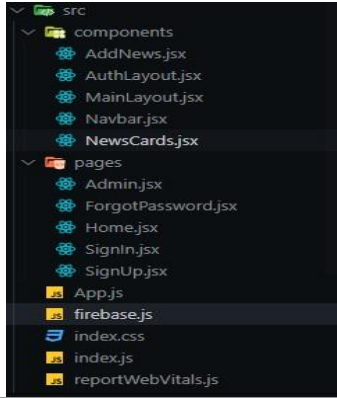


## RE-NEWS (Bitirme Ödevi)



### Proje Oluşturma:

React projemizin kurulumunu yaptıktan sonra kullanacağımız Firebase veritabanına olan bağlantımız için bir JavaScript dosyası oluşturuyoruz.

### Projenin auth ve database bağlantısını:

Firebase veritabanımızdan gelen auth işlemi için “getAuth”, databaseimizi çekmek için “getFirestore” işlemi yaparak bu db yi bir değişkene atıyoruz sonrasında içerisinde ki collectiona(table’a) erişmek için db parametrelili “” içerisinde erişmek istediğimiz collection adını yazıyoruz ve db verileri getiriyoruz.

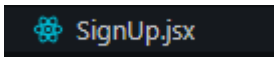
```
// Initialize Firebase
const app = initializeApp(firebaseConfig);

//Authantication işlemi
export const auth = getAuth(app);

//Data
export const db = getFirestore(app);

//Datanın haberler adında ki collectionun çekilmesi
export const ref = collection(db, "haberler");
```

### Kullanıcı oluşturma işlemi :



Auth ve db’yi çekmek için gerekli şeyleri yaptıktan sonra kullanıcı oluşturmak için Sign-Up sayfası açıyoruz. Firebase.js te oluşturduğumuz auth değişkenini bu sayfaya import ediyoruz. Aynı zamanda firebase’den gelen createUser adlı functionumuzuda aynı şekilde sayfamıza import ediyoruz.

```
import App from "../App";
import { useState, useCallBack } from "react";
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
} from "firebase/auth";
import { Link, NavLink, Route, useNavigate } from "react-router-dom";
import { auth } from "../firebase";
import { useAuthState } from "react-firebase-hooks/auth";

//Firebaseden gelen Sign-Up metodu. useStateleri kullanarak inputları
const SignUp = () => {
  const navigate = useNavigate();
  const [user, isLoading] = useAuthState(auth);
  const [email, setEmail] = useState("");
  const [sifre, setSifre] = useState("");

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

      if (!email || !sifre) {
        return;
      }
      //İstedğimiz parametremizi mesela burada username de olabi
      createUserWithEmailAndPassword(auth, email, sifre)
        .then(() => {
          alert("Kullanıcı Kayd Başarılı!");
        })
        .catch((err) => {
          console.log(err);
        });
    },
    [email, sifre]
  );
};
```

Kullanıcı oluşturmak için öncelikle istediğimiz verilerin inputlarını yazıyoruz (email-password). useState kullanarak bu inputlarda gelen değerleri tutuyoruz. Sonrasında bir buton ekleyerek onclick'ine bir function yazıyoruz. Firbase'den gelen bu functionda parametre olarak inputlardan gelen verilerimizi yazıyoruz sonrasında işlem tamamlandıysa bir alert veriyoruz eğer başarısızsa konsola bir error fırlatmasını istiyoruz.

## **SIGN UP SAYFA GÖRÜNÜŞÜ**

### YENİ KULLANICI OLUŞTURUN

E-Mail you@example.com

\*\*\*\*\*

Şifre \*\*\*\*\*

Kayıt Ol

Kaydınız varsa hemen giriş yapın.

## Kullanıcı Giriş İşlemi :

SignIn.jsx

Sign-Up sayfamızda olduğu gibi burada da alacağımız değerler için inputlar oluşturuyoruz ve bu değerleri useStatemizde tutuyoruz. Tutulan değerleri firebase'den gelen Sign-In functionuna parametre olarak yollayarak db içerisinde varlığını kontrol ediyoruz.

