Jobster

Project in Action - Jobster

React Course

My React Course

Support

Find the App Useful? You can always buy me a coffee

Run The App Locally

```
npm run install && npm start
```

- visit url http://localhost:3000/

Setup React App

```
npx create-react-app myApp
```

```
npx create-react-app@latest myApp
```

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

1) Spring Cleaning

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

2) Title and Favicon

- change title in public/index.html
- replace favicon.ico in public
- resource Generate Favicons

3) 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

```
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
- Repo - Default Starter Repo

4) Landing Page - Setup

- 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

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

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

5) Landing Page - Structure

- Landing.js

```
const Landing = () => {
  return (
    <main>
      <nav>
        <img src={logo} alt='jobster logo' className='logo' />
      </nav>
      <div className='container page'>
        {/* info */}
        <div className='info'>
          <h1>
            job <span>tracking</span> app
          </h1>
          <p>some text</p>
          <button className='btn btn-hero'>Login/Register</button>
        </div>
        <img src={main} alt='job hunt' className='img main-img' />
      </div>
    </main>
  );
};

export default Landing;
```

6) Styled Components - Basic Setup

- CSS in JS
- Styled Components
- have logic and styles in component
- no name collisions
- apply javascript logic
- Styled Components Docs
- Styled Components Course

```
npm install styled-components
```

```
import styled from 'styled-components';

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

- element can be any html element (div,button,section, etc)
- no name collisions, since unique class
- vscode-styled-components extension
- colors and bugs
- style entire react component

7) Styled Components - Wrap Component

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

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

8) Wrappers

- assets/wrappers
- only responsible for styling

9) Logo and Images

- logo built in Figma
- Cool Images

11) Logo Component

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

Logo.js

```
import logo from '../assets/images/logo.svg';

const Logo = () => {
  return <img src={logo} alt='jobify' className='logo' />;
};

export default Logo;
```

12) 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

13) React Router 6

- Please Reference React Router 6 Section
- React Router Docs

```
npm install react-router-dom@6
```

- import three components from router

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import { Error, Landing, Register, Dashboard } from './pages';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path='/' element={<Dashboard />} />
        <Route path='landing' element={<Landing />} />
        <Route path='register' element={<Register />} />
        <Route path='*' element={<Error />} />
      </Routes>
    </BrowserRouter>
  );
}
```

- go to Landing.js

```
import { Link } from 'react-router-dom';

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

14) Error Page

```
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>
);
```

15) 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

16) Register Page - Setup

```
import { useState, useEffect } from 'react';
import { Logo } from '../components';
import Wrapper from '../assets/wrappers/RegisterPage';
// redux toolkit and useNavigate later

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

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

  // redux toolkit 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>
  );
}
```

17) Switch To React 18

- index.js

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import 'normalize.css';
import './index.css';
import App from './App';

const container = document.getElementById('root');
const root = createRoot(container);
root.render(<App tab='home' />);
```

18) FormRow Component

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

```jsx
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;
```

19) Toggle Member

```
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>
);
```

20) Handle Change and Empty Values

Dynamic Object Keys

Register.js

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

const onSubmit = (e) => {
  e.preventDefault();
  const { name, email, password, isMember } = values;
  if (!email || !password || (!isMember && !name)) {
    consol.log('Please Fill Out All Fields');
    return;
  }
};
```

21) React Toastify

React Toastify

```
npm install --save react-toastify
```

App.js

```
import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';


return </Routes>
<ToastContainer />
<BrowserRouter>
```

Register.js

```
import { toast } from 'react-toastify';

if (!email || !password || (!isMember && !name)) {
  toast.error('Please Fill Out All Fields');
  return;
}
```

- modifications

position

index.css

```
.Toastify__toast {
  text-transform: capitalize;
}
```

22) User Slice - Setup

- features/user/userSlice.js

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import { toast } from 'react-toastify';

const initialState = {
  isLoading: false,
  user: null,
};

const userSlice = createSlice({
  name: 'user',
  initialState,
});

export default userSlice.reducer;
```

- create store.js

```
import { configureStore } from '@reduxjs/toolkit';

import userSlice from './features/user/userSlice';

export const store = configureStore({
  reducer: {
    user: userSlice,
  },
});
```

- index.js

```
import { store } from './store';
import { Provider } from 'react-redux';

root.render(
  <Provider store={store}>
    <App tab='home' />
  </Provider>
);


npm install @reduxjs/toolkit react-redux
```

23) RegisterUser, LoginUser - Placeholders

- userSlice.js

```
export const registerUser = createAsyncThunk(
  'user/registerUser',
  async (user, thunkAPI) => {
    console.log(`Register User : ${user}`)
);
export const loginUser = createAsyncThunk(
  'user/loginUser',
  async (user, thunkAPI) => {
    console.log(`Login User : ${user}`)
});
```

- Register.js

```
import { useSelector, useDispatch } from 'react-redux';
import { loginUser, registerUser } from '../features/user/userSlice';


const Register = () => {
  const dispatch = useDispatch();
  const { isLoading, user } = useSelector((store) => store.user);


const onSubmit = (e) => {
    e.preventDefault();
    const { name, email, password, isMember } = values;
    if (!email || !password || (!isMember && !name)) {
      toast.error('Please Fill Out All Fields');
      return;
    }
    if (isMember) {
      dispatch(loginUser({ email: email, password: password }));
      return;
    }
    dispatch(registerUser({ name, email, password }));
  };
```

24) HTTP Methods

- GET - get resources from the server
- POST - submit resource to the server
- PUT/PATCH - modify resource on the server
- DELETE - delete resource form the server

```
// GET
axios.get(url, options);
// POST
axios.post(url, resource, options);
// PATCH
axios.patch(url, resource, options);
// DELETE
axios.delete(url, options);


npm install axios
```

25) API

- Root URL
- https://jobify-prod.herokuapp.com/api/v1/toolkit

- NODE COURSE

Register USER

- https://jobify-prod.herokuapp.com/api/v1/toolkit/auth/register
- POST /auth/register
- {name:'john',email:'john@gmail.com',password:'secret'}
- sends back the user object with token

Register USER - TESTING()

- POST /auth/testingRegister
- {name:'john',email:'john@gmail.com',password:'secret'}
- sends back the user object with token

Login USER

- POST /auth/login
- {email:'john@gmail.com',password:'secret'}
- sends back the user object with token

Update USER

- PATCH /auth/updateUser
- { email:'john@gmail.com', name:'john', lastName:'smith', location:'my location' }
- sends back the user object with token

26) Custom Axios Instance

- utils/axios.js

```
import axios from 'axios';

const customFetch = axios.create({
  baseURL: 'https://jobify-prod.herokuapp.com/api/v1/toolkit',
});

export default customFetch;
```

userSlice.js

```
import customFetch from '../../utils/axios';

export const registerUser = createAsyncThunk(
  'user/registerUser',
  async (user, thunkAPI) => {
    try {
      const resp = await customFetch.post('/auth/testingRegister', user);
      console.log(resp);
    } catch (error) {
      console.log(error.response);
    }
  }
);
```

27) Register User

userSlice.js

