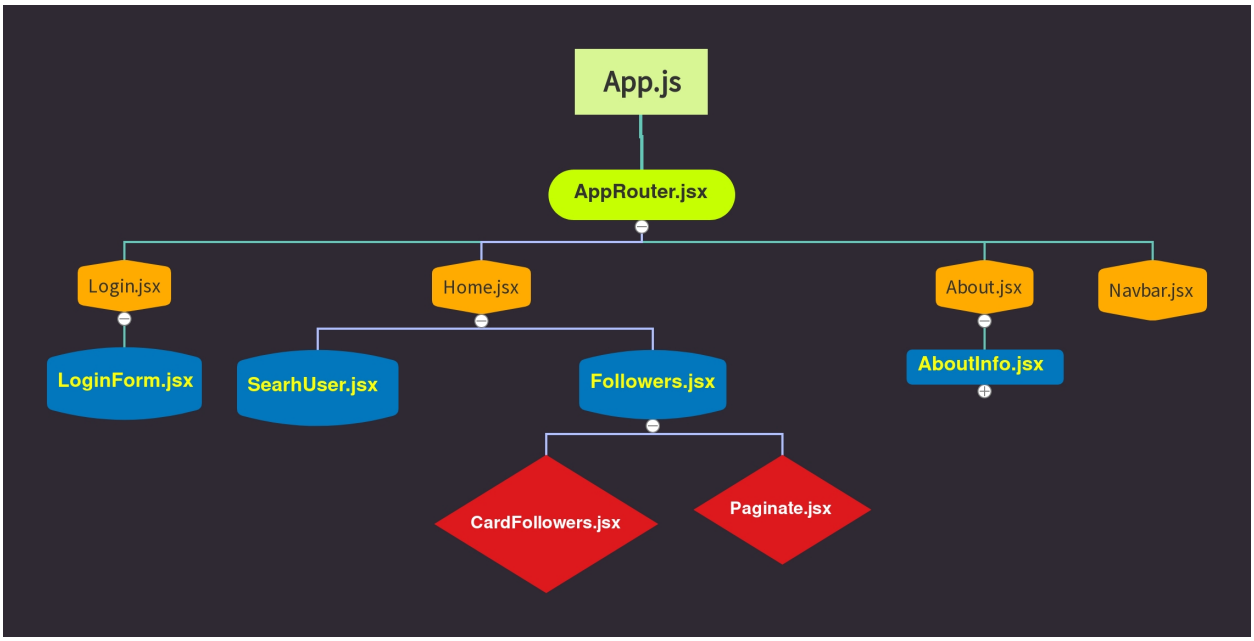
project

Project Tree



Kullanılan Paketler

```
axios,
react-router-dom,
react-bootsrap //index e link veriyoruz
styled-components

npm i axios or yarn add axios//axios ile veri çekeilmek için
npm i react-router-dom or yarn add react-router-dom//router yapısını kullanabilmemiz için
npm i react-bootstrap or yarn add react-bootstrap// style de react-boostrapi kullanabilmek
için
npm i styled-components or yarn add styled-components
```

Proje için kullanılan Api Github Users Api

> https://docs.github.com/en/rest/users

Buradan biz followers kısmını alacağız .Api nin verdği farklı veriler de var

> "https://api.github.com/users/anthonyharold67/followers?per_page=100"

> *per_page default 30 dur sadece 30kişi veriyor maksimumda 100*

React Router

Routing nedir?

```
Yönlendirme, kullanıcıya farklı sayfalar gösterme kapasitesidir. Bu, kullanıcının bir URL
 girerek veya bir öğeye tıklayarak uygulamanın farklı bölümleri arasında hareket edebilece
ği anlamına gelir.
```

Neden react router kullanılıyor ?