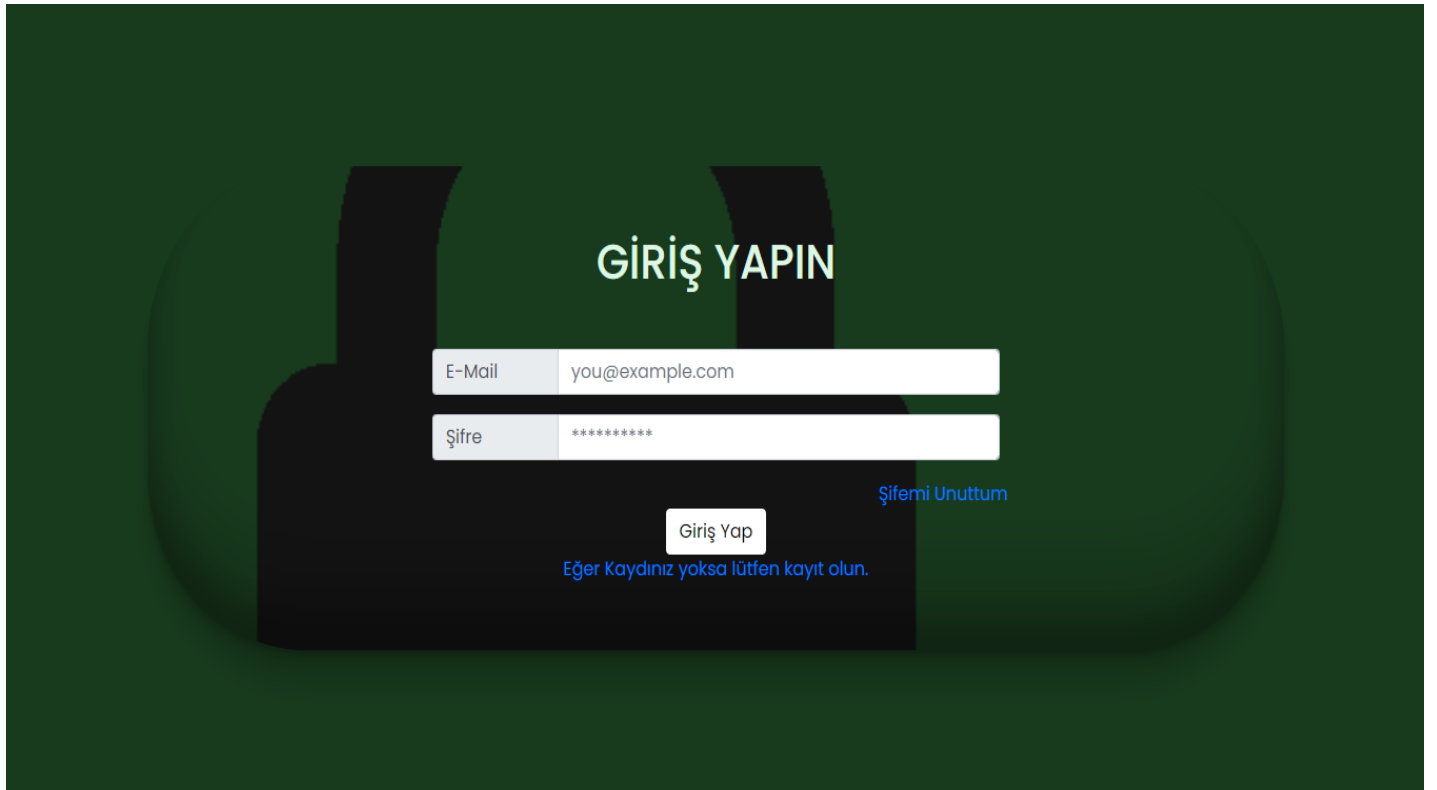
```
const SignIn = () => {
  //firebase'den gelen signin metoduyla beraber statemi
  const [email, setEmail] = useState("");
  const [sifre, setSifre] = useState("");

  const navigate = useNavigate();

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

      if (!email || !sifre) {
        alert("Lütfen boş alan bırakmayınız");
        return;
      }
      //signup metotunda olduğu gibi bu functionda da
      signInWithEmailAndPassword(auth, email, sifre)
        .then(() => {
          alert("Kullanıcı Girişi Başarılı!");
        })
        .catch((err) => {
          console.log(err);
        });
    },
    [email, sifre]
  );
};
```

## SIGN IN SAYFA GÖRÜNÜŞÜ



The image shows a sign-in page with a dark green background. At the top, there is a large, stylized white letter 'A' that serves as a background element. The text 'GİRİŞ YAPIN' is centered in white. Below it, there are two input fields: 'E-Mail' with the value 'you@example.com' and 'Şifre' with the value '\*\*\*\*\*'. To the right of the password field, there is a link that says 'Şifemi Unuttum'. Below the input fields, there is a button labeled 'Giriş Yap'. At the bottom, there is a text link that says 'Eğer Kaydınız yoksa lütfen kayıt olun.'

