter in server.js

    ```js
    app.use("/api/v1/auth", authRouter);
    ```

    #### Jobs Controller and Route Structure

    - jobsController.js
    - create async functions

    ```js
    export { createJob, deleteJob, getAllJobs, updateJob, showStats };
    ```

    - return res.send('function name')

    - jobsRoutes.js
    - setup express router
    - import functions from jobsController.js

    ```js
    router.route("/").post(createJob).get(getAllJobs);
    // place before :id
    router.route("/stats").get(showStats);
    router.route("/:id").delete(deleteJob).patch(updateJob);

    export default router;
    ```

    - in server.js jobsRouter
````

````
```js
app.use("/api/v1/jobs", jobsRouter);
```

#### Postman

- URL global var
- JOBIFY Collection
- auth and jobs folders
- setup routes

#### User Model

- <b>models</b> folder
- User.js
- setup schema
- name, email, password, lastName, location
- all {type:String}

#### Validate Email

```js
validate:{
  validator:(field)=> {return 2 > 1},
  message:'Please provide valid email'
  }
```

- [Validator Package](https://www.npmjs.com/package/validator)

```sh
npm install validator
```

- import in User.js
- validator.isEmail

#### Register User - Initial Setup

- authController
- import User model
- setup temporary try/catch
- await User.create(req.body)
- if success 201 with json({user}) (temp)
- if error 500 with json({msg:'there was an error'})

#### Pass Error to Error Handler

- next(error)

#### Express-Async-Errors Package

- remove try/catch
- [Express-Async-Errors](https://www.npmjs.com/package/express-async-errors)

```sh
npm install express-async-errors

```
````

```
- in server.js
- import 'express-async-errors'

- use throw Error('error') instead of next(error)

#### Http Status Codes

- constants for status codes
- personal preference
- provides consistency
- less bugs
- easier to read/manage

- [Http Status Codes](https://www.npmjs.com/package/http-status-codes)

```sh
npm install http-status-codes
```

- import/setup in authController and error-handler
- setup defaultError

#### Custom Errors

#### Refactor Errors

- create errors folder
- create custom-api, bad-request, not-found, index.js files
- add proper imports
- setup index.js just like in the front-end
- import {BadRequestError} in authController
- gotcha "errors/index.js"

#### Hash Passwords

- one way street, only compare hashed values
- [bcrypt.js](https://www.npmjs.com/package/bcryptjs)

```sh
npm install bcryptjs
```

- User Model
- import bcrypt from 'bcryptjs'
- await genSalt(10)
- await hash(password , salt)
- await compare(requestPassword , currentPassword)
- [mongoose middleware](https://mongoosejs.com/docs/middleware.html)
- UserSchema.pre('save',async function(){
  "this" points to instance created by UserSchema
  })

#### Mongoose - Custom Instance Methods

[Custom Instance Methods](https://mongoosejs.com/docs/guide.html#methods)

- UserSchema.methods.createJWT = function(){console.log(this)}
- register controller
- right after User.create()
- invoke user.createJWT()
```

````
#### JWT

- token
- [jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken)

```sh
npm install jsonwebtoken
```

- User Model
- import jwt from 'jsonwebtoken'
- jwt.sign(payload,secret,options)
- createJWT

```js
return jwt.sign({ userId: this._id }, "jwtSecret", { expiresIn: "1d" });
```

```js
return jwt.sign({ userId: this._id }, process.env.JWT_SECRET, {
  expiresIn: process.env.JWT_LIFETIME,
});
```

#### JWT_SECRET and JWT_LIFETIME

- [Keys Generator](https://www.allkeysgenerator.com/)
- RESTART SERVER!!!!

#### Complete Register

- password : {select:false}
- complete response

#### Concurrently

- front-end and backend (server)
- run separate terminals
- [concurrently](https://www.npmjs.com/package/concurrently)

```sh
npm install concurrently --save-dev

```

- package.json

```js
// --kill-others switch, all commands are killed if one dies
// --prefix client - folder
// cd client && npm start
// escape quotes

"scripts": {
    "server": "nodemon server --ignore client",
    "client": "npm start --prefix client",
    "start": "concurrently --kill-others-on-fail \"npm run server\" \" npm run client\""
  },
```
````

````
#### Cors Error

[Cors Error](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS)

- two fixes (cors package and proxy)

#### Cors Package

[cors package](https://www.npmjs.com/package/cors)

```sh
npm install cors
```

```js
import cors from "cors";

app.use(cors());
```

#### Proxy

- access from anywhere
- don't want to use full url

[cra proxy](https://create-react-app.dev/docs/proxying-api-requests-in-development/)

```js
"proxy":"http://localhost:5000"
```

- my preference to remove trailing slash /
- restart app

#### Register User - Setup

```js
appContext.js;

const initialState = {
  user: null,
  token: null,
  userLocation: "",
};
```

- actions.js REGISTER_USER_BEGIN,SUCCESS,ERROR
- import reducer,appContext

```js
appContext.js;
const registerUser = async (currentUser) => {
  console.log(currentUser);
};
```

- import in Register.js

```js
Register.js;
````
````

````
    const currentUser = { name, email, password };
    if (isMember) {
      console.log("already a member");
    } else {
      registerUser(currentUser);
    }

    return (
      <button type="submit" className="btn btn-block" disabled={isLoading}>
        submit
      </button>
    );
    ```

    #### Axios

    - [axios docs](https://axios-http.com/docs/intro)
    - stop app
    - cd client

    ```sh
    npm install axios
    ```

    - cd ..
    - restart app

    #### Register User - Complete

    ```js
    appContext.js;

    import axios from "axios";

    const registerUser = async (currentUser) => {
      dispatch({ type: REGISTER_USER_BEGIN });
      try {
        const response = await axios.post("/api/v1/auth/register", currentUser);
        console.log(response);
        const { user, token, location } = response.data;
        dispatch({
          type: REGISTER_USER_SUCCESS,
          payload: {
            user,
            token,
            location,
          },
        });

        // will add later
        // addUserToLocalStorage({
        //   user,
        //   token,
        //   location,
        // })
      } catch (error) {
        console.log(error.response);
        dispatch({
          type: REGISTER_USER_ERROR,
          payload: { msg: error.response.data.msg },
        });
````

```
    }
    clearAlert();
  };
  ```

  ```js
  reducer.js;
  if (action.type === REGISTER_USER_BEGIN) {
    return { ...state, isLoading: true };
  }
  if (action.type === REGISTER_USER_SUCCESS) {
    return {
      ...state,
      user: action.payload.user,
      token: action.payload.token,
      userLocation: action.payload.location,
      jobLocation: action.payload.location,
      isLoading: false,
      showAlert: true,
      alertType: "success",
      alertText: "User Created! Redirecting...",
    };
  }
  if (action.type === REGISTER_USER_ERROR) {
    return {
      ...state,
      isLoading: false,
      showAlert: true,
      alertType: "danger",
      alertText: action.payload.msg,
    };
  }
  ```

  #### Navigate To Dashboard

  ```js
  Register.js;
  import { useEffect } from "react";
  import { useNavigate } from "react-router-dom";

  const Register = () => {
    const { user } = useAppContext();
    const navigate = useNavigate();

    useEffect(() => {
      if (user) {
        setTimeout(() => {
          navigate("/");
        }, 3000);
      }
    }, [user, navigate]);
  };
  ```

  #### Local Storage

  ```js
  appContext.js;
  const addUserToLocalStorage = ({ user, token, location }) => {
    localStorage.setItem("user", JSON.stringify(user));
```

````
    localStorage.setItem("token", token);
    localStorage.setItem("location", location);
  };

  const removeUserFromLocalStorage = () => {
    localStorage.removeItem("token");
    localStorage.removeItem("user");
    localStorage.removeItem("location");
  };

  const registerUser = async (currentUser) => {
    // in try block
    addUserToLocalStorage({
      user,
      token,
      location,
    });
  };

  // set as default
  const token = localStorage.getItem("token");
  const user = localStorage.getItem("user");
  const userLocation = localStorage.getItem("location");

  const initialState = {
    user: user ? JSON.parse(user) : null,
    token: token,
    userLocation: userLocation || "",
    jobLocation: userLocation || "",
  };
  ```

  #### Morgan Package

  - http logger middleware for node.js
  - [morgan docs](https://www.npmjs.com/package/morgan)

  ```sh
  npm install morgan
  ```

  ```js
  import morgan from "morgan";

  if (process.env.NODE_ENV !== "production") {
    app.use(morgan("dev"));
  }
  ```

  #### UnauthenticatedError

  - unauthenticated.js in errors
  - import/export

  ```js
  import { StatusCodes } from "http-status-codes";
  import CustomAPIError from "./custom-api.js";

  class UnauthenticatedError extends CustomAPIError {
    constructor(message) {
      super(message);
````

```
        this.statusCode = StatusCodes.UNAUTHORIZED;
      }
    }
    ```

    #### Compare Password

    ```js
    User.js in models;

    UserSchema.methods.comparePassword = async function (candidatePassword) {
      const isMatch = await bcrypt.compare(candidatePassword, this.password);
      return isMatch;
    };
    ```

    ```js
    authController.js;
    const login = async (req, res) => {
      const { email, password } = req.body;
      if (!email || !password) {
        throw new BadRequestError("Please provide all values");
      }
      const user = await User.findOne({ email }).select("+password");

      if (!user) {
        throw new UnauthenticatedError("Invalid Credentials");
      }
      const isPasswordCorrect = await user.comparePassword(password);
      if (!isPasswordCorrect) {
        throw new UnauthenticatedError("Invalid Credentials");
      }
      const token = user.createJWT();
      user.password = undefined;
      res.status(StatusCodes.OK).json({ user, token, location: user.location });
    };
    ```

    - test in Postman

    #### Login User - Setup

    - actions.js LOGIN_USER_BEGIN,SUCCESS,ERROR
    - import reducer,appContext

    ```js
    appContext.js;
    const loginUser = async (currentUser) => {
      console.log(currentUser);
    };
    ```

    - import in Register.js

    ```js
    Register.js;

    if (isMember) {
      loginUser(currentUser);
    } else {
      registerUser(currentUser);
```

