React Routing & Back-End Integration

Gold - Chapter 7 - Topic 2

Selamat datang di Chapter 7 Topik 2 online course Fullstack Web dari Binar Academy!

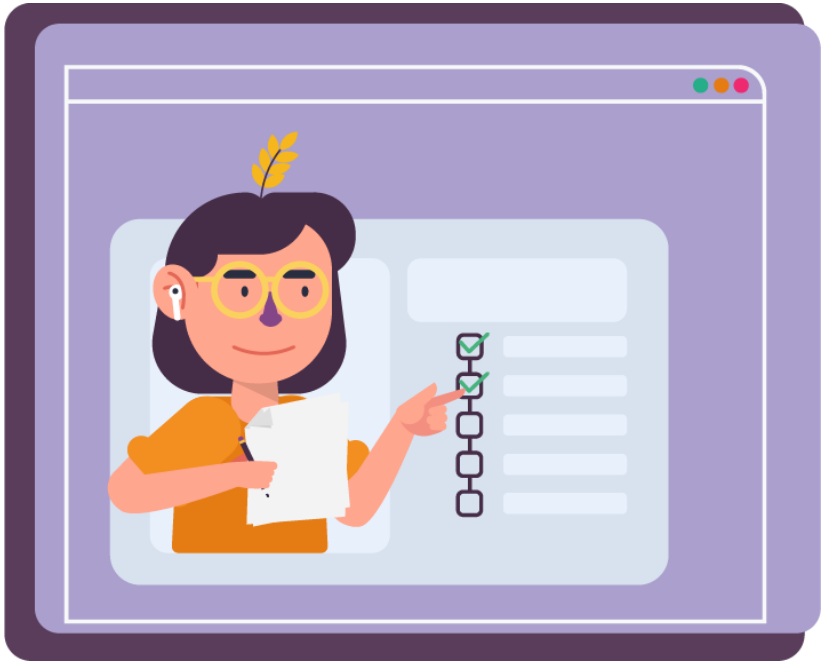
Belajar Full Stack Web bareng Sabrina, selamat datang kembali di Topik 2~

Di chapter 7 ini kamu akan belajar **membuat sebuah aplikasi dengan menggunakan ReactJS dan juga menerapkan OAuth di dalam React dan Express.** Mulai dari pengenalan ReactJs, React Router, Backend Integration, Oauth, dan Redux.

Setelah belajar apa itu ReactJs, di topik ini kita akan **bahas lebih dalam lagi mengenai ReactJs dan gimana cara integrasi backend dalam aplikasi dengan React**. Mari kita lanjutttttttt~

Detailnya, kita bakal bahas hal-hal berikut ini:

- Memahami routing pada React
- Memahami konsep HTTP request
- Konsep dan cara melakukan file processing
- Mengembangkan fitur authentication pada aplikasi React
- Konsep dan implementasi private route



Kalau di topik sebelumnya kita sudah mempelajari aplikasi sederhana dengan React yang hanya memiliki satu halaman saja.

Sekarang kita akan lanjut bahas cara membuat website React yang lebih dari satu halaman, dengan fungsinya masing-masing. Yaitu dengan bantuan routing !

Nah, untuk **membuat beberapa halaman yang akan di-render berdasarkan URL-nya di dalam React, kita bisa menggunakan routing**. Untuk melakukan routing di dalam react, kita bisa menggunakan react-router.

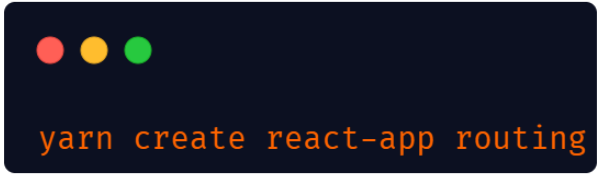
Kalau masih bingung, kita langsung coba praktek aja yuk!

Sekarang kita akan coba untuk membuat 1 halaman sederhana, yaitu Home, dan About:

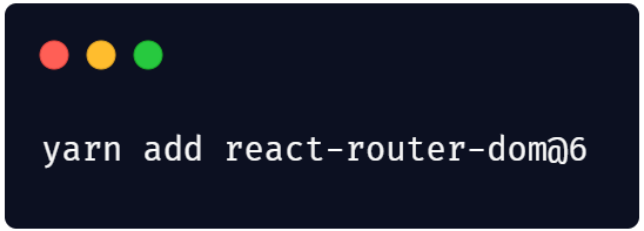
- Halaman Home akan ditampilkan jika user membuka **localhost:3000**
- Halaman About akan ditampilkan jika user membuka **localhost:3000/about**

Oke, langsung kita buat aja dengan mengikuti langkah-langkah berikut.