React hepimizn bildiği gibi single page bir yapı. Biz bu yapı sayfalar arası gezinebilmek için router yapısını kullanırız.

```
Singe Page Nedir?

Single page application yani kısa adıyla SPA, tek HTML sayfası yükleyen bir uygulamadır ve
uygulamanın çalışması için gerekli tüm dosyaları (JavaScript, CSS vb) içerir. Sayfa veya s
onraki sayfalarla olan herhangi bir etkileşim için servera gidip gelmesi gerektirmez; bu d
a sayfanın yeniden yüklenmediği anlamına gelir.

Reactte SPA oluşturabilmenize rağmen, bu bir zorunluluk değildir. React, hali hazırda çalı
şan bir sitenin küçük bölümlerini geliştirmek için de kullanılabilir. React’te yazılmış ko
d, diğer diller ile de kullanılabilir. Facebook’un sitesi buna en iyi örnektir
```

Yani normal html projelerinde bizim örneğin home.html,about.html,profile.html yapılarımız var.Biz napıyoruz bunları biribirne <a> tagi ile bağlıyoruz. Aslında 3sayfada tek projeye ait ama html yapısından ötürü biz home.html den about.html e gitmek istediğimizde farklı bir sayfaya gidiyoruz.

React ta ise tek bir sayfa içinde sayfa olarak oluşturduğumuz componentler arasında geziniyoruz

Kurulum

```
npm install react-router-dom
# veya
yarn add react-router-dom
```

Kurulum bittikten sonra router yapısını nerede kullanacaksanız ilgili yerde react-router-dom paketinden lazım olan componentlerimizi import ediyoruz. Best practice dediğimiz router ı app.js de değil src klasörü altında router adında bir klasör oluşturup o klasör içinde AppRouter.jsx dosyasını oluşturup router yapısını burada kurup App.js te de sadece AppRouter.jsx componentini çağırıyoruz.

Peki bize react-router-dom paketinden ne lazım Approuter yapısı için?

BrowserRouter: Tanımlanan Yerlere Sayfaların Render Edileceğini Bildiren component. En dış sarmalayıcıdır.

Routes: Konum her değiştiğinde, Routes en iyi eşleşmeyi bulmak için childları olan tüm alt Route öğelerine bakar ve kullanıcı arabiriminin bu dalını oluşturur.

Route: Url pathinde gelene göre hangi sayfanın(yani hangi componentin) render edileceğini belirten component. Ve eğer birden fazla Route varsa bunları Routes sarmalında tanımlamamız gerekir. Yoksa şu şekilde hata alırız:

```
Bir <Route>, yalnızca <Routes> öğesinin alt öğesi olarak kullanılır, hiçbir zaman doğrudan
işlenmez. Lütfen <Route>'unuzu bir <Routes> ile sarın.
```

Bunları react-router-dom pkaetinden import ediyoruz:

```
import {
  BrowserRouter,
  Routes,
  Route
} from 'react-router-dom'
```

NavLink: Geçerli URL ile eşleştiğinde, render edilmiş elemente css ekleyecek bir `<Link>` sürümüdür.

`Navlink` *kullanmak için* `react-router-dom` *dan import etmemiz gerekiyor*

Kısaca Link ve a href farkında bahsedelim

```
Single page application uygulamarında temel mantık aynı sayfada sayfa yenilenmeden istenil
en komponentin sayfaya çağırılmasıdır.

Burada bazı react projelerinde dikkat çeken bir nokta projenin navigasyonunda gezerken say
fanın her linke tıklandığında sayfanın kendini yenilemesi (refresh) dir. Bunun nedeni ise
 linke tıklandığında çağırılan komponentin

<a href="/componentAdi"></a>
şeklinde çağırılmış olmasıdır.

Eğer a etiketi yerine react-dom elementi olan Link etiketi kullanılırsa

<Link to="/componentAdi"></Link>
sayfa yenileme problemi ortadan kalkacaktır.

Bu kullanım sadece React özelinde değil diğer SPA uygulamarı içinde geçerlidir.
```

Projemizdeki AppRouter componentimiz

```
import React from 'react'
import { BrowserRouter, Route,Routes } from 'react-router-dom'
import Navbar from '../components/NavBar'
import About from '../pages/About'
import Home from '../pages/Home'
import Login from '../pages/Login'
//ilgili sayfaları burada import ediyoruz
const AppRouter = () => {
  return (
    <BrowserRouter>
        <Navbar/>//her sayfada olmasını istediğimiz componentleri Routesun dışında tanımla
yarak çaığırıyoruz
      <Routes>
            <Route path="/" element={<Login/>} />
            <Route path="/about" element={<About/>} />
            <Route path="/home" element={<Home/>}  />
      </Routes>
    </BrowserRouter>
  )
```

```
}

export default AppRouter

path=Tarayıcıdan hangi url ile erişeceğimizi belirtir.Yani http://localhost:3000/ adresimi
zin sonuna eklenene göre eşleşiyor
element/component= İlgili path özelliği geldiğinde hangi componenti render edeceğini belir
tir.

//kullanıcı uygulamamızı açtığında ilk karşılayacak sayfa login diye belirtmiş olduk
```