````
    }
```

#### Login User - Complete

```js
appContext.js;
const loginUser = async (currentUser) => {
  dispatch({ type: LOGIN_USER_BEGIN });
  try {
    const { data } = await axios.post("/api/v1/auth/login", currentUser);
    const { user, token, location } = data;

    dispatch({
      type: LOGIN_USER_SUCCESS,
      payload: { user, token, location },
    });

    addUserToLocalStorage({ user, token, location });
  } catch (error) {
    dispatch({
      type: LOGIN_USER_ERROR,
      payload: { msg: error.response.data.msg },
    });
  }
  clearAlert();
};
```

```js
reducer.js;

if (action.type === LOGIN_USER_BEGIN) {
  return {
    ...state,
    isLoading: true,
  };
}
if (action.type === LOGIN_USER_SUCCESS) {
  return {
    ...state,
    isLoading: false,
    user: action.payload.user,
    token: action.payload.token,
    userLocation: action.payload.location,
    jobLocation: action.payload.location,
    showAlert: true,
    alertType: "success",
    alertText: "Login Successful! Redirecting...",
  };
}
if (action.type === LOGIN_USER_ERROR) {
  return {
    ...state,
    isLoading: false,
    showAlert: true,
    alertType: "danger",
    alertText: action.payload.msg,
  };
}
```
````
````

````
#### Refactor

```js
actions.js;
export const SETUP_USER_BEGIN = "SETUP_USER_BEGIN";
export const SETUP_USER_SUCCESS = "SETUP_USER_SUCCESS";
export const SETUP_USER_ERROR = "SETUP_USER_ERROR";
```

```js
appContext.js;

const setupUser = async ({ currentUser, endPoint, alertText }) => {
  dispatch({ type: SETUP_USER_BEGIN });
  try {
    const { data } = await axios.post(`/api/v1/auth/${endPoint}`, currentUser);

    const { user, token, location } = data;
    dispatch({
      type: SETUP_USER_SUCCESS,
      payload: { user, token, location, alertText },
    });
    addUserToLocalStorage({ user, token, location });
  } catch (error) {
    dispatch({
      type: SETUP_USER_ERROR,
      payload: { msg: error.response.data.msg },
    });
  }
  clearAlert();
};
```

```js
reducer.js;
if (action.type === SETUP_USER_BEGIN) {
  return { ...state, isLoading: true };
}
if (action.type === SETUP_USER_SUCCESS) {
  return {
    ...state,
    isLoading: false,
    token: action.payload.token,
    user: action.payload.user,
    userLocation: action.payload.location,
    jobLocation: action.payload.location,
    showAlert: true,
    alertType: "success",
    alertText: action.payload.alertText,
  };
}
if (action.type === SETUP_USER_ERROR) {
  return {
    ...state,
    isLoading: false,
    showAlert: true,
    alertType: "danger",
    alertText: action.payload.msg,
  };
}
````
````

````
```

- import/export

```js
Register.js;

const onSubmit = (e) => {
  e.preventDefault();
  const { name, email, password, isMember } = values;
  if (!email || !password || (!isMember && !name)) {
    displayAlert();
    return;
  }
  const currentUser = { name, email, password };
  if (isMember) {
    setupUser({
      currentUser,
      endPoint: "login",
      alertText: "Login Successful! Redirecting...",
    });
  } else {
    setupUser({
      currentUser,
      endPoint: "register",
      alertText: "User Created! Redirecting...",
    });
  }
};
```

#### Nested Pages in React Router 6

#### Dashboard pages

- delete Dashboard.js
- fix imports/exports
- replace in home route

```js
<Route path="/" element={<div>dashboard</div>} />
```

- create <b>dashboard</b> directory in pages
- create AddJob,AllJobs,Profile,Stats,SharedLayout, index.js
- setup basic returns

```js
return <h1>Add Job Page</h1>;
```

- export all with index.js (just like components)
- import all pages in App.js

#### Nested Structure

```js
App.js

<Route path='/' >
  <Route path="stats" element={<Stats />} />
```
````
````

```
        <Route path='all-jobs' element={<AllJobs />}></Route>
        <Route path='add-job' element={<AddJob />}></Route>
        <Route path='profile' element={<Profile />}></Route>
      </Route>
      ```

      #### Shared Layout

      ```js
      App.js

      <Route path='/' element={<SharedLayout/>} >
      ```

      ```js
      SharedLayout.js;

      import { Outlet, Link } from "react-router-dom";
      import Wrapper from "../../assets/wrappers/SharedLayout";

      const SharedLayout = () => {
        return (
          <Wrapper>
            <nav>
              <Link to="all-jobs">all jobs</Link>
              <Link to="add-job">all jobs</Link>
            </nav>
            <Outlet />
          </Wrapper>
        );
      };

      export default SharedLayout;
      ```

      ```js
      App.js

      <Route index element={<Stats/>} >
      ```

      #### Protected Route

      - create ProtectedRoute.js in pages
      - import/export
      - wrap SharedLayout in App.js

      ```js
      <Route
        path="/"
        element={
          <ProtectedRoute>
            <SharedLayout />
          </ProtectedRoute>
        }
      />
      ```

      ```js
      ProtectedRoute.js;
```

````
    import { Navigate } from "react-router-dom";
    import { useAppContext } from "../context/appContext";

    const ProtectedRoute = ({ children }) => {
      const { user } = useAppContext();
      if (!user) {
        return <Navigate to="/landing" />;
      }
      return children;
    };
    ```

    #### Navbar, SmallSidebar, BigSidebar

    - create Navbar, SmallSidebar, BigSidebar in components
    - import Wrappers from assets/wrappers
    - simple return
    - import/export

    ```js
    SharedLayout.js;

    import { Outlet } from "react-router-dom";
    import { Navbar, SmallSidebar, BigSidebar } from "../../components";
    import Wrapper from "../../assets/wrappers/SharedLayout";

    const SharedLayout = () => {
      const { user } = useAppContext();
      return (
        <>
          <Wrapper>
            <main className="dashboard">
              <SmallSidebar />
              <BigSidebar />
              <div>
                <Navbar />
                <div className="dashboard-page">
                  <Outlet />
                </div>
              </div>
            </main>
          </Wrapper>
        </>
      );
    };

    export default SharedLayout;
    ```

    #### React Icons

    [React Icons](https://react-icons.github.io/react-icons/)

    ```sh
    npm install react-icons
    ```

    ```js
    Navbar.js

    import Wrapper from '../assets/wrappers/Navbar'
````

````
    import {FaHome} from 'react-icons/fa'
    const Navbar = () => {
      return (
        <Wrapper>
          <h4>navbar</h4>
          <FaHome>
        </Wrapper>
      )
    }

    export default Navbar

    ```

    #### Navbar Setup

    ```js
    Navbar.js;

    import { useState } from "react";
    import { FaAlignLeft, FaUserCircle, FaCaretDown } from "react-icons/fa";
    import { useAppContext } from "../context/appContext";
    import Logo from "./Logo";
    import Wrapper from "../assets/wrappers/Navbar";
    const Navbar = () => {
      return (
        <Wrapper>
          <div className="nav-center">
            <button
              className="toggle-btn"
              onClick={() => console.log("toggle sidebar")}
            >
              <FaAlignLeft />
            </button>

            <div>
              <Logo />
              <h3 className="logo-text">dashboard</h3>
            </div>

            <div className="btn-container">
              <button className="btn" onClick={() => console.log("show logout")}>
                <FaUserCircle />
                john
                <FaCaretDown />
              </button>
              <div className="dropdown show-dropdown">
                <button
                  onClick={() => console.log("logout user")}
                  className="dropdown-btn"
                >
                  logout
                </button>
              </div>
            </div>
          </div>
        </Wrapper>
      );
    };

    export default Navbar;
````

```
```

#### Toggle Sidebar

```js
actions.js;

export const TOGGLE_SIDEBAR = "TOGGLE_SIDEBAR";
```

- import/export

```js
appContext.js;

const initialState = {
  showSidebar: false,
};

const toggleSidebar = () => {
  dispatch({ type: TOGGLE_SIDEBAR });
};
```

```js
reducer.js;

if (action.type === TOGGLE_SIDEBAR) {
  return { ...state, showSidebar: !state.showSidebar };
}
```

```js
Navbar.js;

const { toggleSidebar } = useAppContext();

return (
  <button className="toggle-btn" onClick={toggleSidebar}>
    <FaAlignLeft />
  </button>
);
```

#### Toggle Dropdown

```js
Navbar.js

const [showLogout, setShowLogout] = useState(false)

<div className='btn-container'>
  <button className='btn' onClick={() => setShowLogout(!showLogout)}>
    <FaUserCircle />
      {user.name}
    <FaCaretDown />
  </button>
  <div className={showLogout ? 'dropdown show-dropdown' : 'dropdown'}>
    <button onClick={() => logoutUser()} className='dropdown-btn'>
      logout
    </button>
```

````
      </div>
    </div>

    ```

    #### Logout User

    ```js
    actions.js;

    export const LOGOUT_USER = "LOGOUT_USER";
    ```

    - import/export

    ```js
    appContext.js

    const logoutUser = () => {
      dispatch({ type: LOGOUT_USER })
      removeUserFromLocalStorage()
    }

    value={{logoutUser}}
    ```

    ```js
    reducer.js;

    import { initialState } from "./appContext";

    if (action.type === LOGOUT_USER) {
      return {
        ...initialState,
        user: null,
        token: null,
        userLocation: "",
        jobLocation: "",
      };
    }
    ```

    ```js
    Navbar.js;

    const { user, logoutUser, toggleSidebar } = useAppContext();

    return (
      <div className="btn-container">
        <button className="btn" onClick={() => setShowLogout(!showLogout)}>
          <FaUserCircle />
          {user.name}
          {user && user.name}
          {user?.name} // optional chaining
          <FaCaretDown />
        </button>
        <div className={showLogout ? "dropdown show-dropdown" : "dropdown"}>
          <button onClick={logoutUser} className="dropdown-btn">
            logout
          </button>
        </div>
````

