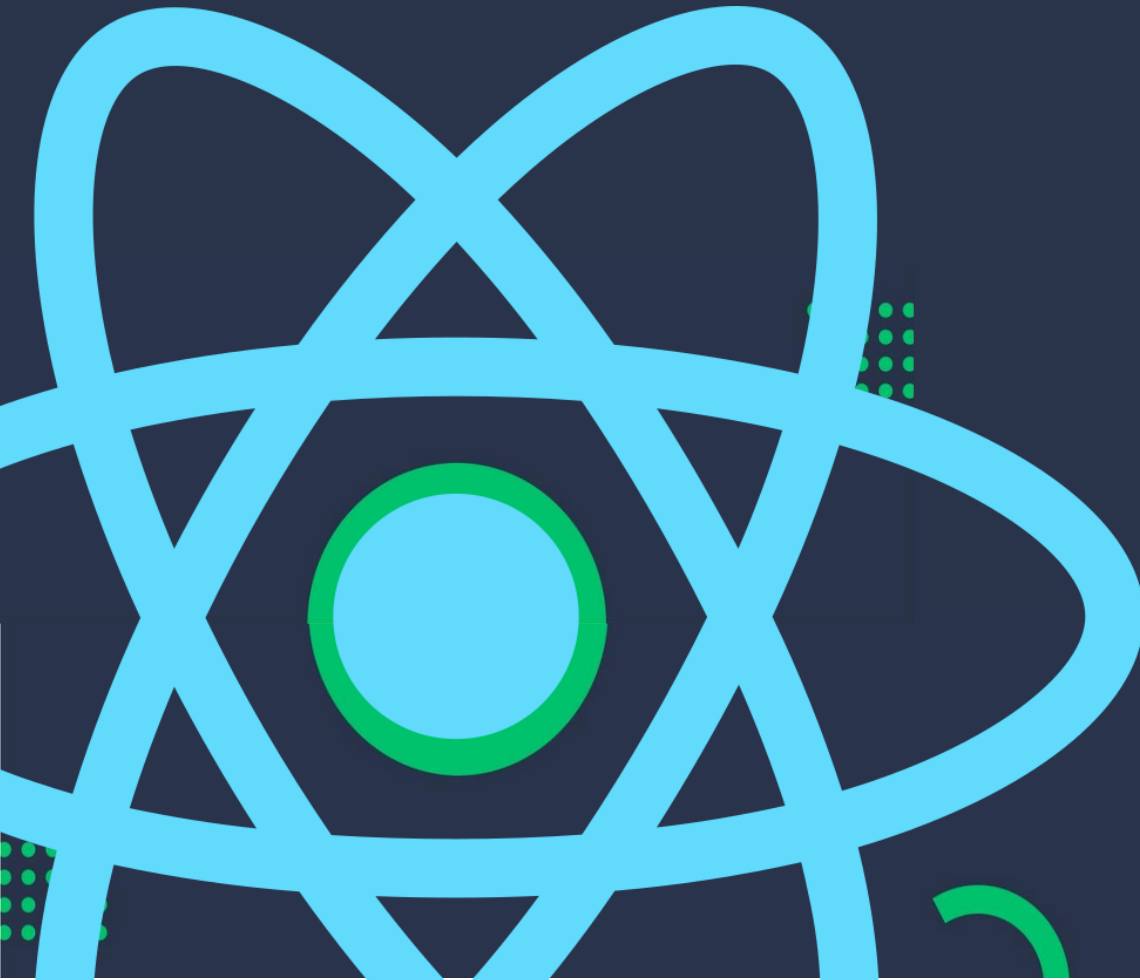


Front-End Bootcamp Final Project

HAMİ BERKAY AKTAŞ

tech
career



İçindekiler

React Kütüphaneleri

1. [lottiefiles/react-lottie-player](#) 3.4.7
2. [mui/material](#) 5.9.0
3. [reduxjs/toolkit](#) 1.8.3
4. [axios](#) 0.27.2
5. [bootstrap](#) 2.0.0
6. [formik](#) 2.2.9
7. [react](#) 18.2.0
8. [react-animated-numbers](#) 0.13.0
9. [react-circular-progressbar](#) 2.1.0
10. [react-hot-toast](#) 2.3.0
11. [react-redux](#) 8.0.2
12. [react-router-dom](#) 6.3.0
13. [react-wow](#) 1.0.0
14. [wowjs](#) 1.1.3
15. [yup](#) 0.32.11