Projemizdeki App.js

```
import './App.css';
import AppRouter from './routers/AppRouter';

function App() {
  return (
    <div className='bg-dark h-100 text-light' style={{minHeight:"100vh"}}>
      <AppRouter/>
    </div>
  );
}

export default App;
```

Navbar componentimizi oluşturalım

Navbar.jsx

Şimdi biz navbar da göstermelik bir yapı kuracağız . Kullanıcı login olduğunda kullanıicnın emailini alıp sessionStorageda tutacağızBuradki email bilgisini sessionStoragedan çekmemiz gerekiyor.Bunun için bir state tanımlayıp eğer session da email bilgisi varsa getir yoksa false bir değer ver dememiz gerekiyorki navbarımızı kullanıcıya ona göre göstersin isityoruz. Diğer türlü kullanıcı login olmadan da navbara tıklayarak sayfalara erişebilir.Tanımladığımız state te göre de return kısmında bir ternary yapısı kurmalıyız. email varsa home,about,logout linkerlnigöster yoksa sadece Calrusway yazan bir navabr göster

Style için react-bootstrap kullanıyoruz .react-bootstrap bize hazır bir yapı sağlıyor gidelim bize uyan bir navbar alalım

https://react-bootstrap.netlify.app/components/navbar/#rb-docs-content

```
import React from 'react'
import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import { NavLink } from 'react-router-dom';

const NavBar = () => {
    const [isOpen,setIsOpen] = React.useState(sessionStorage.getItem('email') || false);
    const handleClick=()=>{
      setIsOpen(false)//navbarı güncellemesi için state i değiştirdik
      sessionStorage.clear()//session storage i temizledik
    }
  return (
    <>
    <Navbar bg="success" variant="dark">
        {isOpen ? (<Container> <Navbar.Brand href="/home">Clarusway</Navbar.Brand>
        <Nav className="me-auto">
          <NavLink to="/home">Home</NavLink>
          <NavLink to="/about">About</NavLink>
          <NavLink to="/" onClick={handleClick}>Logout</NavLink>
        </Nav></Container>): <Container><Navbar.Brand href="">Clarusway</Navbar.Brand></Co
ntainer>}
    </Navbar>
  </>
  )
}

export default NavBar

//&NavLink react router a özgü bir component.a tagi gibi sayfayı tekrardan reload etmemizi
engeller
//email string olduğu için sessionStoragedan dönüştürme yapmadan direk çekebiliyoruz
//aldığımız hazır yapıyla gelen Nav.Link lerin yerine react routerdan NavLink i kullanadık
//Navbar.Brand href kısmını bilerek bıraktım sayfayı reload ettiğini görmek için
//logouta tıkladığımzda da hem logine yönlendirdik hem de sessionStorage ı temizledik
```

AppRouter da tanımladığımız sayfaları oluşturalım.

Login.jsx

Projemizde göstermelik bir login sayfası oluşturacaz. Yani kullanıcıdan bir email ve password alacağız. Kullanıcı submit ettiğinde kullanıcının girmiş olduğu emaili

sessionStoragede tutalım ki kullanıcıyı login olmaya zorlayalım yani login olmazsa navbar daki linkleri göremesin.