```
        </div>
      );
      ```

      #### Setup Links

      - create <b>utils</b>in the <b>src</b>
      - setup links.js

      ```js
      import { IoBarChartSharp } from "react-icons/io5";
      import { MdQueryStats } from "react-icons/md";
      import { FaWpforms } from "react-icons/fa";
      import { ImProfile } from "react-icons/im";

      const links = [
        {
          id: 1,
          text: "stats",
          path: "/",
          icon: <IoBarChartSharp />,
        },
        {
          id: 2,
          text: "all jobs",
          path: "all-jobs",
          icon: <MdQueryStats />,
        },
        {
          id: 3,
          text: "add job",
          path: "add-job",
          icon: <FaWpforms />,
        },
        {
          id: 4,
          text: "profile",
          path: "profile",
          icon: <ImProfile />,
        },
      ];

      export default links;
      ```

      #### Small Sidebar - Setup

      ```js
      SmallSidebar.js;

      import Wrapper from "../assets/wrappers/SmallSidebar";
      import { FaTimes } from "react-icons/fa";
      import { useAppContext } from "../context/appContext";
      import links from "../utils/links";
      import { NavLink } from "react-router-dom";
      import Logo from "./Logo";

      export const SmallSidebar = () => {
        return (
          <Wrapper>
            <div className="sidebar-container show-sidebar">
```

```
            <div className="content">
              <button className="close-btn" onClick={() => console.log("toggle")}>
                <FaTimes />
              </button>
              <header>
                <Logo />
              </header>
              <div className="nav-links">nav links</div>
            </div>
          </div>
        </Wrapper>
      );
    };

    export default SmallSidebar;
    ```

    #### Small Sidebar - Toggle

    ```js
    SmallSidebar.js;

    const { showSidebar, toggleSidebar } = useAppContext();
    ```

    ```js
    SmallSidebar.js;

    return (
      <div
        className={
          showSidebar ? "sidebar-container show-sidebar" : "sidebar-container"
        }
      ></div>
    );
    ```

    ```js
    SmallSidebar.js;

    return (
      <button className="close-btn" onClick={toggleSidebar}>
        <FaTimes />
      </button>
    );
    ```

    #### Small Sidebar - Nav Links

    ```js
    SmallSidebar.js;

    import { NavLink } from "react-router-dom";

    return (
      <div className="nav-links">
        {links.map((link) => {
          const { text, path, id, icon } = link;

          return (
            <NavLink
```

```
              to={path}
              className={({ isActive }) =>
                isActive ? "nav-link active" : "nav-link"
              }
              key={id}
              onClick={toggleSidebar}
            >
              <span className="icon">{icon}</span>
              {text}
            </NavLink>
          );
        })}
      </div>
    );
    ```

    #### Nav Links Component

    - in <b>components</b> create NavLinks.js
    - styles still set from Wrapper
    - also can setup in links.js, preference

    ```js
    import { NavLink } from "react-router-dom";
    import links from "../utils/links";

    const NavLinks = ({ toggleSidebar }) => {
      return (
        <div className="nav-links">
          {links.map((link) => {
            const { text, path, id, icon } = link;

            return (
              <NavLink
                to={path}
                key={id}
                onClick={toggleSidebar}
                className={({ isActive }) =>
                  isActive ? "nav-link active" : "nav-link"
                }
              >
                <span className="icon">{icon}</span>
                {text}
              </NavLink>
            );
          })}
        </div>
      );
    };

    export default NavLinks;
    ```

    ```js
    SmallSidebar.js

    import NavLinks from './NavLinks'

    return <NavLinks toggleSidebar={toggleSidebar}>
    ```
```
```

````
#### Big Sidebar

```js
import { useAppContext } from "../context/appContext";
import NavLinks from "./NavLinks";
import Logo from "../components/Logo";
import Wrapper from "../assets/wrappers/BigSidebar";

const BigSidebar = () => {
  const { showSidebar } = useAppContext();
  return (
    <Wrapper>
      <div
        className={
          showSidebar ? "sidebar-container " : "sidebar-container show-sidebar"
        }
      >
        <div className="content">
          <header>
            <Logo />
          </header>
          <NavLinks />
        </div>
      </div>
    </Wrapper>
  );
};

export default BigSidebar;
```

#### REACT ROUTER UPDATE !!!

- [Stack Overflow](https://stackoverflow.com/questions/70644361/react-router-dom-v6-shows-active-for-index-as-well-as-other-subroutes)

```js
<NavLink
to={path}
key={id}
onClick={toggleSidebar}
className={({ isActive }) =>
isActive ? 'nav-link active' : 'nav-link'}


end
>
```

#### Authenticate User Setup

- create auth.js in <b>middleware</b>

```js
const auth = async (req, res, next) => {
  console.log("authenticate user");
  next();
};

export default auth;
```
````

````
```js
authRoutes.js;

import authenticateUser from "../middleware/auth.js";

router.route("/updateUser").patch(authenticateUser, updateUser);
```

- two options

```js
server.js;

import authenticateUser from "./middleware/auth.js";
app.use("/api/v1/jobs", authenticateUser, jobsRouter);
```

```js
jobsRoutes.js;

import authenticateUser from "./middleware/auth.js";

// all routes !!!!

router.route("/stats").get(authenticateUser, showStats);
```

#### Auth - Bearer Schema

```js
Postman

Headers

Authorization: Bearer <token>

```

```js
auth.js;

const auth = async (req, res, next) => {
  const headers = req.headers;
  const authHeader = req.headers.authorization;
  console.log(headers);
  console.log(authHeader);
  next();
};
```

#### Postman - Set Token Programmatically

- register and login routes
- Tests

```js
const jsonData = pm.response.json();
pm.globals.set("token", jsonData.token);

Type: Bearer;
````
````

```
Token: {
  {
    token;
  }
}
```

#### Unauthenticated Error

```js
auth.js;

import { UnAuthenticatedError } from "../errors/index.js";

const auth = async (req, res, next) => {
  const authHeader = req.headers.authorization;

  if (!authHeader) {
    // why, well is it 400 or 404?
    // actually 401
    throw new UnAuthenticatedError("Authentication Invalid");
  }

  next();
};
```

#### Auth Middleware

```js
import jwt from "jsonwebtoken";
import { UnAuthenticatedError } from "../errors/index.js";

const auth = async (req, res, next) => {
  // check header
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith("Bearer")) {
    throw new UnauthenticatedError("Authentication invalid");
  }
  const token = authHeader.split(" ")[1];

  try {
    const payload = jwt.verify(token, process.env.JWT_SECRET);
    // console.log(payload)
    // attach the user request object
    // req.user = payload
    req.user = { userId: payload.userId };
    next();
  } catch (error) {
    throw new UnauthenticatedError("Authentication invalid");
  }
};

export default auth;
```

#### Update User

```js
const updateUser = async (req, res) => {
```

```
  const { email, name, lastName, location } = req.body;
  if (!email || !name || !lastName || !location) {
    throw new BadRequestError("Please provide all values");
  }

  const user = await User.findOne({ _id: req.user.userId });

  user.email = email;
  user.name = name;
  user.lastName = lastName;
  user.location = location;

  await user.save();

  // various setups
  // in this case only id
  // if other properties included, must re-generate

  const token = user.createJWT();
  res.status(StatusCodes.OK).json({
    user,
    token,
    location: user.location,
  });
};
```

#### Modified Paths

- user.save() vs User.findOneAndUpdate

```js
User.js;

UserSchema.pre("save", async function () {
  console.log(this.modifiedPaths());
  console.log(this.isModified("name"));

  // if (!this.isModified('password')) return
  // const salt = await bcrypt.genSalt(10)
  // this.password = await bcrypt.hash(this.password, salt)
});
```

#### Profile Page

```js
appContext.js

const updateUser = async (currentUser) => {
  console.log(currentUser)
}

value={{updateUser}}
```

```js
Profile.js;

import { useState } from "react";
import { FormRow, Alert } from "../../components";
```

```
import { useAppContext } from "../../context/appContext";
import Wrapper from "../../assets/wrappers/DashboardFormPage";

const Profile = () => {
  const { user, showAlert, displayAlert, updateUser, isLoading } =
    useAppContext();
  const [name, setName] = useState(user?.name);
  const [email, setEmail] = useState(user?.email);
  const [lastName, setLastName] = useState(user?.lastName);
  const [location, setLocation] = useState(user?.location);

  const handleSubmit = (e) => {
    e.preventDefault();
    if (!name || !email || !lastName || !location) {
      // test and remove temporary
      displayAlert();
      return;
    }

    updateUser({ name, email, lastName, location });
  };
  return (
    <Wrapper>
      <form className="form" onSubmit={handleSubmit}>
        <h3>profile </h3>
        {showAlert && <Alert />}

        {/* name */}
        <div className="form-center">
          <FormRow
            type="text"
            name="name"
            value={name}
            handleChange={(e) => setName(e.target.value)}
          />
          <FormRow
            labelText="last name"
            type="text"
            name="lastName"
            value={lastName}
            handleChange={(e) => setLastName(e.target.value)}
          />
          <FormRow
            type="email"
            name="email"
            value={email}
            handleChange={(e) => setEmail(e.target.value)}
          />

          <FormRow
            type="text"
            name="location"
            value={location}
            handleChange={(e) => setLocation(e.target.value)}
          />
          <button className="btn btn-block" type="submit" disabled={isLoading}>
            {isLoading ? "Please Wait..." : "save changes"}
          </button>
        </div>
      </form>
    </Wrapper>
```

```
    );
  };

  export default Profile;
  ```

  #### Bearer Token - Manual Approach

  ```js
  appContext.js;

  const updaterUser = async (currentUser) => {
    try {
      const { data } = await axios.patch("/api/v1/auth/updateUser", currentUser, {
        headers: {
          Authorization: `Bearer ${state.token}`,
        },
      });
      console.log(data);
    } catch (error) {
      console.log(error.response);
    }
  };
  ```

  #### Axios - Global Setup

  <!-- IMPORTANT  -->

  In current axios version,
  common property returns undefined,
  so we don't use it anymore!!!

  ```js
  appContext.js;

  axios.defaults.headers["Authorization"] = `Bearer ${state.token}`;
  ```

  #### Axios - Setup Instance

  ```js
  AppContext.js;

  const authFetch = axios.create({
    baseURL: "/api/v1",
    headers: {
      Authorization: `Bearer ${state.token}`,
    },
  });

  const updaterUser = async (currentUser) => {
    try {
      const { data } = await authFetch.patch("/auth/updateUser", currentUser);
    } catch (error) {
      console.log(error.response);
    }
  };
  ```