```
export const registerUser = createAsyncThunk(
  'user/registerUser',
  async (user, thunkAPI) => {
    try {
      const resp = await customFetch.post('/auth/register', user);
      return resp.data;
    } catch (error) {
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
)
   extraReducers: {
    [registerUser.pending]: (state) => {
      state.isLoading = true;
    },
    [registerUser.fulfilled]: (state, { payload }) => {
      const { user } = payload;
      state.isLoading = false;
      state.user = user;
      toast.success(`Hello There ${user.name}`);
    },
    [registerUser.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    }
  }
```

Extra Reducers "Builder Callback" Notation

```
extraReducers: (builder) => {
    builder
      .addCase(registerUser.pending, (state) => {
        state.isLoading = true;
      })
      .addCase(registerUser.fulfilled, (state, { payload }) => {
        const { user } = payload;
        state.isLoading = false;
        state.user = user;
        addUserToLocalStorage(user);
        toast.success(`Hello There ${user.name}`);
      })
      .addCase(registerUser.rejected, (state, { payload }) => {
        state.isLoading = false;
        toast.error(payload);
      })
      .addCase(loginUser.pending, (state) => {
        state.isLoading = true;
      })
      .addCase(loginUser.fulfilled, (state, { payload }) => {
        const { user } = payload;
        state.isLoading = false;
        state.user = user;
        addUserToLocalStorage(user);

        toast.success(`Welcome Back ${user.name}`);
      })
      .addCase(loginUser.rejected, (state, { payload }) => {
        state.isLoading = false;
        toast.error(payload);
      })
      .addCase(updateUser.pending, (state) => {
        state.isLoading = true;
      })
      .addCase(updateUser.fulfilled, (state, { payload }) => {
        const { user } = payload;
        state.isLoading = false;
        state.user = user;
        addUserToLocalStorage(user);

        toast.success(`User Updated!`);
      })
      .addCase(updateUser.rejected, (state, { payload }) => {
        state.isLoading = false;
        toast.error(payload);
      })
      .addCase(clearStore.rejected, () => {
        toast.error('There was an error..');
```

```
      });
  },
```

28) Login User

userSlice.js

```
export const loginUser = createAsyncThunk(
  'user/loginUser',
  async (user, thunkAPI) => {
    try {
      const resp = await customFetch.post('/auth/login', user);
      return resp.data;
    } catch (error) {
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }

   extraReducers: {
    [loginUser.pending]: (state) => {
      state.isLoading = true;
    },
    [loginUser.fulfilled]: (state, { payload }) => {
      const { user } = payload;
      state.isLoading = false;
      state.user = user;
      toast.success(`Welcome Back ${user.name}`);
    },
    [loginUser.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    }
  }

);
```

29) LocalStorage

- utils/localStorage.js

```
export const addUserToLocalStorage = (user) => {
  localStorage.setItem('user', JSON.stringify(user));
};

export const removeUserFromLocalStorage = () => {
  localStorage.removeItem('user');
};

export const getUserFromLocalStorage = () => {
  const result = localStorage.getItem('user');
  const user = result ? JSON.parse(result) : null;
  return user;
};
```

- invoke getUserFromLocalStorage when app loads (set it equal to user)

```
const initialState = {
  isLoading: false,
  user: getUserFromLocalStorage(),
};
```

```
[registerUser.fulfilled]: (state, { payload }) => {
      const { user } = payload;
      state.isLoading = false;
      state.user = user;
      addUserToLocalStorage(user);
      toast.success(`Hello There ${user.name}`);
    },

[loginUser.fulfilled]: (state, { payload }) => {
      const { user } = payload;
      state.isLoading = false;
      state.user = user;
      addUserToLocalStorage(user);
      toast.success(`Welcome Back ${user.name}`);
    },
```

30) Programmatically Navigate To Dashboard

Register.js

```
import { useNavigate } from 'react-router-dom';

const Register = () => {
  const navigate = useNavigate();

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

31) Setup Dashboard Pages

- remove Dashboard.js
- create Dashboard Folder
- create Stats, Profile, AddJob, AllJobs, SharedLayout,
- create index.js and setup import/export

App.js

```
import {
  AllJobs,
  Profile,
  SharedLayout,
  Stats,
  AddJob,
} from './pages/dashboard';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path='/' element={<SharedLayout />}>
          <Route index element={<Stats />} />
          <Route path='all-jobs' element={<AllJobs />} />
          <Route path='add-job' element={<AddJob />} />
          <Route path='profile' element={<Profile />} />
        </Route>
        <Route path='register' element={<Register />} />
        <Route path='landing' element={<Landing />} />
        <Route path='*' element={<Error />} />
      </Routes>
      <ToastContainer position='top-center' />
    </BrowserRouter>
  );
}
```

32) Navbar, SmallSidebar, BigSidebar

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

SharedLayout.js;

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

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

export default SharedLayout;
```

33) Shared Layout CSS

- import Wrappers in BigSidebar,SmallSidebar and Navbar

34) React Icons

React Icons

```
npm install react-icons
```

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
```

35) Navbar Structure

Navbar.js;

```jsx
import Wrapper from '../assets/wrappers/Navbar';
import { FaAlignLeft, FaUserCircle, FaCaretDown } from 'react-icons/fa';
import Logo from './Logo';
import { useState } from 'react';

import { useDispatch, useSelector } from 'react-redux';

const Navbar = () => {
  const { user } = useSelector((store) => store.user);
  const dispatch = useDispatch();

  return (
    <Wrapper>
      <div className='nav-center'>
        <button
          type='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
            type='button'
            className='btn'
            onClick={() => console.log('toggle logout dropdown')}
          >
            <FaUserCircle />
            {user?.name}
            <FaCaretDown />
          </button>
          <div className='dropdown show-dropdown'>
            <button
              type='button'
              className='dropdown-btn'
              onClick={() => {
                console.log('logout user');
              }}
            >
              logout
            </button>
          </div>
        </div>
      </div>
    </Wrapper>
  );
```

```
};

export default Navbar;
```

36) Toggle Sidebar

userSlice.js

```
const initialState = {
  isLoading: false,
  isSidebarOpen: false,
  user: getUserFromLocalStorage(),
};

reducers: {
    toggleSidebar: (state) => {
      state.isSidebarOpen = !state.isSidebarOpen;
    },
  },

export const { toggleSidebar } = userSlice.actions;
```

Navbar.js

```
import { toggleSidebar } from '../features/user/userSlice';

const toggle = () => {
  dispatch(toggleSidebar());
};

<button type='button' className='toggle-btn' onClick={toggle}>
  <FaAlignLeft />
</button>;
```

37) Toggle Dropdown

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={() => console.log('logout user')} className='dropdown-btn'>
      logout
    </button>
  </div>
</div>
```

38) Logout User

userSlice.js

```
reducers: {
    logoutUser: (state) => {
      state.user = null;
      state.isSidebarOpen = false;
      removeUserFromLocalStorage();
    },
    toggleSidebar: (state) => {
      state.isSidebarOpen = !state.isSidebarOpen;
    },
  },

export const { logoutUser, toggleSidebar } = userSlice.actions;
```

Navbar.js

```
import { toggleSidebar, logoutUser } from '../features/user/userSlice';

<div className={showLogout ? 'dropdown show-dropdown' : 'dropdown'}>
  <button
    type='button'
    className='dropdown-btn'
    onClick={() => {
      dispatch(logoutUser());
    }}
  >
    logout
  </button>
</div>;
```

39) Restrict Access

- pages/ProtectedRoute.js