Şimdi bize ne lazım email verilerini tutabileceğimiz bir email state i lazım.Kullanıcı submit olduğunda işlem yapacak bir fonksiyon lazım. Ve kullanıcının verileri gireceği bir form yapısı lazım.Bunları yapalım.

burada da isterseniz Login sayfamızda loginform olarak oluşturduğumuz componenti çağıralım

Bu sayfa için react-bootsrap ten gidelim bize uyan hazır bir form yapısı alalım.

https://react-bootstrap.netlify.app/forms/overview/#rb-docs-content

```
import React from 'react'
import LoginForm from '../components/LoginForm'


const Login = () => {
  return (
    <div className='text-light mt-5'>
        <LoginForm/>
    </div>
  )
}

export default Login
```

şimdi gidelim LoginForm u oluşturalım

LoginForm.jsx

```
import React from 'react'
import {Form,Button} from "react-bootstrap"//react-bootstrap bize tagleri component olarak
veriyor ve biz bunları import ediyoruz

const LoginForm = () => {
    const [email, setEmail] = React.useState('')//emaili tutacağımız state

   const handleSubmit = (e) => {
    e.preventDefault();
    sessionStorage.setItem('email', email)//email string olduğu için dönüştürme işlemine g
```

```
erek kalmadan submit olduğunda emailimizi sessionStoragea kaydettik
    window.location.href = '/home';//şuan için kullanışlı olmayan bu metodu kullanıyoruz.b
u metod bize ne yapıyor kullnıcı başarılı bir şekilde submit yaptıktan sonra bizi belirled
iğimiz pathe yönlendiriyor.Bizim Approuter yapımızda bu pathi karşılayıp ilgili componenti
render ediyor.Bu kısımı useNavigate hookunu gördüğünüzde onunla yapabileceksiniz
   }
  return (
    <div className="container">
       <Form onSubmit={handleSubmit}>
            <Form.Group className="mb-3" controlId="formBasicEmail">
                <Form.Label>Email address</Form.Label>
                <Form.Control type="email" placeholder="Enter email" value={email} onChang
e={(e)=>setEmail(e.target.value)} required />
                <Form.Text className="text-muted">
                We'll never share your email with anyone else.
                </Form.Text>
            </Form.Group>

            <Form.Group className="mb-3" controlId="formBasicPassword">
                <Form.Label>Password</Form.Label>
                <Form.Control type="password" placeholder="Password" required />
            </Form.Group>
            <Form.Group className="mb-3" controlId="formBasicCheckbox">
                <Form.Check type="checkbox" label="Check me out" />
            </Form.Group>
            <Button variant="primary" type="submit">
               Submit
            </Button>
        </Form>
    </div>
  )
}

export default LoginForm
//burada react-bootstrap inputları Form.Control olarak veriyorz.biz inputla ilgili işkemle
rimizi bu yapı içinde oluşturabiliyoruz.
//required bize kontrolü sağlamış oluyor kullnıcı veri girişi yapmadan submit işlemini yap
tırmıyor
```

login işlemlerimiz tamam oldu artık login olduktan sonra yönlendireceğimiz Home sayfasını oluşturalım

Home.jsx

Biz projemizde ne istiyoruz.Githubdan takipçilerimizi çeklim.Bu takipçilerimizin verileri gelene kadar ekranda bir gif dönsün. Ve biz kullanıcıların username lerine göre arama

yapabilelim.Biz yazdıkça o bize kullanıcılaı filtrelesin ve ekranda sadece filtrelenen kullanıcıları göstersin.

Github apiden takipçilerimizi çekeceğiz.Bize burada verileri çekeceğimiz axios fnksiyonu lazım.Çektiğimiz verileri tutabileceğimiz state lazım.gelen datayı filtreleme yapabileceğimiz bir input ve bu inputun valuesunu alacağımız bir state lazım. takipçiler i gösterebileceğimiz bir alt component ve search için bir alt component daha lazım.

Filter için ne yapmamız gerekiyor? çünkü biz kullanıcı veri girdiğinde anlık olarka ekrandaki görüntümüz değişsin istiyoruz.Followers componentimize nasıl bir data göndermemiz gerekiyor? Nasıl bir algoritma kurmalıyız ki bize hem aranan veriyi göstercek hem de arama olmadığında tüm veriyi göstercek?