  #### 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.           });
````

```
      addUserToLocalStorage({ user, location, token });
    } catch (error) {
      dispatch({
        type: UPDATE_USER_ERROR,
        payload: { msg: error.response.data.msg },
      });
    }
    clearAlert();
  };
```

```js
reducer.js
if (action.type === UPDATE_USER_BEGIN) {
  return { ...state, isLoading: true }
}

if (action.type === UPDATE_USER_SUCCESS) {
  return {
    ...state,
    isLoading: false,
    token:action.payload.token
    user: action.payload.user,
    userLocation: action.payload.location,
    jobLocation: action.payload.location,
    showAlert: true,
    alertType: 'success',
    alertText: 'User Profile Updated!',
  }
}
if (action.type === UPDATE_USER_ERROR) {
  return {
    ...state,
    isLoading: false,
    showAlert: true,
    alertType: 'danger',
    alertText: action.payload.msg,
  }
}
```

#### 401 Error - Logout User

```js
appContext.js;
// response interceptor
authFetch.interceptors.response.use(
  (response) => {
    return response;
  },
  (error) => {
    if (error.response.status === 401) {
      logoutUser();
    }
    return Promise.reject(error);
  }
);

const updateUser = async (currentUser) => {
  dispatch({ type: UPDATE_USER_BEGIN });
```

```
      try {
        const { data } = await authFetch.patch("/auth/updateUser", currentUser);

        // no token
        const { user, location } = data;

        dispatch({
          type: UPDATE_USER_SUCCESS,
          payload: { user, location, token },
        });

        addUserToLocalStorage({ user, location, token: initialState.token });
      } catch (error) {
        if (error.response.status !== 401) {
          dispatch({
            type: UPDATE_USER_ERROR,
            payload: { msg: error.response.data.msg },
          });
        }
      }
      clearAlert();
    };
    ```

    #### Job Model

    - Job Model

    ```js
    Job.js;

    import mongoose from "mongoose";

    const JobSchema = new mongoose.Schema(
      {
        company: {
          type: String,
          required: [true, "Please provide company name"],
          maxlength: 50,
        },
        position: {
          type: String,
          required: [true, "Please provide position"],
          maxlength: 100,
        },
        status: {
          type: String,
          enum: ["interview", "declined", "pending"],
          default: "pending",
        },

        jobType: {
          type: String,
          enum: ["full-time", "part-time", "remote", "internship"],
          default: "full-time",
        },
        jobLocation: {
          type: String,
          default: "my city",
          required: true,
        },
```

```
      createdBy: {
        type: mongoose.Types.ObjectId,
        ref: "User",
        required: [true, "Please provide user"],
      },
    },
    { timestamps: true }
  );

  export default mongoose.model("Job", JobSchema);
  ```

  #### Create Job

  ```js
  jobsController.js;

  import Job from "../models/Job.js";
  import { StatusCodes } from "http-status-codes";
  import { BadRequestError, NotFoundError } from "../errors/index.js";

  const createJob = async (req, res) => {
    const { position, company } = req.body;

    if (!position || !company) {
      throw new BadRequestError("Please Provide All Values");
    }

    req.body.createdBy = req.user.userId;

    const job = await Job.create(req.body);
    res.status(StatusCodes.CREATED).json({ job });
  };
  ```

  #### Job State Values

  ```js
  appContext.js;
  const initialState = {
    isEditing: false,
    editJobId: "",
    position: "",
    company: "",
    // jobLocation
    jobTypeOptions: ["full-time", "part-time", "remote", "internship"],
    jobType: "full-time",
    statusOptions: ["pending", "interview", "declined"],
    status: "pending",
  };
  ```

  #### AddJob Page - Setup

  ```js
  import { FormRow, Alert } from "../../components";
  import { useAppContext } from "../../context/appContext";
  import Wrapper from "../../assets/wrappers/DashboardFormPage";
  const AddJob = () => {
    const {
      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">
```

```
                <button
                  className="btn btn-block submit-btn"
                  type="submit"
                  onClick={handleSubmit}
                >
                  submit
                </button>
              </div>
            </div>
          </form>
        </Wrapper>
      );
    };

    export default AddJob;
    ```

    #### Select Input

    ```js
    return (
      // job type
      <div className="form-row">
        <label htmlFor="jobType" className="form-label">
          job type
        </label>

        <select
          name="jobType"
          value={jobType}
          onChange={handleJobInput}
          className="form-select"
        >
          {jobTypeOptions.map((itemValue, index) => {
            return (
              <option key={index} value={itemValue}>
                {itemValue}
              </option>
            );
          })}
        </select>
      </div>
    );
    ```

    #### FormRowSelect

    - create FormRowSelect in components
    - setup import/export

    ```js
    const FormRowSelect = ({ labelText, name, value, handleChange, list }) => {
      return (
        <div className="form-row">
          <label htmlFor={name} className="form-label">
            {labelText || name}
          </label>

          <select
            name={name}
            value={value}
```

```
            onChange={handleChange}
            className="form-select"
          >
            {list.map((itemValue, index) => {
              return (
                <option key={index} value={itemValue}>
                  {itemValue}
                </option>
              );
            })}
          </select>
        </div>
      );
    };

    export default FormRowSelect;
    ```

    ```js
    AddJob.js;

    return (
      <>
        {/* job status */}

        <FormRowSelect
          name="status"
          value={status}
          handleChange={handleJobInput}
          list={statusOptions}
        />

        {/* job type */}
        <FormRowSelect
          labelText="type"
          name="jobType"
          value={jobType}
          handleChange={handleJobInput}
          list={jobTypeOptions}
        />
      </>
    );
    ```

    #### Change State Values With Handle Change

    - [JS Nuggets Dynamic Object Keys](https://youtu.be/_qxCYtWm0tw)

    ```js
    actions.js;

    export const HANDLE_CHANGE = "HANDLE_CHANGE";
    ```

    ```js
    appContext.js

    const handleChange = ({ name, value }) => {
      dispatch({
        type: HANDLE_CHANGE,
        payload: { name, value },
```

```
      })
    }

    value={{handleChange}}
    ```

    ```js
    reducer.js;

    if (action.type === HANDLE_CHANGE) {
      return { ...state, [action.payload.name]: action.payload.value };
    }
    ```

    ```js
    AddJob.js;

    const { handleChange } = useAppContext();

    const handleJobInput = (e) => {
      handleChange({ name: e.target.name, value: e.target.value });
    };
    ```

    #### Clear Values

    ```js
    actions.js;

    export const CLEAR_VALUES = "CLEAR_VALUES";
    ```

    ```js
    appContext.js

    const clearValues = () => {
        dispatch({ type: CLEAR_VALUES })
      }

    value={{clearValues}}
    ```

    ```js
    reducer.js;

    if (action.type === CLEAR_VALUES) {
      const initialState = {
        isEditing: false,
        editJobId: "",
        position: "",
        company: "",
        jobLocation: state.userLocation,
        jobType: "full-time",
        status: "pending",
      };
      return { ...state, ...initialState };
    }
    ```

    ```js
    AddJob.js;
```

````
    const { clearValues } = useAppContext();

    return (
      <div className="btn-container">
        {/* submit button */}

        <button
          className="btn btn-block clear-btn"
          onClick={(e) => {
            e.preventDefault();
            clearValues();
          }}
        >
          clear
        </button>
      </div>
    );
    ```

    #### Create Job

    ```js
    actions.js;

    export const CREATE_JOB_BEGIN = "CREATE_JOB_BEGIN";
    export const CREATE_JOB_SUCCESS = "CREATE_JOB_SUCCESS";
    export const CREATE_JOB_ERROR = "CREATE_JOB_ERROR";
    ```

    ```js
    appContext.js;

    const createJob = async () => {
      dispatch({ type: CREATE_JOB_BEGIN });
      try {
        const { position, company, jobLocation, jobType, status } = state;

        await authFetch.post("/jobs", {
          company,
          position,
          jobLocation,
          jobType,
          status,
        });
        dispatch({
          type: CREATE_JOB_SUCCESS,
        });
        // call function instead clearValues()
        dispatch({ type: CLEAR_VALUES });
      } catch (error) {
        if (error.response.status === 401) return;
        dispatch({
          type: CREATE_JOB_ERROR,
          payload: { msg: error.response.data.msg },
        });
      }
      clearAlert();
    };
    ```
````
````

````
```js
AddJob.js;

const { createJob } = useAppContext();

const handleSubmit = (e) => {
  e.preventDefault();
  // while testing

  // if (!position || !company || !jobLocation) {
  //   displayAlert()
  //   return
  // }
  if (isEditing) {
    // eventually editJob()
    return;
  }
  createJob();
};
```

```js
reducer.js;

if (action.type === CREATE_JOB_BEGIN) {
  return { ...state, isLoading: true };
}
if (action.type === CREATE_JOB_SUCCESS) {
  return {
    ...state,
    isLoading: false,
    showAlert: true,
    alertType: "success",
    alertText: "New Job Created!",
  };
}
if (action.type === CREATE_JOB_ERROR) {
  return {
    ...state,
    isLoading: false,
    showAlert: true,
    alertType: "danger",
    alertText: action.payload.msg,
  };
}
```

#### Get All Jobs

```js
jobsController.js;

const getAllJobs = async (req, res) => {
  const jobs = await Job.find({ createdBy: req.user.userId });

  res
    .status(StatusCodes.OK)
    .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
};
```
````

````
#### Jobs State Values

```js
appContext.js;

const initialState = {
  jobs: [],
  totalJobs: 0,
  numOfPages: 1,
  page: 1,
};
```

#### Get All Jobs Request

```js
actions.js;
export const GET_JOBS_BEGIN = "GET_JOBS_BEGIN";
export const GET_JOBS_SUCCESS = "GET_JOBS_SUCCESS";
```

