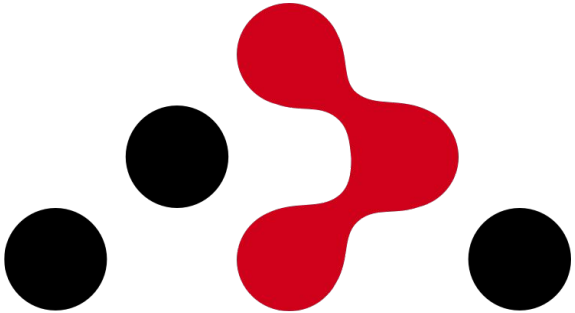
Front End React Development
sesi 15

React Router

Sesi 15 | React Router

Seperti pada aplikasi web pada umumnya, kita akan membutuhkan fasilitas router untuk melakukan routing. Routing ini bisa berupa tombol atau link, sesuai bagaimana temen-temen menerapkan stylingnya. Pada sesi ini, kita akan implementasikan fungsionalitas nya.



Instalasi
Bagian ini terbilang sangat mudah. Silakan buka terminal kamu dan eksekusi perintah berikut :

```
> npm install react-router-dom
```

HACKTIV8

Sesi 15 | React Router

PRIMARY COMPONENTS

Ada 3 component utama dalam meng-implementasikan React Routers :

1. Router, yaitu <BrowserRouter> dan <HashRouter>
2. Route, di antaranya adalah <Route> dan <Switch>
3. Navigation, di antaranya adalah <Link>, <NavLink>, dan <Redirect>

Semua component-component itu dapat temen-temen pakai pada aplikasi berbasis web dengan melakukan import dari “react-router-dom”. Konsep yang akan dipakai di sini adalah dengan melihat Navigation component sebagai “perubah router”

BrowserRouter atau HashRouter harus menjadi element paling atas untuk membungkus App **ATAU** content dari App

Mari kita ikuti step-by-step nya pada slide-slide selanjutnya

Sesi 15 | React Router

Kita awali dengan membuat sebuah React app baru dengan perintah create-react-app. Setelah itu, kita lanjutkan dengan instalasi react-router-dom. Nah, setelah itu, mari kita ikuti langkah-langkah berikut:

1. Implementasi Router

Router yang kita pakai kali ini adalah BrowserRouter. Router ini akan kita simpan sebagai component paling atas dalam App kita, agar semua yang di dalam nya akan menjadi children dari Router dan dapat kita implementasikan fitur-fitur react router

```
import {
  BrowserRouter,
  Switch,
  Route,
  Link
} from "react-router-dom";

function App() {
  return (
    <BrowserRouter>


    </BrowserRouter>
  );
}

export default App;
```

Sesi 15 | React Router

2. Implementasi Route Matchers

Kita akan implementasi Switch dan Route, dimana kombinasi keduanya ini digunakan ketika Switch di render untuk mencari Route mana yang cocok dengan URL. Setelah ketemu, maka kontek dari Route yang cocok akan di render. Dalam hal ini, kita akan render component yang sesuai dengan URL

Switch akan mencari path yg cocok PERTAMA dan langsung melakukan render. Oleh sebab itu, Route yang global biasanya disimpan lebih bawah daripada yang spesifik

Pada contoh di samping, kita menggunakan asumsi kalau component Home, About, dan Contact sudah disediakan

```
8   function App() {
9     return (
10      <BrowserRouter>
11        <Switch>
12          <Route path="/about">
13            <About />
14          </Route>
15          <Route path="/contact">
16            <Contact />
17          </Route>
18          <Route path="/">
19            <Home />
20          </Route>
21        </Switch>
22      </BrowserRouter>
23    );
24  }
```

Sesi 15 | React Router

3. Implementasi Navigation (Route Changers)

Kita biasa menyebut Navigation ini sebagai link. Dan kebetulan, kita akan memakai component Link dalam implementasinya. Dan kita tetap harus implementasi Navigation ini di dalam component BrowserRouter. Link ini akan di render sebagai anchor tag <a>

Sebenarnya ada 1 component lain, yaitu NavLink. Kelebihan nya adalah kita bisa mendefinisikan sebuah class tambahan ke dalam DOM kita, sekiranya NavLink tersebut sedang “active” atau kita sedang berada dalam “page” yang sesuai dengan NavLink tersebut

```
8   function App() {
9     return (
10      <BrowserRouter>
11        <Link to="/">Home</Link> |
12        <Link to="/about">About</Link> |
13        <Link to="/contact">Contact</Link>
14
15        <Switch>
16          <Route path="/about">
17            <About />
18          </Route>
19          <Route path="/contact">
20            <Contact />
21          </Route>
22          <Route path="/">
23            <Home />
24          </Route>
25        </Switch>
26      </BrowserRouter>
27    );
28  }
```