1 Componentlerin React Yapısına Uygun Şekilde Jsx'e Çevrilmesi

İlk önce clean kodumuzu html to jsx çevirici ile reacte uygun hale getirdim ardından sub componentler olaralar Components klasörü altında topladım. Ana yapıyı inşa ettikten sonra index.html'ye bootstrap cdn'ini ekledim enson olarakda react router dom kurarak site iskeletini ayağa kaldırmış oldum

2. React Router Dom Kurulumu ve Routeları tanımlama

```
<BrowserRouter>
  <div data-bs-spy="scroll" data-bs-target="#navbar-example">
    <Navbar />

    <div data-bs-spy="scroll" data-bs-target="#navbar-example"
      data-bs-smooth-scroll="true" className="scrollspy-example" tabIndex="0">

      <Routes>
        <Route path="/" exact element={<MainPage />} />
        <Route path="/todo" element={<TodoPage />} />
        <Route path="/aboutme" element={<AboutMePage />} />
        <Route path="/profile" element={<ProfilePage />} />
      </Routes>
    </div>
  </div>
  <Footer/>
  <BackToTop/>
</BrowserRouter>
```

Projemizde bir giriş kısmı var ana sayfa bir adet todo sayfamız bir adet profile sayfası login ve register kısmından oluşur ve son olarak da about me kısmımız bulunuyor

Bu route larımızı ve browser router kısmımızı ayarladıktan sonra bu sayfamızı oluştururuz sonra burada routes komponenti içinde renderlarız

3 Her sayfada olan komponentler

3.a Navbar

3.b Footer

3.c BacktoTop

3.a Navbar

Bu komponentimizde ilk olarak react router dom yapılarından link yapısını kullanacağız

Ardından arama kısmında auto complete için material ui auto complete componentini kullanacağız

İcon kısımlarında font awesome kullandık ve modal yapılarında önyüz validasyonu yapmak için yup validatör ve formik kullandım

Register

Name

Zorunlu alan.

Username

dsa

username en az 5 karakter olmalıdır.

Email

ads

Geçerli bir email girin.

Password

....

Parolanız en az 5 karakter olmalıdır.

Register

Login

Email

dasads

Geçerli bir email girin.

Password

....

Parolanız en az 5 karakter olmalıdır.

Sign in

```
const formik = useFormik({
  initialValues: {
    email: "",
    password: "",
  },
  validationSchema,
  onSubmit: async (values, bag) => {
    try{
      dispatch(postLoginAsync({
        email:values.email,
        password:values.password
      })))

      //console.log(Neco)

      //
    }catch(e){
      bag.setErrors({general:e.response.data.error});
      //console.log(e.response.data.error)
    }
  },
});
```

Formik içerisinde yup olarak validation shema tanımlanır (aşağıda göstermekteyim)

```

1 import * as yup from "yup";
2 const validations = yup.object().shape({
3   email: yup
4     .string()
5     .email("Geçerli bir email girin.")
6     .required("Zorunlu alan."),
7   password: yup
8     .string()
9     .min(5, "Parolanız en az 5 karakter olmalıdır.")
10    .required("Zorunlu alan."),
11 });
12 export default validations;
13

```

Yup dosyamızda neler istiyoruz bunları yazarız.

Burada yupdan gelen hatayı formik error bag e atarım ve ardından eğer kullanıcı input field 'a dokunup boş ayrıldıysa hatayı bastırırım ve input field'ın çevresinin kırmızı yaparım.