## Oluşturulan Haberleri(Data) Sayfaya Getirme İşlemi:

NewsCards.jsx

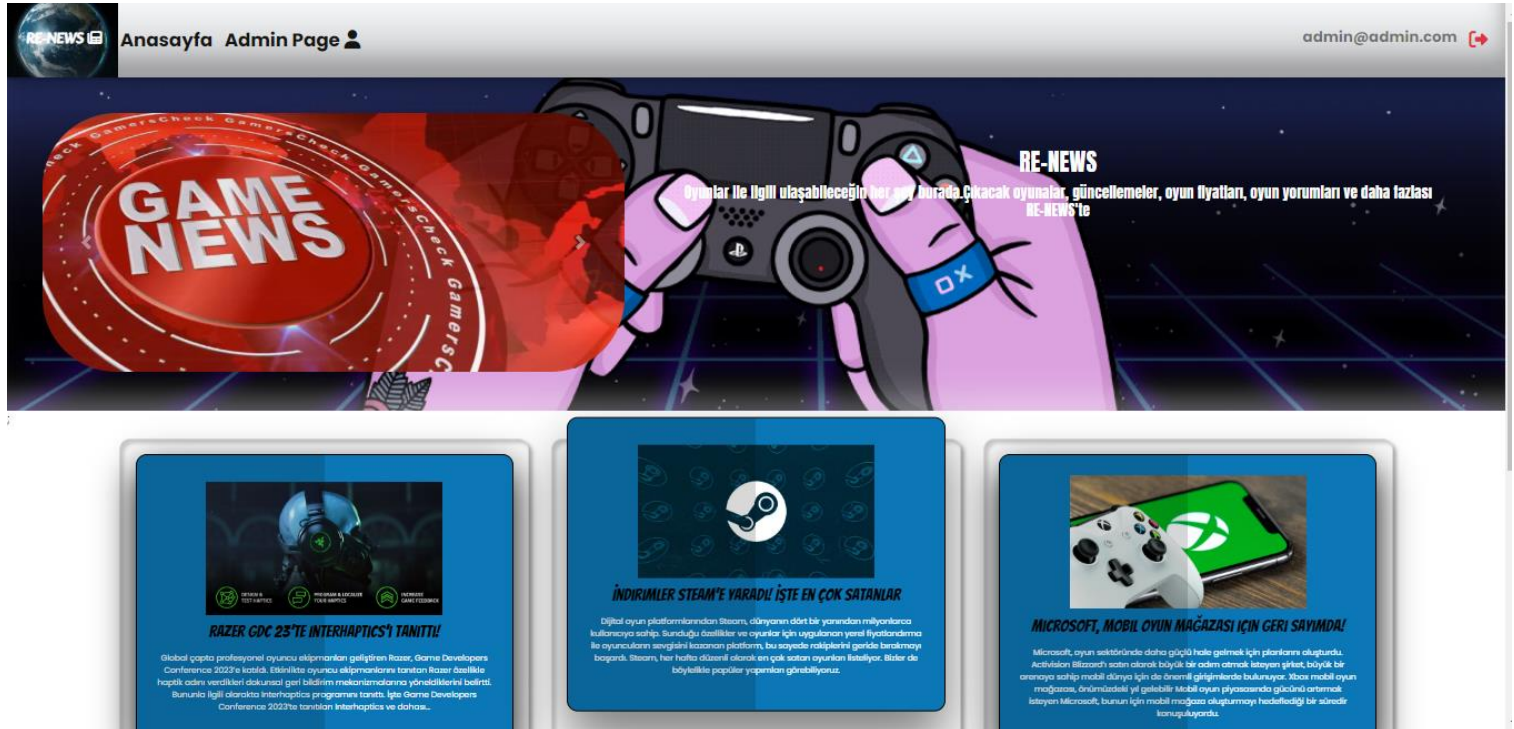
Home Page'imizde fazla kod kalabalığı olmaması adına components klasörü açarak dataları buraya çektim sonrasında bu dosyası Home Page'e aktardık.

Firebase.js dosyamızda **ref** adında oluşturduğumuz değişkende db içerisindeki collectionu zaten çekmiştik. Bu sayfamıza import ederek data adında biz diziye attık. Sonrasında bu datayı mapleyerek cardlarımıza aktardık.