Langkah pertama, **buatlah sebuah project React** dengan menggunakan CRA.



Lalu, instal **react-router** dengan detail instalasi dapat dilihat di [tautan berikut](#).

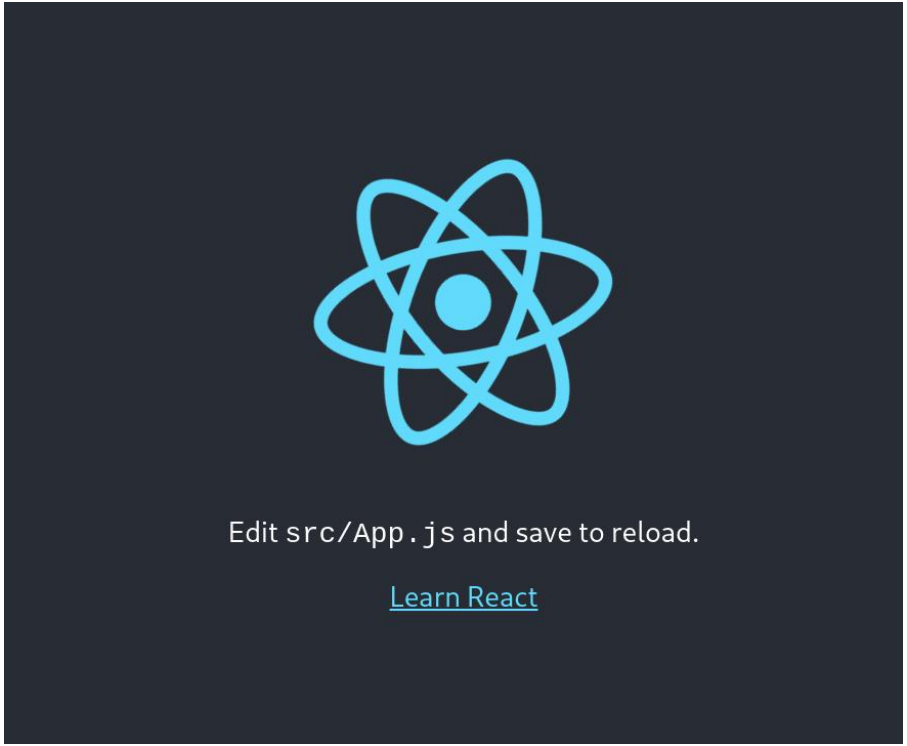


Memasang react-router di src/index.js dengan cara melakukan import **react-router-dom**, dan menggunakan **BrowserRouter**. Saat ini kita hanya punya 1 halaman saja, tapi kita akan menambahkannya lagi nanti.

src/index.js

```
import React from 'react';
import ReactDOM from "react-dom/client";
import {
  BrowserRouter,
  Routes,
  Route,
} from "react-router-dom";
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(
  document.getElementById("root")
);

root.render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<App />} />
    </Routes>
  </BrowserRouter>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Jika kamu jalankan aplikasi react tersebut dan membuka **localhost:3000**, kamu bakal ngeliat basic CRA.



Lalu, kita akan buat halaman lain, yaitu halaman **About**. Kita akan buat halaman simple dengan copy yang ada di **App.js** terus paste di **About.js**.

src/About.js

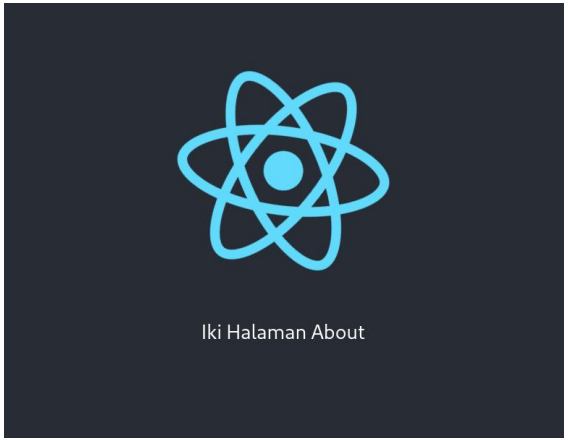
```
import logo from './logo.svg';
import './App.css';