Sesi 15 | React Router

URL Parameters

Seringkali kita membuat halaman yang dinamis menggunakan parameter. Misal untuk halaman profil anggota team, halaman detail comment sebuah group, dll. Kita bisa manfaatkan fasilitas pembacaan parameters dengan hook yang bernama **useParams**

1. Update import

Mari kita import hook **useParams**

```
import {
  BrowserRouter,
  Switch,
  Route,
  Link,
  useParams
} from "react-router-dom";
```

2. Implementasi Navigation

Silakan temen-temen berikan Link seperti berikut ada lokasi yang temen-temen inginkan

```
        <Link to="/about/fulan">About Fulan</Link> | 
        <Link to="/about/fulana">About Fulan</Link> | 
```

HACKTIV8

Sesi 15 | React Router

URL Parameters

3. Implementasi Route Matchers

Dengan menggunakan Switch dan Route. Di sini kita update Route Matchers untuk “/about” dan memberitahukan component mana yang akan menjadi children dari component About

```
38        <Route path="/about/:name" children={<About />} />
```

4. Implementasi Hook

Mari kita update component About kita untuk melakukan implementasi hook **useParams**

```
15  function About() {
16    let { name } = useParams()
17
18    return(
19      <h1>About {name}</h1>
20    )
21  }
```

HACKTIV8

Sesi 15 | React Router

Nesting

Bagaimana jika kita punya sebuah halaman, yang memiliki Link-Link tersendiri ? Kita bisa melakukan implementasi Nesting, dengan hook yang bernama **useRouteMatch**

1. Update import

Mari kita import hook **useRouteMatch**

```
1  import {
2    BrowserRouter,
3    Switch,
4    Route,
5    Link,
6    useParams,
7    useRouteMatch
8  } from "react-router-dom";
```

2. Implementasi Navigation

Kita update component App untuk link About menjadi component baru bernama Abouts, dimana masing2 nama akan tetap dimunculkan pada component About

```
57    <Link to="/abouts">Abouts</Link> | 
```

3. Implementasi Route Matchers

Route matchers di component App menjadi seperti berikut

```
61      <Route path="/abouts">
62        <Abouts />
63      </Route>
```

HACKTIV8

Sesi 15 | React Router

Nesting

4. Implementasi Hook useRouteMatch

Mari kita implementasi hook **useRouteMatch** pada component Abouts. Component About yang pernah kita buat sebelumnya dapat kita jadikan **nesting** dari component Abouts ini

```
24  function Abouts() {
25    let { path, url } = useRouteMatch()
26
27    return(
28      <>
29        <h1>Who do you want to see ?</h1>
30        <ul>
31          <li><Link to={`${url}/fulan`}>Fulan</Link></li>
32          <li><Link to={`${url}/fulana`}>Fulana</Link></li>
33        </ul>
34
35        <Switch>
36          <Route exact path={path}>
37            <h3>Please select a name.</h3>
38          </Route>
39          <Route path={`${path}/:name`}>
40            <About />
41          </Route>
42        </Switch>
43      </>
44    )
45  }
```

Sesi 15 | React Router

Nesting

5. Implementasi Hook useParams

Nah, kita implementasi **useParams** pada component About yang merupakan nested component dari component Abouts

```
16  function About() {
17    let { name } = useParams()
18
19    return(
20      <h3>Hallo {name}</h3>
21    )
22  }
23
```

Bonus : fetch data from API

Lakukan fetch data API pada component About, dengan memanfaatkan hook **useState** dan **useEffect**

```
18  function About() {
19    let { id } = useParams()
20
21    const [user, setUser] = useState([])
22
23    useEffect(() => {
24      fetch(`https://jsonplaceholder.typicode.com/users/${id}`)
25        .then((response) => response.json())
26        .then((data) => setUser(data))
27    })
28
29    return(
30      <>
31        <h3>Hallo {user.name}</h3>
32        <p>{user.company.name}</p>
33      </>
34    )
35  }
```

Sesi 15 | React Router

Redirect

Sesuai dengan namanya, Redirect ini adalah fasilitas dari React Router untuk melakukan redirection. Redirection ini dalam arti sederhananya adalah pemindahan routing dari suatu route ke route yang lain.