Gelen tüm datayı göndersek filtrelemyi nasıl yapcağız?Yada iki ayrı data mı gönderceğiz alt componente ?Şimdi burada search statetimizin initial değerine boş string verelim ve tüm veride kullanıcıların username ine göre filtreleyelim. yani kullanıcı harf yazar yazmaz isimlere bakması lazım bize string metotlarından hangi metot lazım includes.includes içine gelen veriyle usernameleri karşılaştırıp uyanları yakalaycak bize döndürcek.eğer içi boş olursa ne yapacak tüm veriyi döndürcek o yüzden biz altcomponente filterdan geçen veriyi göndermemiz gerekiyor

```
import React, { useEffect, useState } from 'react'
import Followers from '../components/Followers'
import axios from "axios"
import SearchUser from '../components/SearchUser'


const Home = () => {
    const [allFollowers, setAllFollowers] = useState([])
    const [loading, setLoading] = useState(true)
    const [search,setSearch] = useState("")
    const getFollowers = async() =>{
        const {data} = await axios.get("https://api.github.com/users/anthonyharold67/follo
wers?per_page=100")
        setAllFollowers(data)
        console.log(data);
        // setLoading(false)
    }
    const handleChange = (e)=>{
        console.log(e.target.value);
        setSearch(e.target.value)

    }
    useEffect(() =>{
        getFollowers()
```

```
        setTimeout(() => {
          setLoading(false)
        }, 3000);

    },[])
    const followersList = allFollowers.filter(follower => follower.login.includes(search))
  return (
    <div className='container text-dark mt-3'>
        <SearchUser handleChange={handleChange}/>
        <Followers followers={{loading,followersList}}/>

    </div>
  )
}

export default Home

settimeout kullanmamızı sebebi gif hemen kaybaolmasın
const followersList = allFollowers.filter(follower => follower.login.includes(search))
biz buradan dönen veriyi gönderiyoruz ki alt componente her search işlemine göre veri gitm
iş oluyor
//handleChange fonksiyonunu alt componente gönderiyoruz ki orada search  inputunu tıkladığ
ında çağırıyoruz burada çalıştıryoruz
```

Şimdi Searchuserı oluşturalım

SearchUser

input type ı search kullanalım

```
import React from 'react'
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

const SearchUser = ({handleChange}) => {
  return (
    <div className="row">
        <div className="col-md-4 mx-auto">
        <InputGroup >
        <InputGroup.Text id="inputGroup-sizing-default">
          Search User
        </InputGroup.Text>
        <Form.Control
          type="search"
          name="search" id="search"
          onChange={handleChange}
          aria-describedby="inputGroup-sizing-default"
        />
```

```
      </InputGroup>
        </div>
        </div>
  )
}

export default SearchUser

<div className="row">
<div className="col-md-4 mx-auto">
ortalama yapmak için
```

diğer alt componentimizi oluşturalım

Followers.jsx

Bootstrapten pagination yapısını alıp projemizde kullanalım.Yani bütün takipçilerimizi göstermesin verdiğimiz sayıya göre gösstersin ekranda.bunun için bir Paginate componenti lazım.

pagination linki
https://react-bootstrap.netlify.app/components/pagination/#rb-docs-content

card yapısı için bootstrap linki

https://react-bootstrap.netlify.app/components/cards/

Bize ne lazım sayfada kaç kişi gözüksün istiyorsak onu verebilceğimiz vir state lazım. ve pagination kısmında tıklanan syafayı ayarlayabilceğimiz currentpage state i lazım.