Anasayfaya aktarımdan sonra Navbar, Slider gibi birkaç tasarım işlemi gerçekleştirdik.

```
const NewsCards = () => {  
  const [data] = useCollectionData(ref);  
  
  console.log(data);  
  //datamızı sayfaya import ederek mapledikten  
  return (  
    <div>  
      <div class="container">  
        <div className="row">  
          {data?.map((haber) => (  
            <div className="col-md-4">  
              <div class="card">  
                <div class="box">  
                  <div class="content">  
                    <img  
                      src={haber.resim}  
                      style={{ height: 200 }}  
                      alt=""  
                      className="image-card"  
                    />  
                    <h5>{haber.baslik}</h5>  
                    <p>{haber.aciklama}</p>  
                  </div>  
                </div>  
              </div>  
            </div>  
          )</div>  
        </div>  
      </div>  
    )</div>  
  )</div>  
}
```

## ANASAYFA GÖRÜNÜŞÜ



Hover işleminden dolayı kartımız üstüne gelindiğinde yukarı çıkmakta.

## Admin Sayfası:

Admin sayfamızda da kod kalabalığı olmaması adına AddNews adında bir component oluşturarak ekleme işlemini burada gerçekleştirdik.

Bu sayfada Adminimiz yeni haberler ekleyebilecek. Bunun için inputlar koyduk 3 adet inputumuzu başlık, açıklama ve resim url olarak adlandırdık.

Bu inputlardan gelen değerleri useStatemizde tutarak Firebase'den gelen addDoc işlemiyle collectionumuzda nereye denk geleceğini (karşılıklarını) belirttik. İlk parametre olarak da databaseimizi attığımız “ref” adındaki değişkeni veriyoruz. Bu ref db içerisinde ki haberler adlı collectionu belirtiyor.

```
const AddNews = () => {  
  //inputlardan gelen bverileri statelerimizde utarak sonrasında adddoc adındaki  
  const [aciklama, setAciklama] = useState("");  
  const [baslik, setBaslik] = useState("");  
  const [resim, setResim] = useState("");  
  
  const handleSubmit = useCallback(  
    (e) => {  
      e.preventDefault();  
      addDoc(ref, { aciklama: aciklama, baslik: baslik, resim: resim }).then(  
        () => {  
          alert("Haber kaydetme işlemi başarılı");  
        }  
      );  
    },  
    [aciklama, baslik, resim]  
  );  
};
```

```
addDoc(ref, { aciklama: aciklama, baslik: baslik, resim: resim }).
```

## ADMIN SAYFASI GÖRÜNÜŞÜ

### Hoşgeldin Admin

Buradan Yeni Haberini Ekleyebilirsiniz

## Route İşlemleri:

Kullanıcımızın giriş yapmadan anasayfaya ulaşmasını istemiyoruz bu yüzden kullanıcının olup olmadığının kontrolü için bir AuthLayout.jsx ve MainLayout.jsx dosyaları oluşturarak App.js dosyamızın içerisinde route işlemi oluşturduk. Bu sayede Kullanıcı giriş yapmadıysa otomatik olarak SignIn sayfasına eğer giriş yaptıysa MainLayout.jsx dosyamızda navigate ettiğimiz gibi Home sayfasına yönlendirilecek.

isLoading ile beraberde sayfamız yüklenirken bir spin-load gösteriyoruz.

```
const AuthLayout = () => {  
  const [user, isLoading] = useAuthState(auth);  
  const navigate = useNavigate();  
  if (isLoading) {  
    return (  
      <div className="spin-load text-center d-flex align-items-center justify-content-center">  
        <div className="spinner-grow text-primary" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-secondary" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-success" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-danger" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-warning" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-info" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-light" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
        <div className="spinner-grow text-dark" role="status">  
          <span className="sr-only">Loading...</span>  
        </div>  
      </div>  
    );  
  }  
  //eğer kullanıcı mevcutsa navigate işlemiyle beraber anasayfaya yollama işlemini gerçekleştirir  
  if (user) {  
    return <Navigate to="/" replace />;  
  }  
  return <Outlet />;  
};
```

## APP.JS DOSYAMIZ →

```
function App() {  
  return (  
    <BrowserRouter>  
      { /* <Navbar /> */}  
      <Routes>  
        { /* LAYOUT 1 */}  
        <Route path="/" element={<MainLayout />}>  
          <Route path="/" element={<Home />} />  
          <Route path="/admin" element={<Admin />} />  
        </Route>  
        { /* LAYOUT 2 */}  
        <Route path="/" element={<AuthLayout />}>  
          <Route path="/sign-up" element={<SignUp />} />  
          <Route path="/sign-in" element={<SignIn />} />  
          <Route path="/sign-in/forgot-password" element={<ForgotPassword />} />  
        </Route>  
      </Routes>  
    </BrowserRouter>  
  );  
}
```