```
import { Navigate } from 'react-router-dom';
import { useSelector } from 'react-redux';
const ProtectedRoute = ({ children }) => {
  const { user } = useSelector((store) => store.user);
  if (!user) {
    return <Navigate to='/landing' />;
  }
  return children;
};

export default ProtectedRoute;
```

App.js

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

40) Small Sidebar - Setup

SmallSidebar.js;

```
import Wrapper from '../assets/wrappers/SmallSidebar';
import { FaTimes } from 'react-icons/fa';
import { NavLink } from 'react-router-dom';
import Logo from './Logo';
import { useSelector, useDispatch } from 'react-redux';

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;
```

41) Small Sidebar - Toggle

SmallSidebar.js;

```
import { toggleSidebar } from '../features/user/userSlice';


const { isSidebarOpen } = useSelector((store) => store.user);
  const dispatch = useDispatch();
  const toggle = () => {
    dispatch(toggleSidebar());
  };

return (
  <div className={isSidebarOpen ? 'sidebar-container show-sidebar' : 'sidebar-container'}>
    <div className='content'>
        <button type='button' className='close-btn' onClick={toggle}>
          <FaTimes />
        </button>

);
```

42) Setup Links

- create utils/links.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;
```

43) Small Sidebar - Nav Links

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={toggle}
        >
          <span className='icon'>{icon}</span>
          {text}
        </NavLink>
      );
    })}
  </div>
);
```

REACT ROUTER UPDATE !!!

- Stack Overflow

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


end
>
```

44) Nav Links Component

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

```jsx
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;
```

SmallSidebar.js;

```jsx
import NavLinks from './NavLinks';

return <NavLinks toggleSidebar={toggle} />;
```

45) Big Sidebar

```
import NavLinks from './NavLinks';
import Logo from '../components/Logo';
import Wrapper from '../assets/wrappers/BigSidebar';
import { useSelector } from 'react-redux';

const BigSidebar = () => {
  const { isSidebarOpen } = useSelector((store) => store.user);
  return (
    <Wrapper>
      <div
        className={
          isSidebarOpen
            ? 'sidebar-container '
            : 'sidebar-container show-sidebar'
        }
      >
        <div className='content'>
          <header>
            <Logo />
          </header>
          <NavLinks />
        </div>
      </div>
    </Wrapper>
  );
};

export default BigSidebar;
```

46) Profile Page - Structure

Profile.js

```jsx
import { useState } from 'react';
import { FormRow } from '../../components';
import Wrapper from '../../assets/wrappers/DashboardFormPage';
import { useDispatch, useSelector } from 'react-redux';
import { toast } from 'react-toastify';


const Profile = () => {
  const { isLoading, user } = useSelector((store) => store.user);
  const dispatch = useDispatch();

const [userData,setUserData] = useState({
  name:user?.name ||''
  email:user?.email ||''
  lastName:user?.lastName ||''
  location:user?.location ||''
})

  const handleSubmit = (e) => {
    e.preventDefault();
        const { name, email, lastName, location } = userData;

    if (!name || !email || !lastName || !location) {
      toast.error('Please Fill Out All Fields');
      return;
    }
  };
const handleChange = (e) =>{
  const name = e.target.name
  const value = e.target.value
  setUserData({...userData,[name]:value})
}
  return (
    <Wrapper>
      <form className='form' onSubmit={handleSubmit}>
        <h3>profile</h3>

        <div className='form-center'>
          <FormRow
            type='text'
            name='name'
            value={userData.name}
            handleChange={handleChange}
          />
          <FormRow
            type='text'
            labelText='last name'
            name='lastName'
            value={userData.lastName}
            handleChange={handleChange}
```

```
            />
            <FormRow
              type='email'
              name='email'
              value={userData.email}
              handleChange={handleChange}
            />
            <FormRow
              type='text'
              name='location'
              value={userData.location}
              handleChange={handleChange}
            />
            <button className='btn btn-block' type='submit' disabled={isLoading}>
              {isLoading ? 'Please Wait...' : 'save changes'}
            </button>
          </div>
        </form>
      </Wrapper>
    );
  };

  export default Profile;
```

47) Update User - Complete

Update USER

- PATCH /auth/updateUser
- { email:'john@gmail.com', name:'john', lastName:'smith', location:'my location' }
- authorization header : 'Bearer token'
- sends back the user object with token

userSlice.js

```
export const updateUser = createAsyncThunk(
  'user/updateUser',
  async (user, thunkAPI) => {
    try {
      const resp = await customFetch.patch('/auth/updateUser', user, {
        headers: {
          authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
        },
      });
      return resp.data;
    } catch (error) {
      console.log(error.response);
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);
// extra reducers
    [updateUser.pending]: (state) => {
      state.isLoading = true;
    },
    [updateUser.fulfilled]: (state, { payload }) => {
      const { user } = payload;
      state.isLoading = false;
      state.user = user;

      addUserToLocalStorage(user);
      toast.success('User Updated');
    },
    [updateUser.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    },
```

Profile.js

```
import { updateUser } from '../../features/user/userSlice';

const handleSubmit = (e) => {
  e.preventDefault();
  const { name, email, lastName, location } = userData;

  if (!name || !email || !lastName || !location) {
    toast.error('Please Fill Out All Fields');
    return;
  }
  dispatch(updateUser({ name, email, lastName, location }));
};
```

48) Unauthorized - Logout User

userSlice.js

```
export const updateUser = createAsyncThunk(
  'user/updateUser',
  async (user, thunkAPI) => {
    try {
      const resp = await customFetch.patch('/auth/updateUser', user, {
        headers: {
          // authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
          authorization: `Bearer `,
        },
      });

      return resp.data;
    } catch (error) {
      // console.log(error.response);
      if (error.response.status === 401) {
        thunkAPI.dispatch(logoutUser());
        return thunkAPI.rejectWithValue('Unauthorized! Logging Out...');
      }
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);

// logoutUser
logoutUser: (state) => {
      state.user = null;
      state.isSidebarOpen = false;
      toast.success('Logout Successful!');
      removeUserFromLocalStorage();
    },
```

49) Refactor Async Functionality

- features/user/userThunk.js

```
import customFetch from '../../utils/axios';

import { logoutUser } from './userSlice';

export const registerUserThunk = async (url, user, thunkAPI) => {
  try {
    const resp = await customFetch.post(url, user);
    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};

export const loginUserThunk = async (url, user, thunkAPI) => {
  try {
    const resp = await customFetch.post(url, user);
    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};

export const updateUserThunk = async (url, user, thunkAPI) => {
  try {
    const resp = await customFetch.patch(url, user, {
      headers: {
        authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
      },
    });
    return resp.data;
  } catch (error) {
    // console.log(error.response);
    if (error.response.status === 401) {
      thunkAPI.dispatch(logoutUser());
      return thunkAPI.rejectWithValue('Unauthorized! Logging Out...');
    }
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};
```

userSlice.js

```
import {
  loginUserThunk,
  registerUserThunk,
  updateUserThunk,
} from './userThunk';

export const registerUser = createAsyncThunk(
  'user/registerUser',
  async (user, thunkAPI) => {
    return registerUserThunk('/auth/register', user, thunkAPI);
  }
);

export const loginUser = createAsyncThunk(
  'user/loginUser',
  async (user, thunkAPI) => {
    return loginUserThunk('/auth/login', user, thunkAPI);
  }
);
export const updateUser = createAsyncThunk(
  'user/updateUser',
  async (user, thunkAPI) => {
    return updateUserThunk('/auth/updateUser', user, thunkAPI);
  }
);
```