şimdi burada bir de gelen datayı mapleyip gösterebilceğimiz bir alt component lazım. peki biz bu alt componente hangi veriyi mapleyip göndercez paginationdan gelen sayfa sayısına göre datayı manipüle etmemiz lazım. ve manipüle ettiğimiz veriyi Card componentine göndermemiz gerekiyor

```
import { useState } from "react";
import { Row, Container} from "react-bootstrap";
import Paginate from "./Paginate";
import loadingGif from "../assets/loading.gif";
import CardFollowers from "./CardFollowers";

const Followers = ({followers}) => {
    const {loading, followersList} = followers;
```

```
    const [currentPage, setCurrentPage] = useState(1)
    const [followersPerPage] = useState(12)

    const indexOfLastFollower = currentPage * followersPerPage;
    const indexOfFirstFollower = indexOfLastFollower - followersPerPage;
    const currentFollowers = followersList.slice(indexOfFirstFollower, indexOfLastFollowe
r);
    const totalPages = Math.ceil(followersList.length / followersPerPage);

  return (

      <div>
        {loading ?
        <div style={{minHeight:"87vh",maxHeight:"87vh"}}><img src={loadingGif} style={{min
Height:"85vh",maxHeight:"85vh"}} alt=""/> </div>:<div>

        <Row xs={2} md={3} lg={4} className="g-4 mt-4">
          {currentFollowers.map((follower, index) => (
            <div key={index}>
                <CardFollowers follower={follower}/>
            </div>
          ))}
        </Row>

      <div className="d-flex justify-content-center mt-3">
            <Paginate  pages={totalPages} setCurrentPage= {setCurrentPage}/>

      </div>
        </div>
}
      </div>

  );
};
export default Followers;
//paginate componentine totalpages sayısını ve aktif olan sayfayı ayarlayabilmek için setC
urretnPage  metodunu gönderiyoruz
```

```
    const [currentPage, setCurrentPage] = useState(1)
    const [followersPerPage] = useState(12)

    const indexOfLastFollower = currentPage * followersPerPage;
    const indexOfFirstFollower = indexOfLastFollower - followersPerPage;
    const currentFollowers = followersList.slice(indexOfFirstFollower, indexOfLastFollowe
r);
    const totalPages = Math.ceil(followersList.length / followersPerPage);

//currentpage ilk render olduğunda syfa 1.paginate syfasını göstersin istediğimiz için ini
tal olarka 1 verdik
```

```
//biizm veriyi slicelamamız lazım ki sayfa sayfa olarak görebilelim.peki neye göre slice l
ayacaz?
//ensonuncu indexi nasıl bulacaz currentpage ile verdiğimiz sayfa sayısını çarparak last i
ndexi bulacağız.burada slice zaten son indexi kabul etmediği için doğru indexi bulabiliyoru
z.
//başlangıç indexini de last indexten verdiğimiz sayfa sayısını çıkartarak buluyoruz
```

şimdi paginate i oluşturalım

PAginate.jsx

```
import React, { useEffect, useState } from 'react'
import Pagination from 'react-bootstrap/Pagination';

const Paginate = ({pages,setCurrentPage}) => {
    const [activePage,setActivePage]=useState(1)

    const handleSelect = (number) => {
        setActivePage(number)
      }
    let items = [];
    for (let number = 1; number <= pages; number++) {
    items.push(
        <Pagination.Item key={number} active={number === activePage} onClick={()=>handleSe
lect(number)}>
        {number}
        </Pagination.Item>,
    );
    }

    useEffect(() => {
        setCurrentPage(activePage)
    }, [setCurrentPage,activePage])
  return (
    <div>
    <Pagination >{items}</Pagination>
  </div>
  )
}

export default Paginate

//items arrayine Paginationun itemlarını gönderiyoruz
for (let number = 1; number <= pages; number++) {
    items.push(
        <Pagination.Item key={number} active={number === activePage} onClick={()=>handleSe
lect(number)}>
        {number}
```

```
        </Pagination.Item>,
    );
array içine element attığımız için direk syafamızda kullanabiliyoruz
//useEffecte activepage değiştikçe çalıştırıyoruz ve setCurrentPAgee activepagei veriyoruz
//activepage i de her bir itema tıkandığında itemın muberını alıp stattimizi güncelliyoruz
```

diğer alt component

Card.jsx

```
import React from 'react'
import {  Col, Card, Button } from "react-bootstrap";

const CardFollowers = ({follower}) => {
  return (
    <div><Col>
    <Card>
      <Card.Img variant="top" src={follower.avatar_url} />
      <Card.Body>
        <Card.Text>{follower.login}</Card.Text>
        <Button
        href={follower.html_url}
        style={{borderRadius:'50px'}}
        variant="primary">VIEW PROFILE</Button>
      </Card.Body>
    </Card>
  </Col>
  </div>
  )
}

export default CardFollowers
```