```js
appContext.js

import React, { useReducer, useContext, useEffect } from 'react'

const getJobs = async () => {
  let url = `/jobs`

  dispatch({ type: GET_JOBS_BEGIN })
  try {
    const { data } = await authFetch(url)
    const { jobs, totalJobs, numOfPages } = data
    dispatch({
      type: GET_JOBS_SUCCESS,
      payload: {
        jobs,
        totalJobs,
        numOfPages,
      },
    })
  } catch (error) {
    console.log(error.response)
    logoutUser()
  }
  clearAlert()
}

useEffect(() => {
  getJobs()
}, [])

value={{getJobs}}

```

```js
reducer.js;

if (action.type === GET_JOBS_BEGIN) {
  return { ...state, isLoading: true, showAlert: false };
````
````

```
    }
    if (action.type === GET_JOBS_SUCCESS) {
      return {
        ...state,
        isLoading: false,
        jobs: action.payload.jobs,
        totalJobs: action.payload.totalJobs,
        numOfPages: action.payload.numOfPages,
      };
    }
    ```

    #### AllJobs Page Setup

    - create
    - SearchContainer export
    - JobsContainer export
    - Job
    - JobInfo

    ```js
    AllJobs.js;

    import { JobsContainer, SearchContainer } from "../../components";
    const AllJobs = () => {
      return (
        <>
          <SearchContainer />
          <JobsContainer />
        </>
      );
    };

    export default AllJobs;
    ```

    ```js
    JobsContainer.js;
    import { useAppContext } from "../context/appContext";
    import { useEffect } from "react";
    import Loading from "./Loading";
    import Job from "./Job";
    import Wrapper from "../assets/wrappers/JobsContainer";

    const JobsContainer = () => {
      const { getJobs, jobs, isLoading, page, totalJobs } = useAppContext();
      useEffect(() => {
        getJobs();
      }, []);

      if (isLoading) {
        return <Loading center />;
      }
      if (jobs.length === 0) {
        return (
          <Wrapper>
            <h2>No jobs to display...</h2>
          </Wrapper>
        );
      }
      return (
```

```
        <Wrapper>
          <h5>
            {totalJobs} job{jobs.length > 1 && "s"} found
          </h5>
          <div className="jobs">
            {jobs.map((job) => {
              return <Job key={job._id} {...job} />;
            })}
          </div>
        </Wrapper>
      );
    };

    export default JobsContainer;
    ```

    ```js
    Job.js;

    import moment from "moment";

    const Job = ({ company }) => {
      return <h5>{company}</h5>;
    };

    export default Job;
    ```

    #### Moment.js

    - Format Dates
    - [moment.js](https://momentjs.com/)

    - stop server
    - cd client

    ```sh
    npm install moment

    ```

    ```js
    Job.js;

    import moment from "moment";

    const Job = ({ company, createdAt }) => {
      let date = moment(createdAt);
      date = date.format("MMM Do, YYYY");
      return (
        <div>
          <h5>{company}</h5>
          <h5>{date}</h5>
        </div>
      );
    };

    export default Job;
    ```

    #### Job Component - Setup
```
```

````
```js
appContext.js

const setEditJob = (id) => {
  console.log(`set edit job : ${id}`)
}
const deleteJob = (id) =>{
  console.log(`delete : ${id}`)
}
value={{setEditJob,deleteJob}}
```

```js
Job.js;

import { FaLocationArrow, FaBriefcase, FaCalendarAlt } from "react-icons/fa";
import { Link } from "react-router-dom";
import { useAppContext } from "../context/appContext";
import Wrapper from "../assets/wrappers/Job";
import JobInfo from "./JobInfo";

const Job = ({
  _id,
  position,
  company,
  jobLocation,
  jobType,
  createdAt,
  status,
}) => {
  const { setEditJob, deleteJob } = useAppContext();

  let date = moment(createdAt);
  date = date.format("MMM Do, YYYY");

  return (
    <Wrapper>
      <header>
        <div className="main-icon">{company.charAt(0)}</div>
        <div className="info">
          <h5>{position}</h5>
          <p>{company}</p>
        </div>
      </header>
      <div className="content">
        {/* content center later */}
        <footer>
          <div className="actions">
            <Link
              to="/add-job"
              onClick={() => setEditJob(_id)}
              className="btn edit-btn"
            >
              Edit
            </Link>
            <button
              type="button"
              className="btn delete-btn"
              onClick={() => deleteJob(_id)}
            >
````
````

```
                  Delete
                </button>
              </div>
            </footer>
          </div>
        </Wrapper>
      );
    };

    export default Job;
    ```

    #### JobInfo

    ```js
    JobInfo.js;

    import Wrapper from "../assets/wrappers/JobInfo";

    const JobInfo = ({ icon, text }) => {
      return (
        <Wrapper>
          <span className="icon">{icon}</span>
          <span className="text">{text}</span>
        </Wrapper>
      );
    };

    export default JobInfo;
    ```

    ```js
    Job.js;
    return (
      <div className="content">
        <div className="content-center">
          <JobInfo icon={<FaLocationArrow />} text={jobLocation} />
          <JobInfo icon={<FaCalendarAlt />} text={date} />
          <JobInfo icon={<FaBriefcase />} text={jobType} />
          <div className={`status ${status}`}>{status}</div>
        </div>
        {/* footer content */}
      </div>
    );
    ```

    #### SetEditJob

    ```js
    actions.js;
    export const SET_EDIT_JOB = "SET_EDIT_JOB";
    ```

    ```js
    appContext.js

    const setEditJob = (id) => {
      dispatch({ type: SET_EDIT_JOB, payload: { id } })
    }
    const editJob = () => {
      console.log('edit job')
```

````
      }
      value={{editJob}}
      ```

      ```js
      reducer.js;

      if (action.type === SET_EDIT_JOB) {
        const job = state.jobs.find((job) => job._id === action.payload.id);
        const { _id, position, company, jobLocation, jobType, status } = job;
        return {
          ...state,
          isEditing: true,
          editJobId: _id,
          position,
          company,
          jobLocation,
          jobType,
          status,
        };
      }
      ```

      ```js
      AddJob.js;
      const { isEditing, editJob } = useAppContext();
      const handleSubmit = (e) => {
        e.preventDefault();

        if (!position || !company || !jobLocation) {
          displayAlert();
          return;
        }
        if (isEditing) {
          editJob();
          return;
        }
        createJob();
      };
      ```

      #### Edit Job - Server

      ```js
      jobsController.js;

      const updateJob = async (req, res) => {
        const { id: jobId } = req.params;

        const { company, position } = req.body;

        if (!company || !position) {
          throw new BadRequestError("Please Provide All Values");
        }

        const job = await Job.findOne({ _id: jobId });

        if (!job) {
          throw new NotFoundError(`No job with id ${jobId}`);
        }
````

```
    // check permissions

    const updatedJob = await Job.findOneAndUpdate({ _id: jobId }, req.body, {
      new: true,
      runValidators: true,
    });

    res.status(StatusCodes.OK).json({ updatedJob });
};
```

#### Alternative Approach

- optional
- multiple approaches
- different setups
- course Q&A

```js
jobsController.js;
const updateJob = async (req, res) => {
  const { id: jobId } = req.params;
  const { company, position, jobLocation } = req.body;

  if (!position || !company) {
    throw new BadRequestError("Please provide all values");
  }
  const job = await Job.findOne({ _id: jobId });

  if (!job) {
    throw new NotFoundError(`No job with id :${jobId}`);
  }

  // check permissions

  // alternative approach

  job.position = position;
  job.company = company;
  job.jobLocation = jobLocation;

  await job.save();
  res.status(StatusCodes.OK).json({ job });
};
```

#### Check Permissions

```js
jobsController.js;

const updateJob = async (req, res) => {
  const { id: jobId } = req.params;
  const { company, position, status } = req.body;

  if (!position || !company) {
    throw new BadRequestError("Please provide all values");
  }
  const job = await Job.findOne({ _id: jobId });

  if (!job) {
```

````
      throw new NotFoundError(`No job with id :${jobId}`);
    }

    // check permissions
    // req.user.userId (string) === job.createdBy(object)
    // throw new UnAuthenticatedError('Not authorized to access this route')

    // console.log(typeof req.user.userId)
    // console.log(typeof job.createdBy)

    checkPermissions(req.user, job.createdBy);

    const updatedJob = await Job.findOneAndUpdate({ _id: jobId }, req.body, {
      new: true,
      runValidators: true,
    });

    res.status(StatusCodes.OK).json({ updatedJob });
  };
  ```

  - utils folder
  - checkPermissions.js
  - import in jobsController.js

  ```js
  checkPermissions.js;

  import { UnAuthenticatedError } from "../errors/index.js";

  const checkPermissions = (requestUser, resourceUserId) => {
    // if (requestUser.role === 'admin') return
    if (requestUser.userId === resourceUserId.toString()) return;
    throw new UnauthorizedError("Not authorized to access this route");
  };

  export default checkPermissions;
  ```

  #### Remove/Delete Job

  ```js
  jobsController.js;

  const deleteJob = async (req, res) => {
    const { id: jobId } = req.params;

    const job = await Job.findOne({ _id: jobId });

    if (!job) {
      throw new NotFoundError(`No job with id : ${jobId}`);
    }

    checkPermissions(req.user, job.createdBy);

    await job.remove();
    res.status(StatusCodes.OK).json({ msg: "Success! Job removed" });
  };
  ```

  #### 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 },
````

```
        });
      }
      clearAlert();
    };
    ```

    ```js
    reducer.js;

    if (action.type === EDIT_JOB_BEGIN) {
      return { ...state, isLoading: true };
    }
    if (action.type === EDIT_JOB_SUCCESS) {
      return {
        ...state,
        isLoading: false,
        showAlert: true,
        alertType: "success",
        alertText: "Job Updated!",
      };
    }
    if (action.type === EDIT_JOB_ERROR) {
      return {
        ...state,
        isLoading: false,
        showAlert: true,
        alertType: "danger",
        alertText: action.payload.msg,
      };
    }
    ```

    #### Create More Jobs

    - [Mockaroo](https://www.mockaroo.com/)
    - setup mock-data.json in the root

    #### Populate Database

    - create populate.js in the root

    ```js
    populate.js;

    import { readFile } from "fs/promises";

    import dotenv from "dotenv";
    dotenv.config();

    import connectDB from "./db/connect.js";
    import Job from "./models/Job.js";

    const start = async () => {
      try {
        await connectDB(process.env.MONGO_URL);
        await Job.deleteMany();

        const jsonProducts = JSON.parse(
          await readFile(new URL("./mock-data.json", import.meta.url))
        );
        await Job.create(jsonProducts);
```

```
      console.log("Success!!!!");
      process.exit(0);
    } catch (error) {
      console.log(error);
      process.exit(1);
    }
};

start();
```