50) Job Slice

- features/job/jobSlice.js

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import { toast } from 'react-toastify';
import customFetch from '../../utils/axios';
import { getUserFromLocalStorage } from '../../utils/localStorage';

const initialState = {
  isLoading: false,
  position: '',
  company: '',
  jobLocation: '',
  jobTypeOptions: ['full-time', 'part-time', 'remote', 'internship'],
  jobType: 'full-time',
  statusOptions: ['interview', 'declined', 'pending'],
  status: 'pending',
  isEditing: false,
  editJobId: '',
};

const jobSlice = createSlice({
  name: 'job',
  initialState,
});

export default jobSlice.reducer;
```

store.js

```
import jobSlice from './features/job/jobSlice';

export const store = configureStore({
  reducer: {
    user: userSlice,
    job: jobSlice,
  },
});
```

51) Add Job - Structure

AddJob.js

```
import { FormRow } from '../../components';
import Wrapper from '../../assets/wrappers/DashboardFormPage';
import { useSelector, useDispatch } from 'react-redux';
import { toast } from 'react-toastify';

const AddJob = () => {
  const {
    isLoading,
    position,
    company,
    jobLocation,
    jobType,
    jobTypeOptions,
    status,
    statusOptions,
    isEditing,
    editJobId,
  } = useSelector((store) => store.job);
  const dispatch = useDispatch();

  const handleSubmit = (e) => {
    e.preventDefault();

    if (!position || !company || !jobLocation) {
      toast.error('Please Fill Out All Fields');
      return;
    }
  };
  const handleJobInput = (e) => {
    const name = e.target.name;
    const value = e.target.value;
  };

  return (
    <Wrapper>
      <form className='form'>
        <h3>{isEditing ? 'edit job' : 'add job'}</h3>

        <div className='form-center'>
          {/* position */}
          <FormRow
            type='text'
            name='position'
            value={position}
            handleChange={handleJobInput}
          />
          {/* company */}
          <FormRow
            type='text'
            name='company'
```

```
            value={company}
            handleChange={handleJobInput}
          />
          {/* location */}
          <FormRow
            type='text'
            labelText='job location'
            name='jobLocation'
            value={jobLocation}
            handleChange={handleJobInput}
          />
          {/* job status */}

          {/* job type */}

          {/* btn container */}
          <div className='btn-container'>
            <button
              type='button'
              className='btn btn-block clear-btn'
              onClick={() => console.log('clear values')}
            >
              clear
            </button>
            <button
              type='submit'
              className='btn btn-block submit-btn'
              onClick={handleSubmit}
              disabled={isLoading}
            >
              submit
            </button>
          </div>
        </div>
      </form>
    </Wrapper>
  );
};

export default AddJob;
```

52) FormRowSelect

```
// job status

return (
  <div className='form-row'>
    <label htmlFor='status' className='form-label'>
      status
    </label>
    <select
      name='status'
      value={status}
      onChange={handleJobInput}
      className='form-select'
    >
      {statusOptions.map((itemValue, index) => {
        return (
          <option key={index} value={itemValue}>
            {itemValue}
          </option>
        );
      })}
    </select>
  </div>
);
```

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

export default FormRowSelect;
```

AddJob.js

```
  /* job status */
<FormRowSelect
  name='status'
  value={status}
  handleChange={handleJobInput}
  list={statusOptions}
/>

<FormRowSelect
  name='jobType'
  labelText='job type'
  value={jobType}
  handleChange={handleJobInput}
  list={jobTypeOptions}
/>
```

53) User Slice - HandleChange Reducer

jobSlice.js

```
    // reducers
    handleChange: (state, { payload: { name, value } }) => {
      state[name] = value;
    },

export const { handleChange } = jobSlice.actions;
```

AddJob.js

```
import { handleChange } from '../../features/job/jobSlice';

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

54) User Slice - ClearValues Reducer

```
    // reducers
    clearValues: () => {
      return {
        ...initialState
      };
      return initialState
    },

export const { handleChange, clearValues } = jobSlice.actions;
```

AddJob.js

```
import { clearValues, handleChange } from '../../features/job/jobSlice';

return (
  <button
    type='button'
    className='btn btn-block clear-btn'
    onClick={() => dispatch(clearValues())}
  >
    clear
  </button>
);
```

55) Create Job Request

- POST /jobs
- { position:'position', company:'company', jobLocation:'location', jobType:'full-time', status:'pending' }
- authorization header : 'Bearer token'
- sends back the job object

```
export const createJob = createAsyncThunk(
  'job/createJob',
  async (job, thunkAPI) => {
    try {
      const resp = await customFetch.post('/jobs', job, {
        headers: {
          authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
        },
      });
      thunkAPI.dispatch(clearValues());
      return resp.data;
    } catch (error) {
      // basic setup
      return thunkAPI.rejectWithValue(error.response.data.msg);
      // logout user
      if (error.response.status === 401) {
      thunkAPI.dispatch(logoutUser());
      return thunkAPI.rejectWithValue('Unauthorized! Logging Out...');
    }
    return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);

// extra reducers

extraReducers: {
    [createJob.pending]: (state) => {
      state.isLoading = true;
    },
    [createJob.fulfilled]: (state, action) => {
      state.isLoading = false;
      toast.success('Job Created');
    },
    [createJob.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    },
}
```

AddJob.js

```
import {
  clearValues,
  handleChange,
  createJob,
} from '../../features/job/jobSlice';

const handleSubmit = (e) => {
  e.preventDefault();

  if (!position || !company || !jobLocation) {
    toast.error('Please Fill Out All Fields');
    return;
  }

  dispatch(createJob({ position, company, jobLocation, jobType, status }));
};
```

56) Use Existing User Location

AddJob.js

```
const { user } = useSelector((store) => store.user);

useEffect(() => {
  // eventually will check for isEditing
  if (!isEditing) {
    dispatch(handleChange({ name: 'jobLocation', value: user.location }));
  }
}, []);
```

jobSlice.js

```
    // reducers
    clearValues: () => {
      return {
        ...initialState,
        jobLocation: getUserFromLocalStorage()?.location || '',
      };
    },
```

57) Logout Message

userSlice.js

```
logoutUser: (state,{payload}) => {
      state.user = null;
      state.isSidebarOpen = false;
      removeUserFromLocalStorage();
      if(payload){
        toast.success(payload)
      }
    },
```

Navbar.js

```
<button
  type='button'
  className='dropdown-btn'
  onClick={() => dispatch(logoutUser('Logging out...'))}
>
  logout
</button>
```

58) AllJobs Slice

- features/allJobs/allJobsSlice.js

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import { toast } from 'react-toastify';
import customFetch from '../../utils/axios';

const initialFiltersState = {
  search: '',
  searchStatus: 'all',
  searchType: 'all',
  sort: 'latest',
  sortOptions: ['latest', 'oldest', 'a-z', 'z-a'],
};

const initialState = {
  isLoading: false,
  jobs: [],
  totalJobs: 0,
  numOfPages: 1,
  page: 1,
  stats: {},
  monthlyApplications: [],
  ...initialFiltersState,
};

const allJobsSlice = createSlice({
  name: 'allJobs',
  initialState,
});

export default allJobsSlice.reducer;
```

store.js

```
import { configureStore } from '@reduxjs/toolkit';

import userSlice from './features/user/userSlice';
import jobSlice from './features/job/jobSlice';
import allJobsSlice from './features/allJobs/allJobsSlice';

export const store = configureStore({
  reducer: {
    user: userSlice,
    job: jobSlice,
    allJobs: allJobsSlice,
  },
});
```