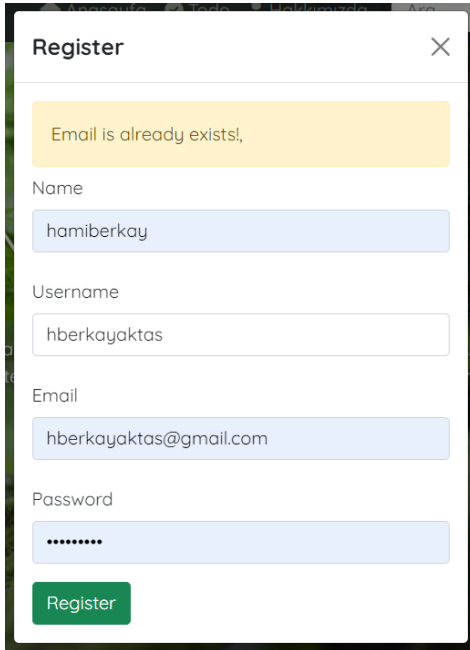
Önyüz ve filtrelemelerini yaptıktan sonra arka planda validation işlemi için Express-validator ile validation yapıyoruz burada custom bir Validation error'u ekleyebiliyoruz.

```

1 const User = require("../models/user");
2 const { body } = require("express-validator");
3
4 exports.registerValidator = [
5   body("name").not().isEmpty().withMessage("Please Enter your Name"),
6   body("userName")
7     .custom((username) => {
8       return User.findOne({ userName: username }).then((user) => {
9         if (user) {
10           return Promise.reject("User name is already exists!");
11         }
12       });
13     }),
14   body("email")
15     .isEmail()
16     .withMessage("Please Enter Valid Email")
17     .custom((userEmail) => {
18       return User.findOne({ email: userEmail }).then((user) => {
19         if (user) {
20           return Promise.reject("Email is already exists!");
21         }
22       });
23     }),
24   body("password").not().isEmpty().withMessage("Please Enter A Password <br>"),
25 ];

```

Hatanın geri çıktısı

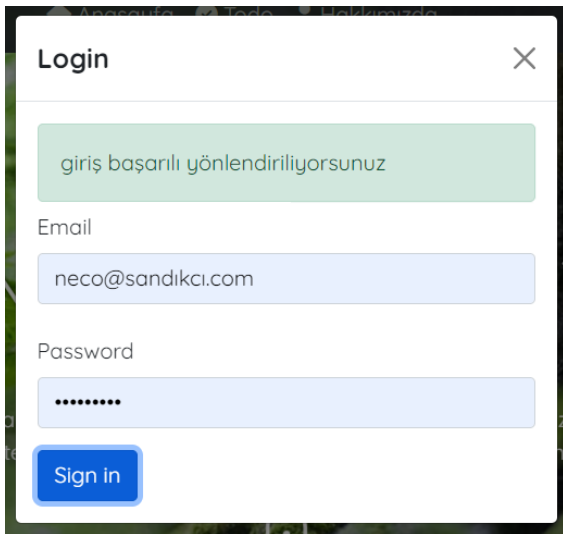


A screenshot of a web application's 'Register' form. The form is titled 'Register' with a close button (X) in the top right corner. It contains several input fields: 'Name' (filled with 'hamiberkay'), 'Username' (filled with 'hberkayaktas'), 'Email' (filled with 'hberkayaktas@gmail.com'), and 'Password' (filled with eight dots). A green 'Register' button is at the bottom. A yellow error message box at the top left of the form area displays the text 'Email is already exists!,'.

Veritabanında aradıktan sonra uniq olsun diye hata geri döner.

Eğer başarılı bir giriş yapıldı ise react router dom elemanlarından

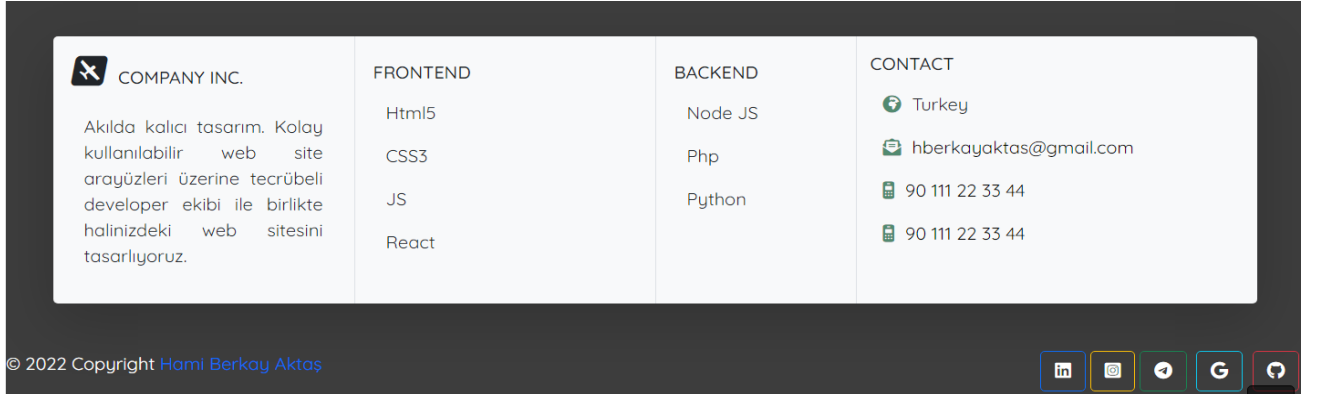
Use navigate ile sayfa yönlendirmesi yapılır.