#### Show Stats - Structure

- aggregation pipeline
- step by step
- [Aggregation Pipeline](https://docs.mongodb.com/manual/core/aggregation-pipeline/)

```js
jobsController.js;

import mongoose from "mongoose";

const showStats = async (req, res) => {
  let stats = await Job.aggregate([
    { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
    { $group: { _id: "$status", count: { $sum: 1 } } },
  ]);

  res.status(StatusCodes.OK).json({ stats });
};
```

#### Show Stats - Object Setup

- [Reduce Basics](https://youtu.be/3WkW9nrS2mw)
- [Reduce Object Example ](https://youtu.be/5BFkp8JjLEY)

```js
jobsController.js;

const showStats = async (req, res) => {
  let stats = await Job.aggregate([
    { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
    { $group: { _id: "$status", count: { $sum: 1 } } },
  ]);

  stats = stats.reduce((acc, curr) => {
    const { _id: title, count } = curr;
    acc[title] = count;
    return acc;
  }, {});

  res.status(StatusCodes.OK).json({ stats });
};
```

#### Show Stats - Default Stats

```js
jobsController.js;
```

````
const showStats = async (req, res) => {
  let stats = await Job.aggregate([
    { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
    { $group: { _id: "$status", count: { $sum: 1 } } },
  ]);
  stats = stats.reduce((acc, curr) => {
    const { _id: title, count } = curr;
    acc[title] = count;
    return acc;
  }, {});

  const defaultStats = {
    pending: stats.pending || 0,
    interview: stats.interview || 0,
    declined: stats.declined || 0,
  };
  let monthlyApplications = [];
  res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications });
};
```

#### Show Stats - Function Setup

```js
actions.js;

export const SHOW_STATS_BEGIN = "SHOW_STATS_BEGIN";
export const SHOW_STATS_SUCCESS = "SHOW_STATS_SUCCESS";
```

```js
appContext.js

const initialState = {
  stats: {},
  monthlyApplications: []

}

const showStats = async () => {
    dispatch({ type: SHOW_STATS_BEGIN })
    try {
      const { data } = await authFetch('/jobs/stats')
      dispatch({
        type: SHOW_STATS_SUCCESS,
        payload: {
          stats: data.defaultStats,
          monthlyApplications: data.monthlyApplications,
        },
      })
    } catch (error) {
console.log(error.response)
      // logoutUser()
    }

clearAlert()
  }
  value={{showStats}}
```

```js
````

````
reducers.js;
if (action.type === SHOW_STATS_BEGIN) {
  return { ...state, isLoading: true, showAlert: false };
}
if (action.type === SHOW_STATS_SUCCESS) {
  return {
    ...state,
    isLoading: false,
    stats: action.payload.stats,
    monthlyApplications: action.payload.monthlyApplications,
  };
}
```

#### Stats Page - Structure

- components
- StatsContainer.js
- ChartsContainer.js
- StatsItem.js
- simple return
- import/export index.js

```js
Stats.js;

import { useEffect } from "react";
import { useAppContext } from "../../context/appContext";
import { StatsContainer, Loading, ChartsContainer } from "../../components";

const Stats = () => {
  const { showStats, isLoading, monthlyApplications } = useAppContext();
  useEffect(() => {
    showStats();
  }, []);

  if (isLoading) {
    return <Loading center />;
  }

  return (
    <>
      <StatsContainer />
      {monthlyApplications.length > 0 && <ChartsContainer />}
    </>
  );
};

export default Stats;
```

#### StatsContainer

```js
StatsContainer.js;

import { useAppContext } from "../context/appContext";
import StatItem from "./StatItem";
import { FaSuitcaseRolling, FaCalendarCheck, FaBug } from "react-icons/fa";
import Wrapper from "../assets/wrappers/StatsContainer";
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.
```

````
```js
jobsController.js;

let monthlyApplications = await Job.aggregate([
  { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
  {
    $group: {
      _id: {
        year: {
          $year: "$createdAt",
        },
        month: {
          $month: "$createdAt",
        },
      },
      count: { $sum: 1 },
    },
  },
  { $sort: { "_id.year": -1, "_id.month": -1 } },
  { $limit: 6 },
]);
```

#### Refactor Data

- install moment.js on the SERVER

```sh
npm install moment

```

```js
jobsController.js;

import moment from "moment";

monthlyApplications = monthlyApplications
  .map((item) => {
    const {
      _id: { year, month },
      count,
    } = item;
    // accepts 0-11
    const date = moment()
      .month(month - 1)
      .year(year)
      .format("MMM Y");
    return { date, count };
  })
  .reverse();
```

#### Charts Container

- BarChart.js
- AreaChart.js

```js
ChartsContainer.js;
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 />
````

```
              <Bar dataKey="count" fill="#2cb1bc" barSize={75} />
            </BarChart>
          </ResponsiveContainer>
        );
      };
      ```

      #### Area Chart

      ```js
      import {
        ResponsiveContainer,
        AreaChart,
        Area,
        XAxis,
        YAxis,
        CartesianGrid,
        Tooltip,
      } from "recharts";

      const AreaChartComponent = ({ data }) => {
        return (
          <ResponsiveContainer width="100%" height={300}>
            <AreaChart
              data={data}
              margin={{
                top: 50,
              }}
            >
              <CartesianGrid strokeDasharray="3 3" />
              <XAxis dataKey="date" />
              <YAxis allowDecimals={false} />
              <Tooltip />
              <Area type="monotone" dataKey="count" stroke="#2cb1bc" fill="#bef8fd" />
            </AreaChart>
          </ResponsiveContainer>
        );
      };
      ```

      #### Filter

      #### Get All Jobs - Initial Setup

      ```js
      jobsController.js;

      const getAllJobs = async (req, res) => {
        const { search, status, jobType, sort } = req.query;

        const queryObject = {
          createdBy: req.user.userId,
        };

        // NO AWAIT
        let result = Job.find(queryObject);

        // chain sort conditions

        const jobs = await result;

```

```
        res
          .status(StatusCodes.OK)
          .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
      };
      ```

      #### Status

      ```js
      jobsController.js;

      const getAllJobs = async (req, res) => {
        const { search, status, jobType, sort } = req.query;

        const queryObject = {
          createdBy: req.user.userId,
        };

        if (status !== "all") {
          queryObject.status = status;
        }

        // NO AWAIT
        let result = Job.find(queryObject);

        // chain sort conditions

        const jobs = await result;

        res
          .status(StatusCodes.OK)
          .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
      };
      ```

      #### JobType

      ```js
      jobsController.js;

      const getAllJobs = async (req, res) => {
        const { search, status, jobType, sort } = req.query;

        const queryObject = {
          createdBy: req.user.userId,
        };

        if (status !== "all") {
          queryObject.status = status;
        }
        if (jobType !== "all") {
          queryObject.jobType = jobType;
        }
        // NO AWAIT
        let result = Job.find(queryObject);

        // chain sort conditions

        const jobs = await result;

        res
```

```
        .status(StatusCodes.OK)
        .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
    };
    ```

    #### Search

    ```js
    jobsController.js;

    const getAllJobs = async (req, res) => {
      const { search, status, jobType, sort } = req.query;

      const queryObject = {
        createdBy: req.user.userId,
      };

      if (status !== "all") {
        queryObject.status = status;
      }
      if (jobType !== "all") {
        queryObject.jobType = jobType;
      }
      if (search) {
        queryObject.position = { $regex: search, $options: "i" };
      }
      // NO AWAIT
      let result = Job.find(queryObject);

      // chain sort conditions
      if (sort === "latest") {
        result = result.sort("-createdAt");
      }
      if (sort === "oldest") {
        result = result.sort("createdAt");
      }
      if (sort === "a-z") {
        result = result.sort("position");
      }
      if (sort === "z-a") {
        result = result.sort("-position");
      }
      const jobs = await result;

      res
        .status(StatusCodes.OK)
        .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
    };
    ```

    #### Search Context Setup

    ```js
    appContext.js

    const initialState = {
      jobType: 'full-time',
      jobTypeOptions: ['full-time', 'part-time', 'remote', 'internship'],
      status: 'pending',
      statusOptions: ['pending', 'interview', 'declined']
      //
```

```
        //
        //
        search: '',
        searchStatus: 'all',
        searchType: 'all',
        sort: 'latest',
        sortOptions: ['latest', 'oldest', 'a-z', 'z-a'],
      }

      const clearFilters = () =>{
      console.log('clear filters')
      }

      value={{clearFilters}}

      // remember this function :)
      const handleChange = ({ name, value }) => {
          dispatch({
            type: HANDLE_CHANGE,
            payload: { name, value },
          })
        }

      ```

      #### Search Container - Setup

      ```js
      SearchContainer.js;

      import { FormRow, FormRowSelect } from ".";
      import { useAppContext } from "../context/appContext";
      import Wrapper from "../assets/wrappers/SearchContainer";
      const SearchContainer = () => {
        const {
          isLoading,
          search,
          searchStatus,
          searchType,
          sort,
          sortOptions,
          statusOptions,
          jobTypeOptions,
          handleChange,
          clearFilters,
        } = useAppContext();

        const handleSearch = (e) => {
          if (isLoading) return;
          handleChange({ name: e.target.name, value: e.target.value });
        };

        return (
          <Wrapper>
            <form className="form">
              <h4>search form</h4>
              {/* search position */}
              <div className="form-center">
                <FormRow
                  type="text"
                  name="search"
```