59) AllJobs Page Structure

- create
- components/SearchContainer.js
- components/JobsContainer.js
- components/Job.js
- import/export

AllJobs.js

```
import { JobsContainer, SearchContainer } from '../../components';

const AllJobs = () => {
  return (
    <>
      <SearchContainer />
      <JobsContainer />
    </>
  );
};

export default AllJobs;
```

60) JobsContainer.js

```
import { useEffect } from 'react';
import Job from './Job';
import Wrapper from '../assets/wrappers/JobsContainer';
import { useSelector, useDispatch } from 'react-redux';

const JobsContainer = () => {
  const { jobs, isLoading } = useSelector((store) => store.allJobs);
  const dispatch = useDispatch();

  if (isLoading) {
    return (
      <Wrapper>
        <h2>Loading...</h2>
      </Wrapper>
    );
  }

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

  return (
    <Wrapper>
      <h5>jobs info</h5>
      <div className='jobs'>
        {jobs.map((job) => {
          return <Job key={job._id} {...job} />;
        })}
      </div>
    </Wrapper>
  );
};

export default JobsContainer;
```

CSS Only Loading Spinner

Loading.js

```
const Loading = ({ center }) => {
  return <div className={center ? 'loading loading-center' : 'loading'}></div>;
};

export default Loading;
```

JobsContainer.js

```
import Loading from './Loading';

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

61) GetAllJobs Request

- GET /jobs
- authorization header : 'Bearer token'
- returns {jobs:[],totalJobs:number, numOfPages:number }

allJobsSlice.js

```
export const getAllJobs = createAsyncThunk(
  'allJobs/getJobs',
  async (_, thunkAPI) => {
    let url = `/jobs`;

    try {
      const resp = await customFetch.get(url, {
        headers: {
          authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
        },
      });

      return resp.data;
    } catch (error) {
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);

// extra reducers

extraReducers: {
    [getAllJobs.pending]: (state) => {
      state.isLoading = true;
    },
    [getAllJobs.fulfilled]: (state, { payload }) => {
      state.isLoading = false;
      state.jobs = payload.jobs;
    },
    [getAllJobs.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    },
}
```

JobsContainer.js

```
import { getAllJobs } from '../features/allJobs/allJobsSlice';

useEffect(() => {
  dispatch(getAllJobs());
}, []);
```

62) Single Job

Job.js

```jsx
import { FaLocationArrow, FaBriefcase, FaCalendarAlt } from 'react-icons/fa';
import { Link } from 'react-router-dom';
import Wrapper from '../assets/wrappers/Job';
import { useDispatch } from 'react-redux';

const Job = ({
  _id,
  position,
  company,
  jobLocation,
  jobType,
  createdAt,
  status,
}) => {
  const dispatch = useDispatch();

  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'>
        <div className='content-center'>
          <h4>more content</h4>
          <div className={`status ${status}`}>{status}</div>
        </div>
        <footer>
          <div className='actions'>
            <Link
              to='/add-job'
              className='btn edit-btn'
              onClick={() => {
                console.log('edit job');
              }}
            >
              Edit
            </Link>
            <button
              type='button'
              className='btn delete-btn'
              onClick={() => {
                console.log('delete  job');
              }}
            >
              Delete
            </button>
```

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

export default Job;
```

63) JobInfo

- components/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;
```

Job.js

```
const date = createdAt


<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>
```

64) Moment.js

moment.js

```
npm install moment
```

Job.js

```
const date = moment(createdAt).format('MMM Do, YYYY');
```

65) Toggle Loading in AllJobs

allJobsSlice.js

```
reducers: {
    showLoading: (state) => {
      state.isLoading = true;
    },
    hideLoading: (state) => {
      state.isLoading = false;
    },
}
export const {
  showLoading,
  hideLoading,
} = allJobsSlice.actions;
```

66) Delete Job Request

- DELETE /jobs/jobId
- authorization header : 'Bearer token'

jobSlice.js

```
import { showLoading, hideLoading, getAllJobs } from '../allJobs/allJobsSlice';

export const deleteJob = createAsyncThunk(
  'job/deleteJob',
  async (jobId, thunkAPI) => {
    thunkAPI.dispatch(showLoading());
    try {
      const resp = await customFetch.delete(`/jobs/${jobId}`, {
        headers: {
          authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
        },
      });
      thunkAPI.dispatch(getAllJobs());
      return resp.data;
    } catch (error) {
      thunkAPI.dispatch(hideLoading());
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);
```

Job.js

```
<button
  type='button'
  className='btn delete-btn'
  onClick={() => {
    dispatch(deleteJob(_id));
  }}
>
  Delete
</button>
```

67) SetEditJob Reducer

jobSlice.js

```
reducers:{

  setEditJob: (state, { payload }) => {
    return { ...state, isEditing: true, ...payload };
    },
  }

  export const { handleChange, clearValues, setEditJob } = jobSlice.actions;
```

Job.js

```
import { setEditJob, deleteJob } from '../features/job/jobSlice';

<Link
  to='/add-job'
  className='btn edit-btn'
  onClick={() => {
    dispatch(
      setEditJob({
        editJobId: _id,
        position,
        company,
        jobLocation,
        jobType,
        status,
      })
    );
  }}
>
  Edit
</Link>;
```

AddJob.js

```
useEffect(() => {
  if (!isEditing) {
    dispatch(handleChange({ name: 'jobLocation', value: user.location }));
  }
}, []);
```

68) EditJob Request

- PATCH /jobs/jobId
- { position:'position', company:'company', jobLocation:'location', jobType:'full-time', status:'pending' }
- authorization header : 'Bearer token'
- sends back the updated job object

jobSlice.js

```
export const editJob = createAsyncThunk(
  'job/editJob',
  async ({ jobId, job }, thunkAPI) => {
    try {
      const resp = await customFetch.patch(`/jobs/${jobId}`, job, {
        headers: {
          authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
        },
      });
      thunkAPI.dispatch(clearValues());
      return resp.data;
    } catch (error) {
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);


extraReducers:{
  [editJob.pending]: (state) => {
      state.isLoading = true;
    },
    [editJob.fulfilled]: (state) => {
      state.isLoading = false;
      toast.success('Job Modified...');
    },
    [editJob.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    },
}
```

AddJob.js

```
import {
  clearValues,
  handleChange,
  createJob,
  editJob,
} from '../../features/job/jobSlice';

if (isEditing) {
  dispatch(
    editJob({
      jobId: editJobId,
      job: {
        position,
        company,
        jobLocation,
        jobType,
        status,
      },
    })
  );
  return;
}
```

69) Job Thunk

- features/job/jobThunk.js