Berikut adalah contoh routing sederhana untuk melakukan Redirect dari route “/” ke “/home”. Redirect ini akan selalu di eksekusi pada setiap akses ke route “/”

```
60          <Switch>
61            <Route path="/abouts">
62              <Abouts />
63            </Route>
64            <Route path="/contact">
65              <Contact />
66            </Route>
67            <Route path="/home">
68              <Home />
69            </Route>
70            <Redirect exact from="/" to="/home" />
71          </Switch>
```

Sesi 15 | React Router

Redirect untuk route Root

Ada kalanya kita ingin agar route “/” mengarahkan kita sesuai dengan kondisi otorisasi : apakah kita sudah login atau belum. Biasanya kita akan di redirect ke sebuah route “dashboard” jika sudah loggedIn

```
<Route exact path="/">
  {loggedIn ? <Redirect to="/dashboard" /> : <PublicHomePage />}
</Route>
```

Redirect akibat authentication

Nah, biasanya kita pun membutuhkan fungsionalitas redirection untuk halaman yang hanya dapat diakses oleh authorized user (user yang sudah login)

```
48  function ProtectedPage() {
49    let auth = auth.signedIn()
50
51    if (!auth) {
52      return(
53        <Redirect to="/" />
54      )
55    }
56
57    return(
58      <h1>Contact</h1>
59    )
60  }
```

Sesi 15 | React Router

Redirect dengan Object sebagai option

Redirect dilengkapi dengan kemampuan untuk menggunakan Object sebagai option, jadi, tidak harus berupa String

```
48  function ProtectedPage() {
49    let auth = auth.signedIn()
50
51    if (!auth) {
52      return(
53        <Redirect to={{
54          pathname: "/login",
55          state: { from: location }
56        }} />
57      )
58    }
59
60    return(
61      <h1>Contact</h1>
62    )
63  }
```

Sesi 15 | React Router

Redirect programatically

Ada saatnya kita membuat sebuah method / function dan pada akhirnya, kita membutuhkan fasilitas Redirect ke halaman lain sebagai bagian dari flow aplikasi

Pada Class Component kita bisa memakai method **withRouter**

Pada Functional Component kita bisa memakai hook yang bernama **useHistory.** Pada contoh di samping, kita hanya akan mendemokan penggunaan Functional Component

```
1  import { useHistory } from 'react-router-dom';
2
3  export default function Header() {
4    const history = useHistory()
5
6    const logout = () => {
7      // proses logout di sini, lanjut ke baris 8
8      history.push("/")
9    }
```

Sesi 15 | React Router

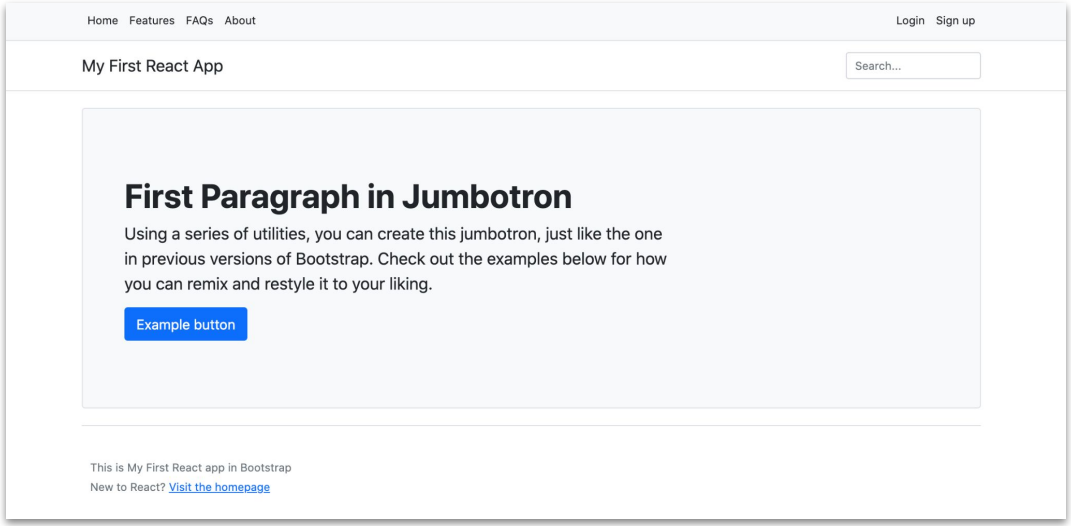
Navlink, digunakan untuk memberikan opsi class tambahan pada link yang page nya sedang aktif

```jsx
import React from 'react'
import {
  NavLink
} from "react-router-dom";

const Header = () => {
  return (
    <header className="d-flex flex-wrap justify-content-center py-3 border-bottom">
      <a href="/" className="d-flex align-items-center mb-3 mb-md-0 me-md-auto text-dark text-decoration-none">
        <span className="fs-4">Harry Potter</span>
      </a>

      <ul className="nav nav-pills">
        <li className="nav-item">
          <NavLink to="/home" className="nav-link" activeClassName="active">Home</NavLink>
        </li>
        <li className="nav-item">
          <NavLink to="/favorites" className="nav-link" activeClassName="active">Favorites</NavLink>
        </li>
      </ul>
    </header>
  )
}

export default Header
```