```
                value={search}
                handleChange={handleSearch}
              ></FormRow>
              {/* rest of the inputs */}
            </div>
          </form>
        </Wrapper>
      );
    };

    export default SearchContainer;
    ```

    #### Search Container - Complete

    ```js
    SearchContainer.js;

    import { FormRow, FormRowSelect } from ".";
    import { useAppContext } from "../context/appContext";
    import Wrapper from "../assets/wrappers/SearchContainer";

    const SearchContainer = () => {
      const {
        isLoading,
        search,
        handleChange,
        searchStatus,
        statusOptions,
        jobTypeOptions,
        searchType,
        clearFilters,
        sort,
        sortOptions,
      } = useAppContext();

      const handleSearch = (e) => {
        if (isLoading) return;
        handleChange({ name: e.target.name, value: e.target.value });
      };
      const handleSubmit = (e) => {
        e.preventDefault();
        clearFilters();
      };
      return (
        <Wrapper>
          <form className="form">
            <h4>search form</h4>
            {/* search position */}
            <div className="form-center">
              <FormRow
                type="text"
                name="search"
                value={search}
                handleChange={handleSearch}
              ></FormRow>
              {/* search by status */}
              <FormRowSelect
                labelText="job status"
                name="searchStatus"
                value={searchStatus}
```

```
                handleChange={handleSearch}
                list={["all", ...statusOptions]}
              ></FormRowSelect>
              {/* search by type */}

              <FormRowSelect
                labelText="job type"
                name="searchType"
                value={searchType}
                handleChange={handleSearch}
                list={["all", ...jobTypeOptions]}
              ></FormRowSelect>
              {/* sort */}

              <FormRowSelect
                name="sort"
                value={sort}
                handleChange={handleSearch}
                list={sortOptions}
              ></FormRowSelect>
              <button
                className="btn btn-block btn-danger"
                disabled={isLoading}
                onClick={handleSubmit}
              >
                clear filters
              </button>
            </div>
          </form>
        </Wrapper>
      );
    };

    export default SearchContainer;
    ```

    #### Clear Filters

    ```js
    actions.js;

    export const CLEAR_FILTERS = "CLEAR_FILTERS";
    ```

    ```js
    appContext.js;

    const clearFilters = () => {
      dispatch({ type: CLEAR_FILTERS });
    };
    ```

    ```js
    reducer.js;

    if (action.type === CLEAR_FILTERS) {
      return {
        ...state,
        search: "",
        searchStatus: "all",
        searchType: "all",
```

````
        sort: "latest",
      };
    }
    ```

    #### Refactor Get All Jobs

    ```js
    const getJobs = async () => {
      // will add page later
      const { search, searchStatus, searchType, sort } = state;
      let url = `/jobs?status=${searchStatus}&jobType=${searchType}&sort=${sort}`;
      if (search) {
        url = url + `&search=${search}`;
      }
      dispatch({ type: GET_JOBS_BEGIN });
      try {
        const { data } = await authFetch(url);
        const { jobs, totalJobs, numOfPages } = data;
        dispatch({
          type: GET_JOBS_SUCCESS,
          payload: {
            jobs,
            totalJobs,
            numOfPages,
          },
        });
      } catch (error) {
        // logoutUser()
      }
      clearAlert();
    };
    ```

    ```js
    JobsContainer.js

    const JobsContainer = () => {
      const {
        getJobs,
        jobs,
        isLoading,
        page,
        totalJobs,
        search,
        searchStatus,
        searchType,
        sort,

      } = useAppContext()
      useEffect(() => {
        getJobs()
      }, [ search, searchStatus, searchType, sort])

    ```

    #### Limit and Skip

    ```js
    jobsController.js;

````

````
    const getAllJobs = async (req, res) => {
      const { search, status, jobType, sort } = req.query;
      const queryObject = {
        createdBy: req.user.userId,
      };
      if (search) {
        queryObject.position = { $regex: search, $options: "i" };
      }
      if (status !== "all") {
        queryObject.status = status;
      }
      if (jobType !== "all") {
        queryObject.jobType = jobType;
      }
      let result = Job.find(queryObject);

      if (sort === "latest") {
        result = result.sort("-createdAt");
      }
      if (sort === "oldest") {
        result = result.sort("createdAt");
      }
      if (sort === "a-z") {
        result = result.sort("position");
      }
      if (sort === "z-a") {
        result = result.sort("-position");
      }

      const totalJobs = await result;

      // setup pagination
      const limit = 10;
      const skip = 1;

      result = result.skip(skip).limit(limit);
      // 23
      // 4 7 7 7 2
      const jobs = await result;
      res
        .status(StatusCodes.OK)
        .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
    };
    ```

    #### Page and Limit

    ```js
    jobsController.js;

    const getAllJobs = async (req, res) => {
      const { search, status, jobType, sort } = req.query;
      const queryObject = {
        createdBy: req.user.userId,
      };
      if (search) {
        queryObject.position = { $regex: search, $options: "i" };
      }
      if (status !== "all") {
        queryObject.status = status;
      }
````

```
    if (jobType !== "all") {
      queryObject.jobType = jobType;
    }
    let result = Job.find(queryObject);

    if (sort === "latest") {
      result = result.sort("-createdAt");
    }
    if (sort === "oldest") {
      result = result.sort("createdAt");
    }
    if (sort === "a-z") {
      result = result.sort("position");
    }
    if (sort === "z-a") {
      result = result.sort("-position");
    }

    // setup pagination
    const page = Number(req.query.page) || 1;
    const limit = Number(req.query.limit) || 10;
    const skip = (page - 1) * limit; //10
    result = result.skip(skip).limit(limit);
    // 75
    // 10 10 10 10 10 10 10 5
    const jobs = await result;
    res
      .status(StatusCodes.OK)
      .json({ jobs, totalJobs: jobs.length, numOfPages: 1 });
};
```

#### Total Jobs and Number Of Pages

```js
jobsController.js;

const getAllJobs = async (req, res) => {
  const { search, status, jobType, sort } = req.query;
  const queryObject = {
    createdBy: req.user.userId,
  };
  if (search) {
    queryObject.position = { $regex: search, $options: "i" };
  }
  if (status !== "all") {
    queryObject.status = status;
  }
  if (jobType !== "all") {
    queryObject.jobType = jobType;
  }
  let result = Job.find(queryObject);

  if (sort === "latest") {
    result = result.sort("-createdAt");
  }
  if (sort === "oldest") {
    result = result.sort("createdAt");
  }
  if (sort === "a-z") {
    result = result.sort("position");
```

```
    }
    if (sort === "z-a") {
      result = result.sort("-position");
    }

    // setup pagination
    const page = Number(req.query.page) || 1;
    const limit = Number(req.query.limit) || 10;
    const skip = (page - 1) * limit;

    result = result.skip(skip).limit(limit);

    const jobs = await result;

    const totalJobs = await Job.countDocuments(queryObject);
    const numOfPages = Math.ceil(totalJobs / limit);

    res.status(StatusCodes.OK).json({ jobs, totalJobs, numOfPages });
};
```

#### PageBtnContainer Setup

- PageBtnContainer.js

```js
JobsContainer.js;

import PageBtnContainer from "./PageBtnContainer";

const { numOfPages } = useAppContext();

return (
  <Wrapper>
    <h5>
      {totalJobs} job{jobs.length > 1 && "s"} found
    </h5>
    <div className="jobs">
      {jobs.map((job) => {
        return <Job key={job._id} {...job} />;
      })}
    </div>
    {numOfPages > 1 && <PageBtnContainer />}
  </Wrapper>
);
```

#### PageBtnContainer - Structure

```js
PageBtnContainer.js;

import { useAppContext } from "../context/appContext";
import { HiChevronDoubleLeft, HiChevronDoubleRight } from "react-icons/hi";
import Wrapper from "../assets/wrappers/PageBtnContainer";

const PageButtonContainer = () => {
  const { numOfPages, page } = useAppContext();

  const prevPage = () => {
    console.log("prev page");
```

```
    };
    const nextPage = () => {
      console.log("next page");
    };

    return (
      <Wrapper>
        <button className="prev-btn" onClick={prevPage}>
          <HiChevronDoubleLeft />
          prev
        </button>

        <div className="btn-container">buttons</div>

        <button className="next-btn" onClick={nextPage}>
          next
          <HiChevronDoubleRight />
        </button>
      </Wrapper>
    );
  };

  export default PageButtonContainer;
  ```

  #### Button Container

  - [Array.from] (https://youtu.be/zg1Bv4xubwo)

  ```js
  PageBtnContainer.js;

  const pages = Array.from({ length: numOfPages }, (_, index) => {
    return index + 1;
  });

  return (
    <div className="btn-container">
      {pages.map((pageNumber) => {
        return (
          <button
            type="button"
            className={pageNumber === page ? "pageBtn active" : "pageBtn"}
            key={pageNumber}
            onClick={() => console.log(page)}
          >
            {pageNumber}
          </button>
        );
      })}
    </div>
  );
  ```

  #### Change Page

  ```js
  actions.js;
  export const CHANGE_PAGE = "CHANGE_PAGE";
  ```
```
```

````
```js
appContext.js
const changePage = (page) => {
  dispatch({ type: CHANGE_PAGE, payload: { page } })
}
value={{changePage}}
```

```js
reducer.js;

if (action.type === CHANGE_PAGE) {
  return { ...state, page: action.payload.page };
}
```

```js
PageBtnContainer.js;

const { changePage } = useAppContext();
return (
  <button
    type="button"
    className={pageNumber === page ? "pageBtn active" : "pageBtn"}
    key={pageNumber}
    onClick={() => changePage(pageNumber)}
  >
    {pageNumber}
  </button>
);
```

#### Prev and Next Buttons

```js
PageBtnContainer.js;
const prevPage = () => {
  let newPage = page - 1;
  if (newPage < 1) {
    // newPage = 1
    // alternative
    newPage = numOfPages;
  }
  changePage(newPage);
};
const nextPage = () => {
  let newPage = page + 1;
  if (newPage > numOfPages) {
    // newPage = numOfPages
    // alternative
    newPage = 1;
  }
  changePage(newPage);
};
```

#### Trigger New Page

```js
appContext.js;

```
````

````
      const getJobs = async () => {
        const { page, search, searchStatus, searchType, sort } = state;

        let url =
    `/jobs?page=${page}&status=${searchStatus}&jobType=${searchType}&sort=${sort}`;
        // rest of the code
      };
      ```

      ```js
      JobsContainer.js;

      const { page } = useAppContext();
      useEffect(() => {
        getJobs();
      }, [page, search, searchStatus, searchType, sort]);
      ```