About Page

About.jsx

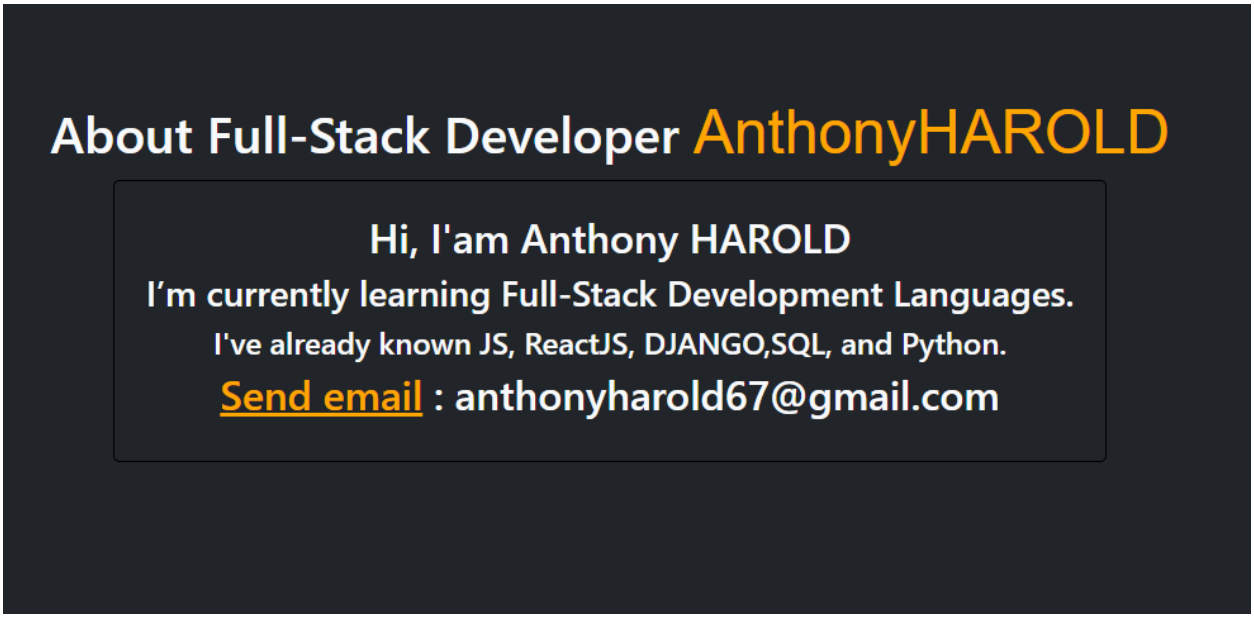
```
import React from 'react'
import AboutInfo from '../components/about/AboutInfo'
```

```
const About = () => {
  return (
    <div>
      <AboutInfo/>
    </div>
  )
}

export default About
```

AboutInfo componentini styled-component kullanark oluşturduk. o nedenle önce AboutStyles.jsx i oluşturalım

AboutStyles.jsx



Bu yapıyı kuracaz bize burada ne lazım Bir kapsayıcı div. Onun altına bir header divi ve header dviini içine de bir span rengini değiştirmek için.Ve bir de infonun olduğu bir div

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

export const AboutContainer = styled.div`

  display: flex;
  flex-wrap: wrap;
```

```
  flex-direction: column;
  justify-content: center;
  align-items: center;
  min-height: calc(100vh - 80px); // tamamen ortalasın diye
  line-height: 2;
  span {
    color: orange;
    font-family: "Girassol", sans-serif;
    font-size: 3rem;
  }
`;

export const InfoContainer = styled.div`
  text-align: center;
  margin: 0 10px;
  max-width: 1000px;
  border: 1px solid black;
  padding: 25px;
  border-radius: 5px;
  a {
    color: orange;
  }
`;

export const HeaderContainer = styled.div`
  text-align: center;
`;
```

şimdi artık oluşturduğum bu styled-componentleri kullanma zamanı

```
import React from "react";
import {
  AboutContainer,
  HeaderContainer,
  InfoContainer,
} from "./AboutStyles";


const AboutInfo = () => {
  return (
    <AboutContainer>

      <HeaderContainer>
        <h1>
          About Full-Stack Developer <span>AnthonyHAROLD </span>
        </h1>
      </HeaderContainer>
      <InfoContainer>
        <h2>Hi, I'am Anthony HAROLD</h2>
```

```
        <h3>I’m currently learning Full-Stack Development Languages.</h3>
        <h4>
          I've already known JS, ReactJS, DJANGO,SQL, and Python.
        </h4>
        <h2>
          <a href="mailto:anthonyharold67@gmail.com">Send email</a> :
          anthonyharold67@gmail.com
        </h2>
      </InfoContainer>

    </AboutContainer>
  );
};

export default AboutInfo;
```