function About() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Iki Halaman About
        </p>
      </header>
    </div>
  );
}

export default About;
```

Lalu, pasang component tersebut ke **src/index.js** dengan kode ini

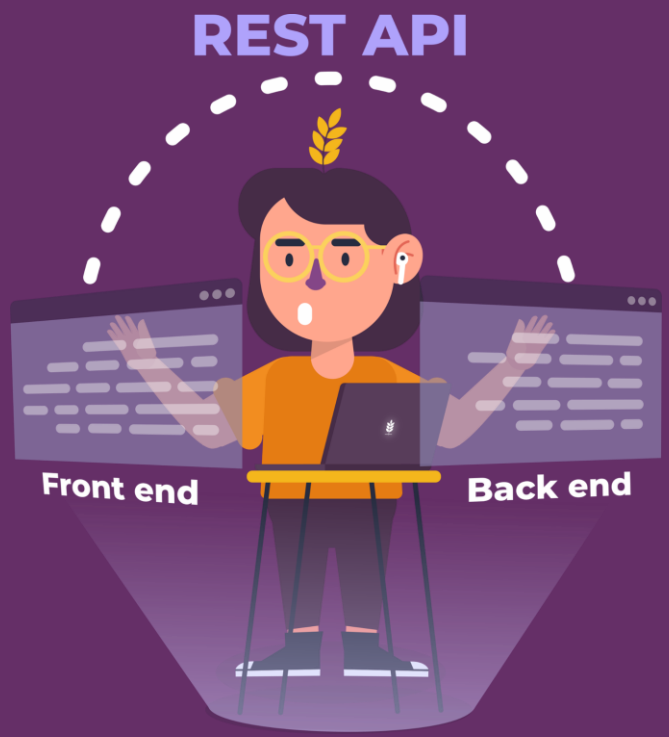
src/index.js

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<App />} />
    <Route path="/about" element={<About />} />
  </Routes>
</BrowserRouter>
```

Jika sudah, buka halaman about dengan membuka **localhost:3000/about**. Maka tampilan yang muncul akan seperti gambar dibawah ini



Nahhh ~ itulah cara routing di React. Repository bisa kamu lihat disini ya.



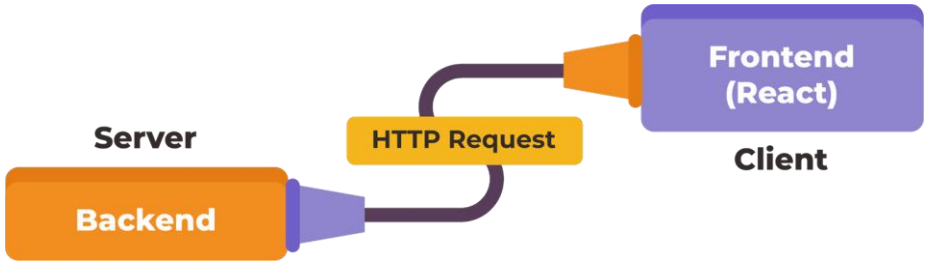
Oke, karena backend dan frontend adalah aplikasi yang terpisah, maka cara untuk mengintegrasikan kedua aplikasi tersebut adalah dengan menggunakan Web API (REST API). Yang mana lingkup API tersebut menggunakan **HTTP Request**.

HTTP Request

Dalam pembahasan kita kali ini konteksnya adalah **backend sebagai server, dan frontend (React) sebagai client**.

Kalau mereka berdua berkomunikasi menggunakan HTTP Request, maka **React merupakan HTTP Client**. HTTP Client ini bisa jadi keyword kalian dalam mencari library dan solusi dari masalah-masalah yang mungkin nanti bakal kalian temukan.