Sesi 15 | React Router

Implementasi React Router pada First React App kita

Masih ingat dengan aplikasi React pertama kita ? Mari kita lengkapi dengan memasang React Router pada masing2 link



HACKTIV8

Sesi 15 | React Router

Implementasi React Router pada First React App kita

Sebelum lanjut, mari kita rancang dulu fitur-fitur aplikasi kita agar update kita terarah ke depannya.

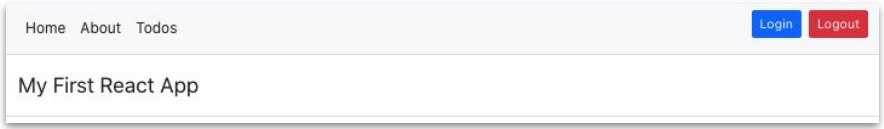
- Halaman public hanya ada 2 : **Home** dan **About**.
- Tambahkan 1 halaman restricted : **Todos**, yang berisi daftar todo yang diambil dari https://jsonplaceholder.typicode.com/todos , tampilkan berupa table. Link ke halaman ini akan selalu muncul bersama 2 Link halaman Public
- Jika user tidak login dan berusaha mengakses halaman **Todos** maka akan di Redirect ke halaman Login
- Halaman Login hanya akan ada 1 buah tombol login yang hanya akan merubah sebuah variable di local storage menjadi TRUE
- Link Logout akan muncul sebagai pengganti link Login, jika user telah login dan akan kembali berganti menjadi link Login jika user logout atau belum login
- Menggunakan **localStorage** sebagai bantuan untuk simulasi aktifitas Login dan Logout

Sesi 15 | React Router

Implementasi React Router pada First React App kita

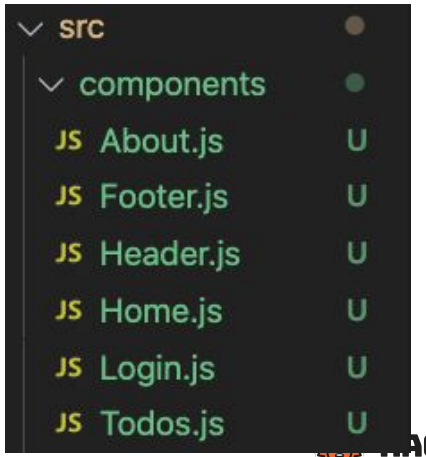
1. Update Top Menu

Update top menu (header) aplikasinya menjadi seperti di bawah ini. Untuk tombol Login kita akan pakai component Link, sedangkan untuk tombol Logout kita akan memakai tombol biasa (<button>)



2. Tambahkan komponen-komponen baru

Tambahkan komponen-komponen baru berupa file kosong sehingga folder components kita akan berisi seperti di bawah ini



Sesi 15 | React Router

Implementasi React Router pada First React App kita

3. Update App.js - import

Update bagian awal file App.js kita yang berisi seluruh import hingga menjadi seperti di samping

```
import 'bootstrap/dist/css/bootstrap.min.css';
import {
  BrowserRouter,
  Switch,
  Route
} from 'react-router-dom';
import Header from './components/Header';
import Home from './components/Home';
import About from './components/About';
import Todos from './components/Todos';
import Login from './components/Login';
import Footer from './components/Footer';
```

Sesi 15 | React Router

Implementasi React Router pada First React App kita

4. Update App.js - return

Update bagian return di functional component file App.js kita hingga menjadi seperti di samping