```
import customFetch from '../../utils/axios';
import { showLoading, hideLoading, getAllJobs } from '../allJobs/allJobsSlice';
import { clearValues } from './jobSlice';

export const createJobThunk = async (job, thunkAPI) => {
  try {
    const resp = await customFetch.post('/jobs', job, {
      headers: {
        authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
      },
    });
    thunkAPI.dispatch(clearValues());
    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};

export const deleteJobThunk = async (jobId, thunkAPI) => {
  thunkAPI.dispatch(showLoading());
  try {
    const resp = await customFetch.delete(`/jobs/${jobId}`, {
      headers: {
        authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
      },
    });
    thunkAPI.dispatch(getAllJobs());
    return resp.data;
  } catch (error) {
    thunkAPI.dispatch(hideLoading());
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};

export const editJobThunk = async ({ jobId, job }, thunkAPI) => {
  try {
    const resp = await customFetch.patch(`/jobs/${jobId}`, job, {
      headers: {
        authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
      },
    });
    thunkAPI.dispatch(clearValues());
    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};
```

jobSlice.js

```
import { createJobThunk, deleteJobThunk, editJobThunk } from './jobThunk';

export const createJob = createAsyncThunk('job/createJob', createJobThunk);

export const deleteJob = createAsyncThunk('job/deleteJob', deleteJobThunk);

export const editJob = createAsyncThunk('job/editJob', editJobThunk);
```

70) AuthHeader - File Approach

jobThunk.js

```
const authHeader = (thunkAPI) => {
  return {
    headers: {
      authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
    },
  };
};

export const createJobThunk = async (job, thunkAPI) => {
  try {
    const resp = await customFetch.post('/jobs', job, authHeader(thunkAPI));
    thunkAPI.dispatch(clearValues());
    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};
```

70) AuthHeader - Utils Approach

- create utils/authHeader.js

```
const authHeader = (thunkAPI) => {
  return {
    headers: {
      authorization: `Bearer ${thunkAPI.getState().user.user.token}`,
    },
  };
};

export default authHeader;
```

jobThunk.js

```
import authHeader from '../../utils/authHeader';
```

72) AuthHeader - Axios Interceptors Approach

- utils/axios.js

```
import axios from 'axios';
import { getUserFromLocalStorage } from './localStorage';

const customFetch = axios.create({
  baseURL: 'https://jobify-prod.herokuapp.com/api/v1/toolkit',
});

customFetch.interceptors.request.use(
  (config) => {
    const user = getUserFromLocalStorage();
    if (user) {
      config.headers['Authorization'] = `Bearer ${user.token}`;
      // in the latest version "common" returns undefined
      // config.headers.common['Authorization'] = `Bearer ${user.token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

export default customFetch;
```

- remove auth header

73) Test User

- email : testUser@test.com
- password : secret
- read only!
- dummy data

Register.js

```
<button
  type='button'
  className='btn btn-block btn-hipster'
  disabled={isLoading}
  onClick={() => {
    dispatch(loginUser({ email: 'testUser@test.com', password: 'secret' }));
  }}
>
  {isLoading ? 'loading...' : 'demo'}
</button>
```

74) Get Stats Request

- GET /jobs/stats
- authorization header : 'Bearer token'
- returns
  {
  defaultStats:{pending:24,interview:27,declined:24},
  monthlyApplications:[{date:"Nov 2021",count:5},{date:"Dec 2021",count:4}]
  }
- last six months
  allJobsSlice.js

```
export const showStats = createAsyncThunk(
  'allJobs/showStats',
  async (_, thunkAPI) => {
    try {
      const resp = await customFetch.get('/jobs/stats');
      console.log(resp.data));
      return resp.data;
    } catch (error) {
      return thunkAPI.rejectWithValue(error.response.data.msg);
    }
  }
);

// extraReducers

    [showStats.pending]: (state) => {
      state.isLoading = true;
    },
    [showStats.fulfilled]: (state, { payload }) => {
      state.isLoading = false;
      state.stats = payload.defaultStats;
      state.monthlyApplications = payload.monthlyApplications;
    },
    [showStats.rejected]: (state, { payload }) => {
      state.isLoading = false;
      toast.error(payload);
    },
```

75) Stats Page

- create
- components/StatsContainer.js
- components/ChartsContainer.js
- import/export

Stats.js

```jsx
import { useEffect } from 'react';
import { StatsContainer, Loading, ChartsContainer } from '../../components';
import { useDispatch, useSelector } from 'react-redux';
import { showStats } from '../../features/allJobs/allJobsSlice';
const Stats = () => {
  const { isLoading, monthlyApplications } = useSelector(
    (store) => store.allJobs
  );
  const dispatch = useDispatch();
  useEffect(() => {
    dispatch(showStats());
    // eslint-disable-next-line
  }, []);
  if (isLoading) {
    return <Loading center />;
  }
  return (
    <>
      <StatsContainer />
      {monthlyApplications.length > 0 && <ChartsContainer />}
    </>
  );
};

export default Stats;
```

76) Stats Container

- create components/StatItem.js

StatsContainer.js

```
import StatItem from './StatItem';
import { FaSuitcaseRolling, FaCalendarCheck, FaBug } from 'react-icons/fa';
import Wrapper from '../assets/wrappers/StatsContainer';
import { useSelector } from 'react-redux';
const StatsContainer = () => {
  const { stats } = useSelector((store) => store.allJobs);
  const defaultStats = [
    {
      title: 'pending applications',
      count: stats.pending || 0,
      icon: <FaSuitcaseRolling />,
      color: '#e9b949',
      bcg: '#fcefc7',
    },
    {
      title: 'interviews scheduled',
      count: stats.interview || 0,
      icon: <FaCalendarCheck />,
      color: '#647acb',
      bcg: '#e0e8f9',
    },
    {
      title: 'jobs declined',
      count: stats.declined || 0,
      icon: <FaBug />,
      color: '#d66a6a',
      bcg: '#ffeeee',
    },
  ];

  return (
    <Wrapper>
      {defaultStats.map((item, index) => {
        return <StatItem key={index} {...item} />;
      })}
    </Wrapper>
  );
};

export default StatsContainer;
```

77) Stat Item

StatItem.js

```
import Wrapper from '../assets/wrappers/StatItem';

const StatItem = ({ count, title, icon, color, bcg }) => {
  return (
    <Wrapper color={color} bcg={bcg}>
      <header>
        <span className='count'>{count}</span>
        <span className='icon'>{icon}</span>
      </header>
      <h5 className='title'>{title}</h5>
    </Wrapper>
  );
};

export default StatItem;
```

78) Charts Container

- create
- components/AreaChart.js
- components/BarChart.js

ChartsContainer.js

```
import React, { useState } from 'react';

import BarChart from './BarChart';
import AreaChart from './AreaChart';
import Wrapper from '../assets/wrappers/ChartsContainer';
import { useSelector } from 'react-redux';
const ChartsContainer = () => {
  const [barChart, setBarChart] = useState(true);
  const { monthlyApplications: data } = useSelector((store) => store.allJobs);
  return (
    <Wrapper>
      <h4>Monthly Applications</h4>
      <button type='button' onClick={() => setBarChart(!barChart)}>
        {barChart ? 'Area Chart' : 'Bar Chart'}
      </button>
      {barChart ? <BarChart data={data} /> : <AreaChart data={data} />}
    </Wrapper>
  );
};

export default ChartsContainer;
```

79) Recharts Library

Recharts

```
npm install recharts
```

- For now does not work with React 18

```
npm install react@17 react-dom@17
```

```
npm install recharts
```

```
npm install react@18 react-dom@18
```

80) AreaChart

AreaChart.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='#1e3a8a' fill='#3b82f6' />
      </AreaChart>
    </ResponsiveContainer>
  );
};

export default AreaChartComponent;
```