Tapi pada materi ini, **kita akan menggunakan fetch sebagai library yang akan kita gunakan untuk melakukan HTTP Request.**



Lalu modifikasi file **src/App.js**.

src/App.js

```
import { useState, useEffect } from "react";
import logo from "./logo.svg";
import "./App.css";

function getRandomInt(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min) + min); //The maximum
is exclusive and the minimum is inclusive
}

const id = getRandomInt(1, 100);

function App() {
  // https://jsonplaceholder.typicode.com/todos/:id
  const [todo, setTodo] = useState(null);
  const style = { textDecoration: todo?.completed ? "line-through" :
"unset" };

  useEffect(() => {
    fetch(`https://jsonplaceholder.typicode.com/todos/${id}`)
      .then((response) => response.json())
      .then((data) => setTodo(data))
      .catch((err) => console.error(err));
  }, []);

  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        {!!todo ? (
          <div>
            <h1 style={style}>{todo.title}</h1>
          </div>
        ) : (
          <div>
            <h1>Loading ...</h1>
          </div>
        )}
      </header>
    </div>
  );
}

export default App;
```

Nahhh, sampai tahap ini kamu sudah tahu gimana cara melakukan **GET** request. Untuk sisanya kamu boleh eksplor sendiri. Bagaimana cara melakukan POST, DELETE, dan PUT request, bagaimana cara mengirim JSON request, bagaimana cara mengirim header dan sebagainya.

Code implementasi bisa dilihat disini.

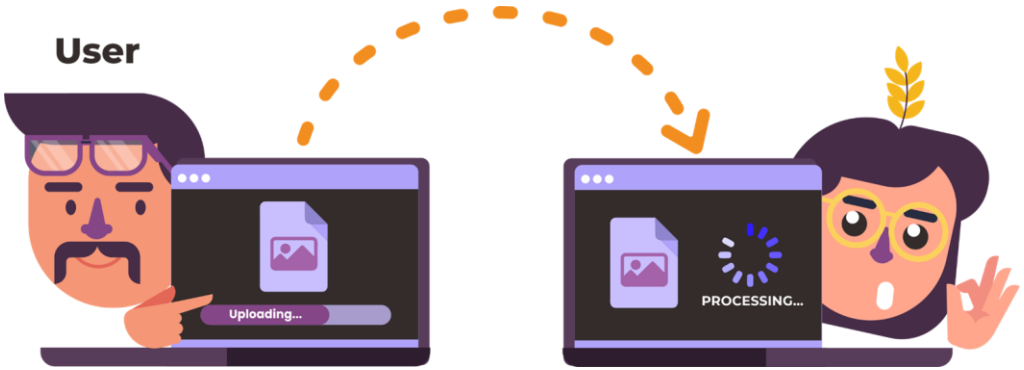
Setelah memahami konsep HTTP Request, pada pembahasan selanjutnya kita akan membahas file processing, authentication dan private route. Letsgooo~

File Processing

File processing mampu **memproses sebuah file (image) yang di-input oleh user melalui sebuah form dan menampilkannya di dalam aplikasi react**.

Untuk memproses file dan menampilkannya secara langsung, kita perlu menggunakan **FileReader**. **FileReader** ini digunakan untuk membaca file dari HTML dan mengolahnya sesuai kebutuhan.

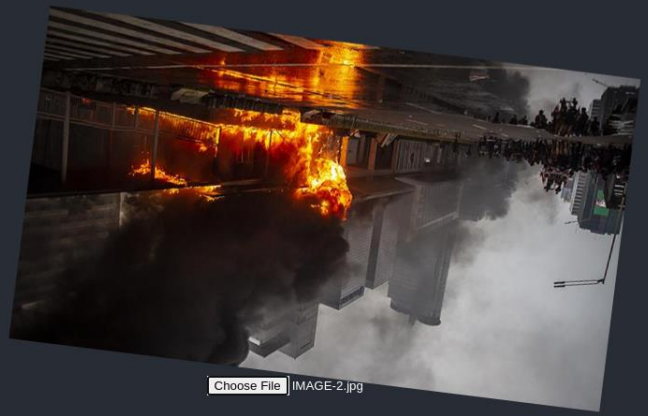
Nah, karena file ini hanya dapat dibaca ketika UI sudah di-render di dalam browser, maka kita perlu menggunakan react ref, dan menempelkannya ke file input.