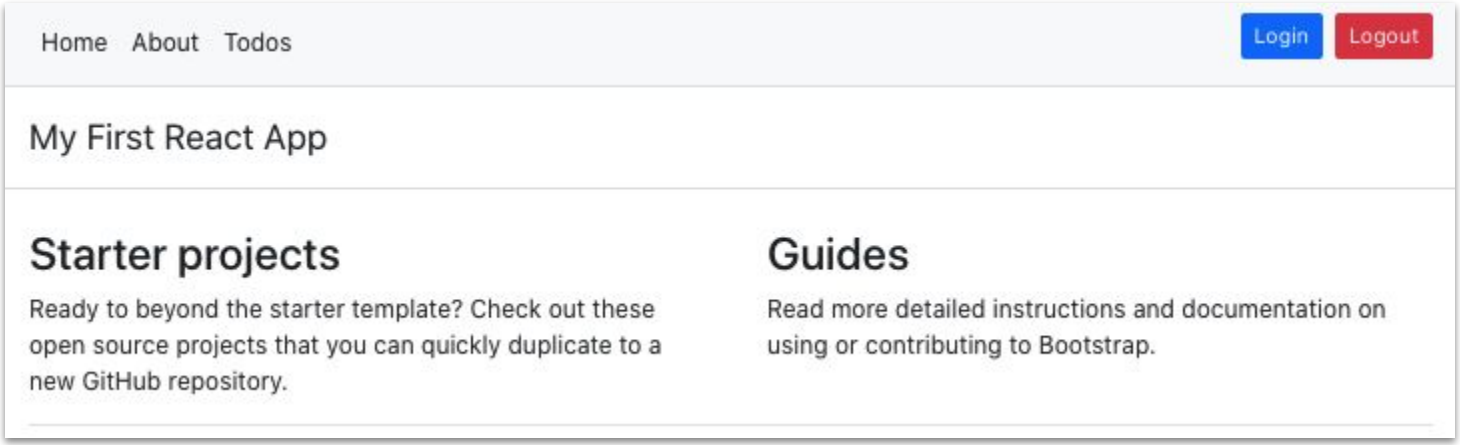
```
14  function App() {
15    return (
16      <BrowserRouter>
17        <Header />
18        <div className="container">
19          <Switch>
20            <Route path="/about">
21              <About />
22            </Route>
23            <Route path="/todos">
24              <Todos />
25            </Route>
26            <Route path="/login">
27              <Login />
28            </Route>
29            <Route path="/">
30              <Home />
31            </Route>
32          </Switch>
33          <Footer />
34        </div>
35      </BrowserRouter>
36    );
37  }
```

Sesi 15 | React Router

Implementasi React Router pada First React App kita

5. Update About.js

Temen-temen bebas isi apapun pada halaman ini. Bebas ber expresi dengan Bootstrap untuk isinya. Berikut hanyalah contoh semata



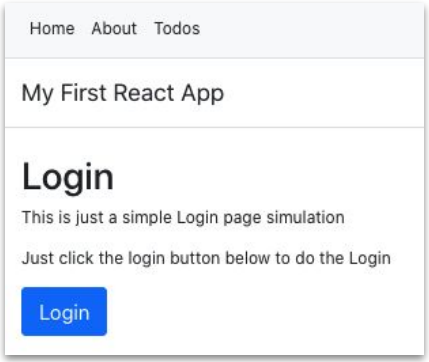
Sesi 15 | React Router

Implementasi React Router pada First React App kita

6. Update Login.js

Login.js ini hanyalah berfungsi sebagai halaman simulasi jika ada fitur login. Fungsional yang dipakai pun sangat sederhana dan bukan fungsional Login yang baik dan benar. Tapi, kita sudah mulai implementasi hook **useHistory** di sini

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

export default function Login() {
  const history = useHistory()

  const doLogin = () => {
    localStorage.setItem('login', 'true')
    history.push("/")
  }

  return (
    <div className="mb-5">
      <h1>Login</h1>
      <p>This is just a simple Login page simulation</p>
      <p>Just click the login button below to do the Login</p>
      <button
        className="btn btn-lg btn-primary"
        onClick={doLogin}>Login</button>
    </div>
  )
}
```



HACKTIV8

Sesi 15 | React Router

Implementasi React Router pada First React App kita

7. Update Header.js

Update Header.js untuk bisa mengimplementasi simulasi fungsional Logout. Jangan lupa untuk update line dimana kita render button, dengan memberikan action untuk event onClick nya

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

export default function Header() {
  const history = useHistory()

  const logout = () => {
    localStorage.removeItem('login')
    history.push("/")
  }
```

```
<button className="btn btn-sm btn-danger" onClick={ logout }>Logout</button>
```