      ```js
      reducer.js;

      if (action.type === HANDLE_CHANGE) {
        // set back to first page

        return { ...state, page: 1, [action.payload.name]: action.payload.value };
      }
      ```

      #### Production Setup - Fix Warnings and logoutUser

      - getJobs,deleteJob,showStats - invoke logoutUser()
      - fix warnings

      ```sh
      // eslint-disable-next-line
      ```

      #### Production Setup - Build Front-End Application

      - create front-end production application

      ```js
      package.json
      "scripts": {
          "build-client": "cd client && npm run build",
          "server": "nodemon server.js --ignore client",
          "client": "cd client && npm run start",
          "start": "concurrently --kill-others-on-fail \"npm run server\" \"npm run client\""

        },

      ```

      ```js
      server.js;

      import { dirname } from "path";
      import { fileURLToPath } from "url";
      import path from "path";

      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,
    and url params.
5382.      - [express-mongo-sanitize](https://www.npmjs.com/package/express-mongo-sanitize)
5383.        Sanitizes user-supplied data to prevent MongoDB Operator Injection.
5384.      - [express-rate-limit](https://www.npmjs.com/package/express-rate-limit)
5385.        Basic rate-limiting middleware for Express.
5386.
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.
````

````
#### Alternative Search with Debounce

client/components/SearchContainer.js

```js

import { useState, useMemo } from 'react';
const SearchContainer = () => {
  const [localSearch, setLocalSearch] = useState('');
  const {
    ....
  } = useAppContext();
  const handleSearch = (e) => {
    handleChange({ name: e.target.name, value: e.target.value });
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    clearFilters();
  };
  const debounce = () => {
    let timeoutID;
    return (e) => {
      setLocalSearch(e.target.value);
      clearTimeout(timeoutID);
      timeoutID = setTimeout(() => {
        handleChange({ name: e.target.name, value: e.target.value });
      }, 1000);
    };
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    setLocalSearch('');
    clearFilters();
  };

  const optimizedDebounce = useMemo(() => debounce(), []);
  return (
    <Wrapper>
      <form className='form'>
        <h4>search form</h4>
        <div className='form-center'>
          {/* search position */}

          <FormRow
            type='text'
            name='search'
            value={localSearch}
            handleChange={optimizedDebounce}
          />
         ........
        </div>
      </form>
    </Wrapper>
  );
};

export default SearchContainer;
```

#### Test User - Initial Setup
````

````
- create new user (test user)
- populate DB with jobs
- create a login button

client/pages/Register.js

```js
const Register = () => {
  return (
    <Wrapper className="full-page">
      <form className="form" onSubmit={onSubmit}>
        <button type="submit" className="btn btn-block" disabled={isLoading}>
          submit
        </button>
        <button
          type="button"
          className="btn btn-block btn-hipster"
          disabled={isLoading}
          onClick={() => {
            setupUser({
              currentUser: { email: "testUser@test.com", password: "secret" },
              endPoint: "login",
              alertText: "Login Successful! Redirecting...",
            });
          }}
        >
          {isLoading ? "loading..." : "demo app"}
        </button>
      </form>
    </Wrapper>
  );
};
export default Register;
```

#### Test User - Restrict Access (server)

- check for test user in auth middleware
- create new property on user object (true/false)
- create new middleware (testUser)
- check for test user, if true send back BadRequest Error
- add testUser middleware in front of routes you want to restrict access to

middleware/auth.js

```js
import jwt from "jsonwebtoken";
import { UnAuthenticatedError } from "../errors/index.js";

UnAuthenticatedError;
const auth = async (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith("Bearer")) {
    throw new UnAuthenticatedError("Authentication Invalid");
  }
  const token = authHeader.split(" ")[1];
  try {
    const payload = jwt.verify(token, process.env.JWT_SECRET);
    // TEST USER
    const testUser = payload.userId === "testUserID";
````
````

````
      req.user = { userId: payload.userId, testUser };
      // TEST USER
      next();
    } catch (error) {
      throw new UnAuthenticatedError("Authentication Invalid");
    }
  };

  export default auth;
  ```

  middleware/testUser

  ```js
  import { BadRequestError } from "../errors/index.js";

  const testUser = (req, res, next) => {
    if (req.user.testUser) {
      throw new BadRequestError("Test User. Read Only!");
    }
    next();
  };

  export default testUser;
  ```

  routes/jobsRoutes

  ```js
  import express from "express";
  const router = express.Router();

  import {
    createJob,
    deleteJob,
    getAllJobs,
    updateJob,
    showStats,
  } from "../controllers/jobsController.js";

  import testUser from "../middleware/testUser.js";

  router.route("/").post(testUser, createJob).get(getAllJobs);
  // remember about :id
  router.route("/stats").get(showStats);
  router.route("/:id").delete(testUser, deleteJob).patch(testUser, updateJob);

  export default router;
  ```

  routes/authRoutes

  ```js
  import express from "express";
  const router = express.Router();

  import rateLimiter from "express-rate-limit";
  const apiLimiter = rateLimiter({
    windowMs: 15 * 60 * 1000, // 15 minutes
    max: 10,
    message: "Too many requests from this IP, please try again after 15 minutes",
````

````
});

import { register, login, updateUser } from "../controllers/authController.js";
import authenticateUser from "../middleware/auth.js";
import testUser from "../middleware/testUser.js";
router.route("/register").post(apiLimiter, register);
router.route("/login").post(apiLimiter, login);
router.route("/updateUser").patch(authenticateUser, testUser, updateUser);

export default router;
```

#### Store JWT in Cookie

- BE PREPARED TO REFACTOR CODE !!!
- PLEASE DON'T RUSH THROUGH THESES VIDEOS
- CHECK FEW TIMES BEFORE REMOVING/ADDING CODE

#### Attach Cookies to Login response

controllers/authController.js

```js
// login controller

const token = user.createJWT();

const oneDay = 1000 * 60 * 60 * 24;

res.cookie("token", token, {
  httpOnly: true,
  expires: new Date(Date.now() + oneDay),
  secure: process.env.NODE_ENV === "production",
});
```

#### Setup Function in Utils

- create attachCookies.js

```js
const attachCookie = ({ res, token }) => {
  const oneDay = 1000 * 60 * 60 * 24;

  res.cookie("token", token, {
    httpOnly: true,
    expires: new Date(Date.now() + oneDay),
    secure: process.env.NODE_ENV === "production",
  });
};

export default attachCookie;
```

- import in authController.js
- invoke in register/login/updateUser

```js
import attachCookie from "../utils/attachCookie.js";

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.
````

````
```js
const token = localStorage.getItem("token");
const user = localStorage.getItem("user");
const userLocation = localStorage.getItem("location");
```

- fix initial state

```js
const initialState = {
  // remove token all together
  user: null,
  userLocation: "",
  jobLocation: "",
};
```

- remove request interceptor

```js
authFetch.interceptors.request.use(
  (config) => {
    config.headers.common["Authorization"] = `Bearer ${state.token}`;
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);
```

- remove both addToLocalStorage and removeFromLocalStorage functions
- remove from setupUser and updateUser (token and local storage functions)
- remove from the reducer token (COMMAND + F)

```js
const logoutUser = async () => {
  dispatch({ type: LOGOUT_USER });
  // remove local storage code
};
```

#### Test Expiration

```js
expires: new Date(Date.now() + 5000),
```

#### GET Current User Route

controllers/authController.js

```js
const getCurrentUser = async (req, res) => {
  const user = await User.findOne({ _id: req.user.userId });
  res.status(StatusCodes.OK).json({ user, location: user.location });
};

export { register, login, updateUser, getCurrentUser };
```
````

routes/authRoutes.js

```js
import {
  register,
  login,
  updateUser,
  getCurrentUser,
} from "../controllers/authController.js";

router.route("/register").post(apiLimiter, register);
router.route("/login").post(apiLimiter, login);
router.route("/updateUser").patch(authenticateUser, testUser, updateUser);
router.route("/getCurrentUser").get(authenticateUser, getCurrentUser);
```

#### GET Current User - Front-End

actions.js

```js
export const GET_CURRENT_USER_BEGIN = "GET_CURRENT_USER_BEGIN";
export const GET_CURRENT_USER_SUCCESS = "GET_CURRENT_USER_SUCCESS";
```

- setup imports (appContext and reducer)

#### GET Current User Request

- first set the state value (default TRUE !!!)
  appContext.js

```js
const initialState = {
  userLoading: true,
};

const getCurrentUser = async () => {
  dispatch({ type: GET_CURRENT_USER_BEGIN });
  try {
    const { data } = await authFetch("/auth/getCurrentUser");
    const { user, location } = data;

    dispatch({
      type: GET_CURRENT_USER_SUCCESS,
      payload: { user, location },
    });
  } catch (error) {
    if (error.response.status === 401) return;
    logoutUser();
  }
};
useEffect(() => {
  getCurrentUser();
}, []);
```

reducer.js

```js
if (action.type === GET_CURRENT_USER_BEGIN) {
```

```
      return { ...state, userLoading: true, showAlert: false };
    }
    if (action.type === GET_CURRENT_USER_SUCCESS) {
      return {
        ...state,
        userLoading: false,
        user: action.payload.user,
        userLocation: action.payload.location,
        jobLocation: action.payload.location,
      };
    }
    ```

    ```js
    if (action.type === LOGOUT_USER) {
      return {
        ...initialState,
        userLoading: false,
      };
    }
    ```

    #### Protected Route FIX

    ```js
    import Loading from "../components/Loading";

    const ProtectedRoute = ({ children }) => {
      const { user, userLoading } = useAppContext();

      if (userLoading) return <Loading />;

      if (!user) {
        return <Navigate to="/landing" />;
      }
      return children;
    };

    export default ProtectedRoute;
    ```

    #### Landing Page

    ```js
    import { Navigate } from "react-router-dom";
    import { useAppContext } from "../context/appContext";

    const Landing = () => {
      const { user } = useAppContext();
      return (
        <React.Fragment>
          {user && <Navigate to="/" />}
          <Wrapper>// rest of the code.....</Wrapper>
        </React.Fragment>
      );
    };

    export default Landing;
    ```

    #### 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
```