Jika kamu jalankan code di samping dan kamu buka aplikasi react-nya, maka akan ter-render satu file input. Dimana ketika kamu ganti, image yang ada di dalam UI akan berubah langsung. Code di samping dapat kamu lihat detail-nya disini.

```
import { useState, useRef } from "react"
import logo from './logo.svg';
import './App.css';

function App() {
  const [imageFile, setImageFile] = useState(logo);
  const fileRef = useRef();

  function handleChange() {
    const image = fileRef.current.files[0];
    const reader = new FileReader()

    reader.addEventListener('load', () => {
      setImageFile(reader.result)
    })

    reader.readAsDataURL(image)
  }

  return (
    <div className="App">
      <header className="App-header">
        <img src={imageFile} className="App-logo" alt="logo" />
        <input ref={fileRef} type="file" placeholder="Gambar" onChange={handleChange}></input>
      </header>
    </div>
  );
}

export default App;
```

Tada! Hasilnya seperti ini~

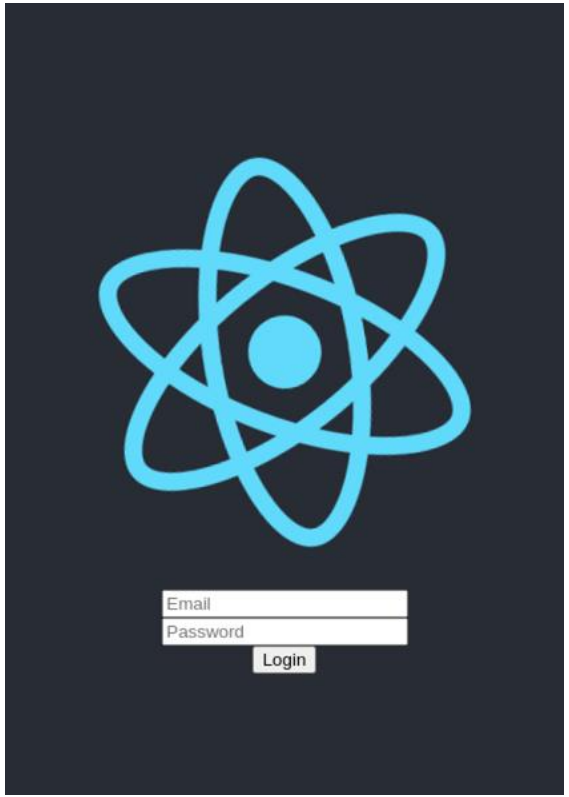


Choose File IMAGE-2.jpg

Authentication

Melalui authentication kita bisa membuat fitur login di aplikasi react yang menggunakan token based authentication.

Implementasi authentication di dalam react cukup sederhana, kamu hanya perlu melakukan hit endpoint **login**, dan menyimpan token-nya di dalam **localStorage**.



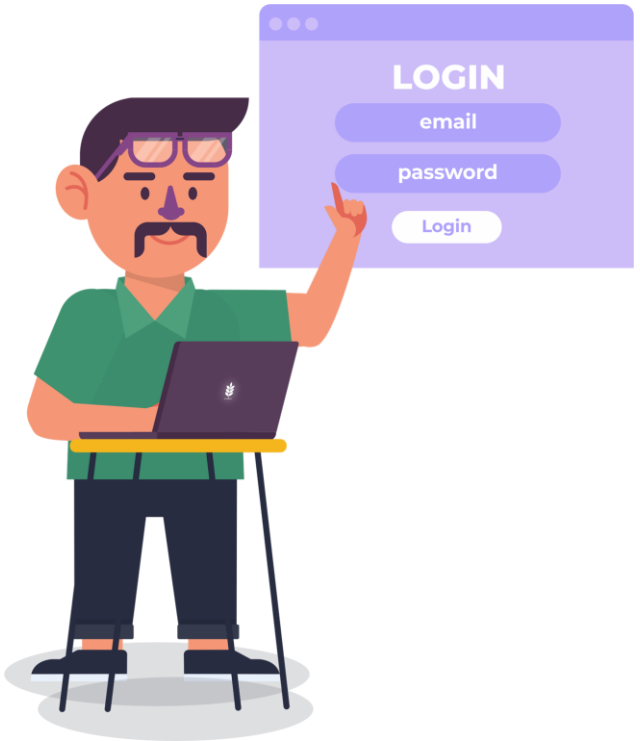
Code di samping dapat kalian akses disini. Untuk backend-nya bisa kalian lihat disini. Jangan lupa nyalain backend-nya dulu sebelum nyobain frontend-nya.

```
import { useState, useEffect } from "react"
import logo from './logo.svg';
import './App.css';

async function doLogin({ email, password }) {
  // Gunakan endpoint-mu sendiri
  const response = await fetch("http://localhost:3001/api/v1/auth/login", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      email,
      password,
    })
  });
  const data = await response.json();
  return data.token;
}

function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [isLoading, setIsLoading] = useState(false);
  const token = localStorage.getItem("token")

  useEffect(() => {
    setIsLoggedIn(!!token);
  }, [token])

  function handleSubmit(e) {
    setIsLoading(true);
    e.preventDefault();
    doLogin({ email, password })
      .then((token) => localStorage.setItem("token", token))
      .catch((err) => console.log(err.message))
      .finally(() => setIsLoading(false))
  }

  function handleLogout(e) {
    setIsLoading(true);
    e.preventDefault();
    localStorage.removeItem("token");
    setIsLoggedIn(false);
    setIsLoading(false);
  }

  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />

        {!isLoggedIn ? (
          <form onSubmit={handleSubmit}>
            <input type="email" placeholder="Email" onChange={(e) => setEmail(e.target.value)} value={email}
/>

            <input type="password" placeholder="Password" onChange={(e) => setPassword(e.target.value)}
value={password} />
            <input type="submit" value={isLoading ? "Loading" : "Login"} />
          </form>
        ) : (
          <input type="submit" value="Logout" onClick={handleLogout} />
        )}
      </header>
    </div>
  );
}

export default Login;
```