A screenshot of a web application's 'Login' form. The form is titled 'Login' with a close button (X) in the top right corner. It contains two input fields: 'Email' (filled with 'neco@sandikci.com') and 'Password' (filled with eight dots). A blue 'Sign in' button is at the bottom. A green success message box at the top left of the form area displays the text 'giriş başarılı yönlendiriliyorsunuz'.

3.b Footer

Footer kısmında özet bir bilgi ekleyip görsel düzenlemeler yaptım



3.c BacktoTop



```
1
2 import React, { useState, useEffect } from 'react';
3
4 function BackToTop() {
5   // inside component:
6
7   const [rightPosition, setRightPosition] = useState("-65px");
8   const handleScroll = () => {
9     const position = window.pageYOffset; // 1
10    //console.log(position);
11    if(position < 100){
12      setRightPosition("-65px") // 2
13    }else{
14      setRightPosition("0px") // 3
15    }
16  };
17
18  useEffect(() => {
19    window.addEventListener('scroll', handleScroll, { passive: true });
20
21    return () => {
22      window.removeEventListener('scroll', handleScroll);
23    };
24  }, []);
25
26  return (
27    <a className="backtotop" style={{right:rightPosition}} id="scrollToTop" href={"#Top"} ><span>&lt;</span></a>
28  )
29 }
30
31 export default BackToTop
```

Bu komponent içerisinde ilk önce sayfa konumunu öğreniyoruz ardından eğer konumumuz pixel cinsinden 100 pixel'in altındaysa sayfanın başlarındayız demektir baack to topo elemanımızı saklıyoruz eğer 100 pixel den büyük ise komponentimizi tekrardan gösteriyoruz.

4. Header sayfası ve ilk wow js entegrasyonu

Bu componentden itibaren wow js ve animasyon css kütüphanesine ihtiyaç duyarız burada ilk olarak direkt olarak npm install olarak kurarız

Npm install wowjs

Ardından use effect ile did mount anında wow js yi initialize ederiz.

```
function App() {  
  useEffect(() => {  
    new WOW.WOW({  
      live: false  
    }).init();  
  }, [])  
  return (  

```

Ardından public klasörü altındaki index html ye animate .css eklenir.

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/animate.css@3.7.2/animate.min.css">
```

```
<link rel="stylesheet"
```

```
href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.0.0/animate.compat.css"/>
```

Ardından class parametrelerine eklenir

```
<div className="pt-5 text-center row justify-content-center">  
  <h1 className="text-uppercase textShadowwed display-3 shadow-lg wow bounceIn animated" data-wow-delay=".40s">  
    Hami Berkay Aktaş  
  </h1>  
  <h4 className="textShadowwed wow bounceIn animated " data-wow-delay=".50s" >Frontend Developer</h4>  
  <p className="text-center textShadowwed col-md-6 wow bounceIn animated " data-wow-delay=".60s" >
```

Ayrıca bu sayfada soll down animasyonunu eklemek için

Npm i react-lottie-player eklenir

Ardınan component içerisine şu şekilde bir şey eklenir