81) BarChart.js

```
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  CartesianGrid,
  Tooltip,
  ResponsiveContainer,
} from 'recharts';

const BarChartComponent = ({ data }) => {
  return (
    <ResponsiveContainer width='100%' height={300}>
      <BarChart data={data} margin={{ top: 50 }}>
        <CartesianGrid strokeDasharray='3 3 ' />
        <XAxis dataKey='date' />
        <YAxis allowDecimals={false} />
        <Tooltip />
        <Bar dataKey='count' fill='#3b82f6' barSize={75} />
      </BarChart>
    </ResponsiveContainer>
  );
};

export default BarChartComponent;
```

82) Search Container

SearchContainer.js

```jsx
import { FormRow, FormRowSelect } from '.';
import Wrapper from '../assets/wrappers/SearchContainer';
import { useSelector, useDispatch } from 'react-redux';

const SearchContainer = () => {
  const { isLoading, search, searchStatus, searchType, sort, sortOptions } =
    useSelector((store) => store.allJobs);
  const { jobTypeOptions, statusOptions } = useSelector((store) => store.job);
  const dispatch = useDispatch();
  const handleSearch = (e) => {};
  const handleSubmit = (e) => {
    e.preventDefault();
  };
  return (
    <Wrapper>
      <form className='form'>
        <h4>search form</h4>
        <div className='form-center'>
          {/* search position */}

          <FormRow
            type='text'
            name='search'
            value={search}
            handleChange={handleSearch}
          />
          {/* search by status */}
          <FormRowSelect
            labelText='status'
            name='searchStatus'
            value={searchStatus}
            handleChange={handleSearch}
            list={['all', ...statusOptions]}
          />
          {/* search by type */}
          <FormRowSelect
            labelText='type'
            name='searchType'
            value={searchType}
            handleChange={handleSearch}
            list={['all', ...jobTypeOptions]}
          />
          {/* sort */}
          <FormRowSelect
            name='sort'
            value={sort}
            handleChange={handleSearch}
            list={sortOptions}
          />
          <button
```

```
            className='btn btn-block btn-danger'
            disabled={isLoading}
            onClick={handleSubmit}
          >
            clear filters
          </button>
        </div>
      </form>
    </Wrapper>
  );
};

export default SearchContainer;
```

83) Handle Change and Clear Filters

allJobsSlice.js

```
reducers:{
  handleChange: (state, { payload: { name, value } }) => {
      // state.page = 1;
      state[name] = value;
    },
    clearFilters: (state) => {
      return { ...state, ...initialFiltersState };
    },
}

export const { showLoading, hideLoading, handleChange, clearFilters } =
  allJobsSlice.actions;
```

SearchContainer.js

```
import { handleChange, clearFilters } from '../features/allJobs/allJobsSlice';

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

84) Pagination Setup

- server returns 10 jobs per page

- create
- components/PageBtnContainer.js

allJobsSlice.js

```
    extraReducers:{

    [getAllJobs.fulfilled]: (state, { payload }) => {
      state.isLoading = false;
      state.jobs = payload.jobs;
      state.numOfPages = payload.numOfPages;
      state.totalJobs = payload.totalJobs;
    },
  }
```

JobsContainer

```
const { jobs, isLoading, page, totalJobs, numOfPages } = useSelector(
  (store) => store.allJobs
);

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>
);
```

85) PageBtnContainer Structure

JS Nuggets - Array.from()

```jsx
import { HiChevronDoubleLeft, HiChevronDoubleRight } from 'react-icons/hi';
import Wrapper from '../assets/wrappers/PageBtnContainer';
import { useSelector, useDispatch } from 'react-redux';
const PageBtnContainer = () => {
  const { numOfPages, page } = useSelector((store) => store.allJobs);
  const dispatch = useDispatch();

  const pages = Array.from({ length: numOfPages }, (_, index) => {
    return index + 1;
  });
  const nextPage = () => {};
  const prevPage = () => {};
  return (
    <Wrapper>
      <button className='prev-btn' onClick={prevPage}>
        <HiChevronDoubleLeft />
        prev
      </button>
      <div className='btn-container'>
        {pages.map((pageNumber) => {
          return (
            <button
              type='button'
              className={pageNumber === page ? 'pageBtn active' : 'pageBtn'}
              key={pageNumber}
              onClick={() => console.log('change page')}
            >
              {pageNumber}
            </button>
          );
        })}
      </div>
      <button className='next-btn' onClick={nextPage}>
        next
        <HiChevronDoubleRight />
      </button>
    </Wrapper>
  );
};

export default PageBtnContainer;
```

86) Change Page

allJobsSlice.js

```
reducers:{
  changePage: (state, { payload }) => {
      state.page = payload;
    },
}
export const {
  showLoading,
  hideLoading,
  handleChange,
  clearFilters,
  changePage,
} = allJobsSlice.actions;
```

PageBtnContainer.js

```
import { changePage } from '../features/allJobs/allJobsSlice';

const nextPage = () => {
  let newPage = page + 1;
  if (newPage > numOfPages) {
    newPage = 1;
  }
  dispatch(changePage(newPage));
};
const prevPage = () => {
  let newPage = page - 1;
  if (newPage < 1) {
    newPage = numOfPages;
  }
  dispatch(changePage(newPage));
};

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

87) Query String Params

allJobsSlice

```
export const getAllJobs = createAsyncThunk(
  'allJobs/getJobs',
  async (_, thunkAPI) => {
    const { page, search, searchStatus, searchType, sort } =
      thunkAPI.getState().allJobs;

    let url = `/jobs?status=${searchStatus}&jobType=${searchType}&sort=${sort}&page=${page}`;
    if (search) {
      url = url + `&search=${search}`;
    }
    try {
      const resp = await customFetch.get(url);
      return resp.data;
    }
  }
)
```

JobsContainer.js

```
const {
  jobs,
  isLoading,
  page,
  totalJobs,
  numOfPages,
  search,
  searchStatus,
  searchType,
  sort,
} = useSelector((store) => store.allJobs);

useEffect(() => {
  dispatch(getAllJobs());
  // eslint-disable-next-line
}, [page, search, searchStatus, searchType, sort]);
```

88) Change Page and isLoading

allJobsSlice.js

```
reducers:{
  handleChange: (state, { payload: { name, value } }) => {
      state.page = 1;
      state[name] = value;
    },
```

SearchContainer.js

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

89) Refactor allJobsSlice.js

- create
- features/allJobs/allJobsThunk.js

```
import customFetch from '../../utils/axios';

export const getAllJobsThunk = async (thunkAPI) => {
  const { page, search, searchStatus, searchType, sort } =
    thunkAPI.getState().allJobs;

  let url = `/jobs?page=${page}&status=${searchStatus}&jobType=${searchType}&sort=${sort}`;
  if (search) {
    url = url + `&search=${search}`;
  }
  try {
    const resp = await customFetch.get(url);

    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};

export const showStatsThunk = async (_, thunkAPI) => {
  try {
    const resp = await customFetch.get('/jobs/stats');
    return resp.data;
  } catch (error) {
    return thunkAPI.rejectWithValue(error.response.data.msg);
  }
};
```

allJobsSlice.js

```
import { showStatsThunk, getAllJobsThunk } from './allJobsThunk';

export const getAllJobs = createAsyncThunk('allJobs/getJobs', getAllJobsThunk);
export const showStats = createAsyncThunk('allJobs/showStats', showStatsThunk);
```