Private Route

Melalui Private Route kita mampu membuat sebuah halaman yang hanya dapat diakses oleh user yang sudah login saja.

Nah, kalo kamu perhatikan, dari website-website yang ada, ada beberapa halaman yang ga bisa diakses kalau user belom login.

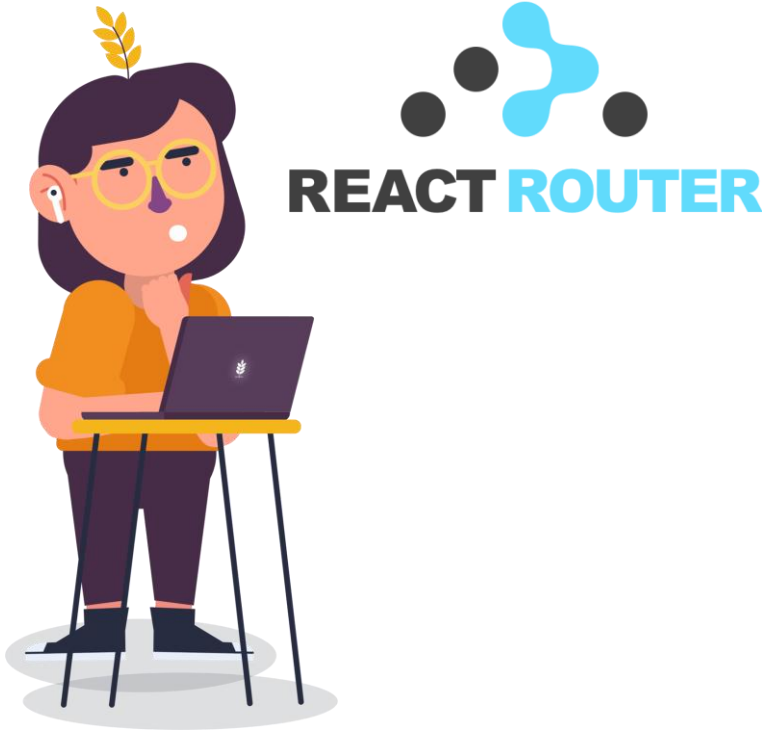


Kalau kita bicara ini di dalam express.js, tentu kamu pasti bakal mikir, “Harus pake middleware nih.", kamu bener.

Tapi kalo di react ini beda cerita, karena di dalam React kita ga pake page based system atau endpoint things, tapi kita pake **router** di dalam react.

Jadi untuk membuat private route, kamu cukup membuat custom component yang akan melakukan cek apakah user udah login atau belum berdasarkan data di dalam **localStorage**.

Kamu bisa langsung lihat repositorynya disini ya!



Referensi buat tambahan kamu belajar ~

- https://reactrouter.com/docs/en/v6/hooks/use-navigate
- https://dev.to/nilanth/how-to-create-public-and-private-routes-using-react-router-72m
- https://jasonwatmore.com/post/2020/01/27/react-fetch-http-get-request-examples
- https://reactjs.org/docs/hooks-reference.html

Terima Kasih!

Next Topic

loading...

BINAR