Sesi 15 | React Router

Implementasi React Router pada First React App kita

8. Update Todo.js - part 1

```
import React, { useEffect, useState } from 'react';
import { Redirect } from 'react-router-dom';

export default function Todos() {
  const [todos, setTodos] = useState([])

  useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/todos')
      .then((response) => response.json())
      .then((data) => setTodos(data.slice(0, 10)))
  }, [])

  if (!localStorage.getItem('login')) {
    return (
      <Redirect to="/login" />
    )
  }
```

9. Update Todo.js - part 2

```
  return (
    <>
      <h1>Todo List</h1>
      <table className="table table-striped">
        <thead className="table-dark">
          <tr>
            <th>Id</th>
            <th>Title</th>
            <th>Completed</th>
          </tr>
        </thead>
        <tbody>
          {
            todos.map((todo, idx) => (
              <tr>
                <td>{todo.id}</td>
                <td>{todo.title}</td>
                <td>{todo.completed ? 'v' : 'x' }</td>
              </tr>
            ))
          }
        </tbody>
      </table>
    </>
  )
}
```

Sesi 15 | React Router

Implementasi React Router pada First React App kita

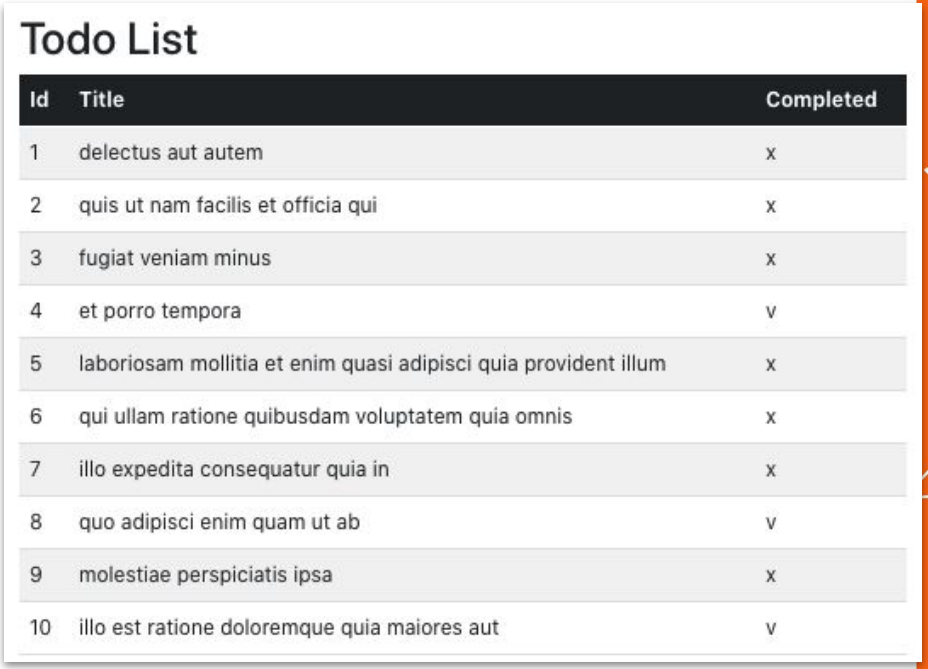
10. Jalankan aplikasi

Mari kita jalankan aplikasi kita dengan perintah :

```
> npm start
```

Jika benar, maka skenario-skenario di bawah ini akan berjalan :

- Jika belum login, maka setiap kita masuk link Todos, kita akan di Redirect ke halaman Login
- Setiap habis login, kita akan di Redirect ke halaman Home
- Halaman Todos akan menampilkan konten seperti di samping ini



Todo List

Id	Title	Completed
1	delectus aut autem	x
2	quis ut nam facilis et officia qui	x
3	fugiat veniam minus	x
4	et porro tempora	v
5	laboriosam mollitia et enim quasi adipisci quia provident illum	x
6	qui ullam ratione quibusdam voluptatem quia omnis	x
7	illo expedita consequatur quia in	x
8	quo adipisci enim quam ut ab	v
9	molestiae perspiciatis ipsa	x
10	illo est ratione doloremque quia maiores aut	v

Sesi 15 | React Router

Implementasi React Router pada First React App kita

Yeay, selamat, nampak aplikasimu sudah semakin canggih ya. Ingin tantangan lebih ? Silakan kamu coba untuk explore bagaimana cara melakukan props pada routing, sehingga kamu bisa mengirimkan pesan yang bisa ditampilkan menggunakan alert Bootstrap ketika Login dan Logout