```
<Player  
  autoplay  
  loop  
  src="https://assets2.lottiefiles.com/packages/lf20_xfge4p6p.json"  
  style={{ height: '100px', width: '100px' }}>  
</Player>
```



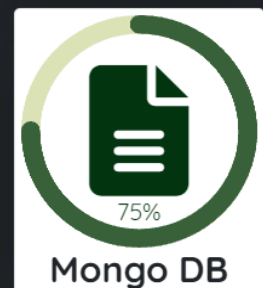
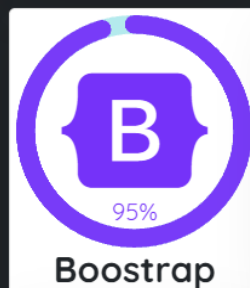
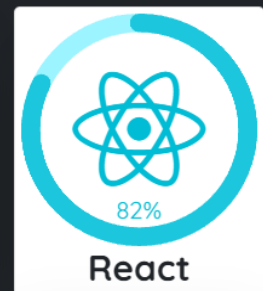
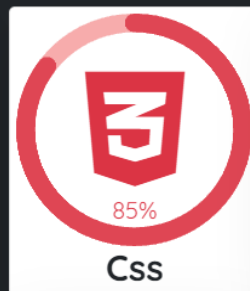
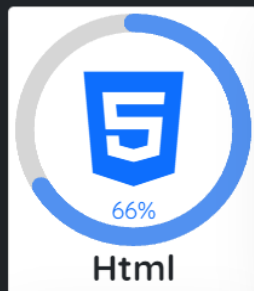
5.Ability Componenti

npm i react-circular-progressbar diyerek dairesel progress bar oluşturmak için kütüphane kurarız

kod yapımız aşağıdaki gibi oldu

```
1 import React from 'react'
2 import 'react-circular-progressbar/dist/styles.css';
3 import { CircularProgressbarWithChildren, buildStyles } from 'react-circular-progressbar';
4
5 function AbilityCard({percentage,icon,colorT,pathcolor,title}) {
6   return (
7     <div className='cardFlush'>
8       <div className="card shadow-lg ghost mb-3 wow bounceIn animated" data-wow-delay=".20s">
9         <div className="card-body p-1">
10
11           <CircularProgressbarWithChildren value={percentage} styles={buildStyles({
12             // Text size
13             // How long animation takes to go from one percentage to another, in seconds
14             pathTransitionDuration: 0.5,
15             // Can specify path transition in more detail, or remove it entirely
16             // pathTransition: 'none'
17             // Colors
18             pathColor: `rgba(${colorT}, ${percentage / 100})`,
19             textColor: '#f88',
20             trailColor: pathcolor,
21             backgroundColor: '#3e98c7',
22           })}>
23             /* Put any JSX content in here that you'd like. It'll be vertically and horizontally centered. */
24             <>
25               <p className="text-center m-0" style={{fontSize: '110px',color: 'rgba('+colorT+')'}}><i className={icon} /></p>
26               <p className="text-center p-0" style={{position:"absolute",bottom:"-5px",color: 'rgba('+colorT+')'}}>>{percentage}%</p>
27             </>
28           </CircularProgressbarWithChildren>
29           <p className="h4 text-center m-0 fw-bold">{title}</p>
30         </div>
31       </div>
32     </div>
33   );
34 }
```

Yetenekler



6.React animated number

Bu komponentde bu güne kadar yaptığım çalışmalardan bilgileri animasyonlu rakamlar ile birlikte renderleriz kod yapısı ve görünüm aşağıdaki gibi olur

npm i react-animated-numbers

```
1  import React from 'react'
2  import AnimatedNumbers from "react-animated-numbers";
3
4  function OurWorkCard({title,count,icon}) {
5    return (
6      <div className="col">
7        <div className="card border-0 bg-dark">
8          <div className="card-body text-center">
9            <h5 className="card-title text-warning fs-1"><i className={icon} /></h5>
10           <span className="card-subtitle mb-2 text-light d-flex justify-content-center">
11             <AnimatedNumbers
12               includeComma
13               animateToNumber={count}>
14             </AnimatedNumbers>
15           </span>
16           <p className="card-text text-orange text-center">{title}</p>
17         </div>
18       </div>
19     </div>
20   )
21 }
22
23 export default OurWorkCard
```



7. Todo Sayfam

Buradaki todo'ları React Redux state yönetim aracı ile yönetiriz.

Kodların içindeki todo slice içeriğinde daha detaylı şekilde todoSlice.js dosyasından erişebiliriz.

Todos



Yapılacaklar

1 Lorem ipsum dolor sit amet



2 Lorem ipsum dolor sit amet



5 Lorem ipsum dolor sit amet



Devam Edenler

3 Lorem ipsum dolor sit amet



Bitirilenler

4 Lorem ipsum dolor sit amet