90) Clear Store - Setup

allJobsSlice.js

```
reducers:{
  clearAllJobsState: () => initialState,
}
```

91) clearStore

userThunk.js

```
import { logoutUser } from './userSlice';
import { clearAllJobsState } from '../allJobs/allJobsSlice';
import { clearValues } from '../job/jobSlice';

export const clearStoreThunk = async (message, thunkAPI) => {
  try {
    // logout user
    thunkAPI.dispatch(logoutUser(message));
    // clear jobs value
    thunkAPI.dispatch(clearAllJobsState());
    // clear job input values
    thunkAPI.dispatch(clearValues());
    return Promise.resolve();
  } catch (error) {
    // console.log(error);
    return Promise.reject();
  }
};
```

userSlice.js

```
import { clearStoreThunk } from './userThunk';
export const clearStore = createAsyncThunk('user/clearStore', clearStoreThunk);

extraReducers:{
  [clearStore.rejected]: () => {
      toast.error('There was an error');
    },
}
```

Navbar.js

```
import { clearStore } from '../features/user/userSlice';

return (
  <button
    type='button'
    className='dropdown-btn'
    onClick={() => {
      dispatch(clearStore('Logout Successful...'));
    }}
  >
    logout
  </button>
);
```

92) UnAuthorized Requests

axios.js

```
import { clearStore } from '../features/user/userSlice';

export const checkForUnauthorizedResponse = (error, thunkAPI) => {
  if (error.response.status === 401) {
    thunkAPI.dispatch(clearStore());
    return thunkAPI.rejectWithValue('Unauthorized! Logging Out...');
  }
  return thunkAPI.rejectWithValue(error.response.data.msg);
};
```

allJobsThunk.js

```
import customFetch, { checkForUnauthorizedResponse } from '../../utils/axios';

export const showStatsThunk = async (_, thunkAPI) => {
  try {
    const resp = await customFetch.get('/jobs/stats');
    return resp.data;
  } catch (error) {
    return checkForUnauthorizedResponse(error, thunkAPI);
  }
};
```

- refactor in all authenticated requests

Refactor All Extra Reducers to Builder Callback Notation

allJobsSlice.js

```
  extraReducers: (builder) => {
     builder
       .addCase(getAllJobs.pending, (state) => {
         state.isLoading = true;
       })
       .addCase(getAllJobs.fulfilled, (state, { payload }) => {
         state.isLoading = false;
         state.jobs = payload.jobs;
         state.numOfPages = payload.numOfPages;
         state.totalJobs = payload.totalJobs;
       })
       .addCase(getAllJobs.rejected, (state, { payload }) => {
         state.isLoading = false;
         toast.error(payload);
       })
       .addCase(showStats.pending, (state) => {
         state.isLoading = true;
       })
       .addCase(showStats.fulfilled, (state, { payload }) => {
         state.isLoading = false;
         state.stats = payload.defaultStats;
         state.monthlyApplications = payload.monthlyApplications;
       })
       .addCase(showStats.rejected, (state, { payload }) => {
         state.isLoading = false;
         toast.error(payload);
       });
   },
```

jobSlice.js

```
  extraReducers: (builder) => {
      builder
        .addCase(createJob.pending, (state) => {
          state.isLoading = true;
        })
        .addCase(createJob.fulfilled, (state) => {
          state.isLoading = false;
          toast.success('Job Created');
        })
        .addCase(createJob.rejected, (state, { payload }) => {
          state.isLoading = false;
          toast.error(payload);
        })
        .addCase(deleteJob.fulfilled, (state, { payload }) => {
          toast.success(payload);
        })
        .addCase(deleteJob.rejected, (state, { payload }) => {
          toast.error(payload);
        })
        .addCase(editJob.pending, (state) => {
          state.isLoading = true;
        })
        .addCase(editJob.fulfilled, (state) => {
          state.isLoading = false;
          toast.success('Job Modified...');
        })
        .addCase(editJob.rejected, (state, { payload }) => {
          state.isLoading = false;
          toast.error(payload);
        });
    },
```

userSlice.js

```
extraReducers: (builder) => {
   builder
     .addCase(registerUser.pending, (state) => {
       state.isLoading = true;
     })
     .addCase(registerUser.fulfilled, (state, { payload }) => {
       const { user } = payload;
       state.isLoading = false;
       state.user = user;
       addUserToLocalStorage(user);
       toast.success(`Hello There ${user.name}`);
     })
     .addCase(registerUser.rejected, (state, { payload }) => {
       state.isLoading = false;
       toast.error(payload);
     })
     .addCase(loginUser.pending, (state) => {
       state.isLoading = true;
     })
     .addCase(loginUser.fulfilled, (state, { payload }) => {
       const { user } = payload;
       state.isLoading = false;
       state.user = user;
       addUserToLocalStorage(user);

       toast.success(`Welcome Back ${user.name}`);
     })
     .addCase(loginUser.rejected, (state, { payload }) => {
       state.isLoading = false;
       toast.error(payload);
     })
     .addCase(updateUser.pending, (state) => {
       state.isLoading = true;
     })
     .addCase(updateUser.fulfilled, (state, { payload }) => {
       const { user } = payload;
       state.isLoading = false;
       state.user = user;
       addUserToLocalStorage(user);

       toast.success(`User Updated!`);
     })
     .addCase(updateUser.rejected, (state, { payload }) => {
       state.isLoading = false;
       toast.error(payload);
     })
     .addCase(clearStore.rejected, () => {
       toast.error('There was an error..');
     });
 },
```

Switch To Local Search

- remove isLoading from handleSearch
- import useState and useMemo from react
- setup localSearch state value
- replace search input functionality so it updates localSearch

```
import { useState, useMemo } from 'react';

const SearchContainer = () => {
  const [localSearch, setLocalSearch] = useState('');

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

  return (
    <Wrapper>
      <form className='form'>
        <h4>search form</h4>
        <div className='form-center'>
          {/* search position */}
          <FormRow
            type='text'
            name='search'
            value={localSearch}
            handleChange={(e) => setLocalSearch(e.target.value)}
          />
          // ...rest of the code
        </div>
      </form>
    </Wrapper>
  );
};
export default SearchContainer;
```

Setup Debounce

```
import { useState, useMemo } from 'react';

const SearchContainer = () => {
  const [localSearch, setLocalSearch] = useState('');

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

  const debounce = () => {
    let timeoutID;
    return (e) => {
      setLocalSearch(e.target.value);
      clearTimeout(timeoutID);
      timeoutID = setTimeout(() => {
        dispatch(handleChange({ name: e.target.name, value: e.target.value }));
      }, 1000);
    };
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    setLocalSearch('');
    dispatch(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}
          />
          // ...rest of the code
        </div>
      </form>
    </Wrapper>
  );
};
export default SearchContainer;
```