**BAB II**

**REACT JS LAYOUT & ROUTER DOM**

1. **Composition sebagai Layout Framework**

Pada bagian ini kita akan mengatur Layouting pada project anda sehingga memudahkan anda untuk melakukan mapping dari sisi UI. Beberapa library yang dibutuhkan dalam membangun desain layout ini ialah sebagai berikut:

* React Bootstrap
* React Bootstrap Icon, dan
* React Router Dom

1. **Installasi library Bootstrap**

Untuk menambahkan library bootstrap baik untuk komponen UI ataupun untuk icon, bukalah path project anda dengan terminal lalu masukan syntax dibawah ini:

|  |
| --- |
| npm install react-bootstrap bootstrap bootstrap-icons |

Syntax diatas akan menginstal tiga buah library sekaligus yaitu react-bootstrap, bootstrap, dan bootrstap-icons. Setelah berhasil menginstall library tersebut, bukalah file index.js dan panggil file library css milik bootstrap:

Text, letter

Description automatically generated

Lalu anda jalankan server React untuk mengetahui apakah sudah berhasil terinstallasi atau tidak.

1. **Structure Project**

Buatlah dua buah folder bernama layouts dan modules. Folder layouts nantinya akan berisikan komponen-komponen mengenai emmet UI berdasarkan mock up yang anda miliki. Sedangkan folder modules, diperuntukan untuk menyimpan file mentah dari komponen react. Dimana file inilah yang akan menjadi titik berat pada aplikasi anda.

Application

Description automatically generated with medium confidence

***Gambar 1.1.*** *Struktur Poject react untuk layouting*

Pada folder *layouts* berisikan 2 buah file yaitu folder bernama *components* dan file RFC *Layouts.js*. folder components nantinya berisikan komponen-komponen UI yang sifatnya dapat digunakan secara global, atau berisi komponen dari bentuk UI pada rancangan web anda. Contoh bentuk komponen UI seperti: *Header*, *Navigasi* atau *Footer*. Sedangkan file Layouts.js berisikan code utama untuk membentuk rancangan Layout UI pada web anda. Berikut ini adalah contoh skema dari layout yang akan dirancang, skema ini merujuk pada bentuk template yang dimiliki bootstrap <https://getbootstrap.com/docs/5.3/examples/offcanvas-navbar/>:

Text

Description automatically generated with medium confidence

Berdasarkan code diatas untuk file *Layouts.js*, memiliki dua buah fungsi umum komponen ui yaitu *HeaderNav* dan *Footers*. Kedua fungsi tersebut berada pada folder *./layouts/components/...* File ini menggunakan teknik Composition dimana komponen *Layouts* akan memiliki beberapa komponen yang dapat diturunkan oleh *Layouts*, untuk menandai adanya komponen turunan kita inisialisasi dengan code props.children.

Graphical user interface, text, application

Description automatically generated

File Footers.js hanya berisikan sebuah code untuk menampilkan komponen JSX yang berisi sebuah tulisan hak cipta terhadap website yang anda buat. Sedangkan untuk file HeaderNav.js berisikan code JSX untuk membentuk sebuah navigasi menu bar pada sisi header web anda. Kedua file tersebut dirancang dengan bentuk *react function component*.

Berikut adalah contoh code yang berasal dari file *Footers.js*:

A picture containing text

Description automatically generated

Dan ini adalah contoh code pada file *HeaderNav.js*:

Timeline

Description automatically generated

Script code navigasi dapat anda lihat pada web dokumentasi dengan merujuk code link bootsrap di <https://getbootstrap.com/docs/5.3/components/navbar/#how-it-works>

Untuk melakukan testing apakah komponen Layouts bekerja sesuai dengan rancangan web anda, jadikanlah komponen Layouts sebagai titik awal aplikasi anda pada file index.js:

Graphical user interface, text, email

Description automatically generated

Jika anda menjadikan komponen Layouts sebagai titik awal aplikasi pada file index.js, maka tampilannya akan seperti berikut:

Graphical user interface, text, application

Description automatically generated

***Gambar 1.2.1.****Output komponen Layouts*

1. **Inisialisasi composition**

Pada bagian sebelumnya anda telah membuat script komponen Layouts, dimana file ini adalah sebuah komponen Parent atau komponen yang akan memiliki turunan dari komponen-komponen lain. Berikut ini adalah contoh komponen class dengan nama Home. Dimana komponen tersebut akan mewarisi komponen layout.

Graphical user interface, text, application

Description automatically generated

File ini hanya menampilkan bentuk JSX dengan tulisan heading satu ***“Hello this is a HOME page”***. File komponen class ini berada pada folder *./modules/Homes/Home.js*. Selanjutnya kita akan membuat class komponen ini menjadi komponen turunan Layouts. Bukalah file *index.js* dan ubah script seperti dibawah ini:

Graphical user interface, text

Description automatically generated

Pada script diatas kita menentukan titik awal aplikasi akan menuju ke class komponen Home, namun komponen Home akan dibungkus dengan emmet dari komponen Layouts, ini menandakan bahwa class komponen Home merupakan turunan dari komponen Layouts. Jika dilihat pada output dari script diatas maka akan memiliki tampilan seperti berikut:

Graphical user interface, text, application

Description automatically generated

***Gambar 1.3.*** *Output dari inisialisasi composition*

1. **React react-router-dom**

Pada bagian sebelumnya kita telah merancang sebuah layout pada web anda, jika melirik pada gambar Gambar 1.3 maka ditemukan ada tempat buah menu navigasi yaitu: Home, Explore, Messages, dan Logout. Jika diklik satu persatu link navigasi tersebut maka tidak akan merujuk kehalaman mana pun, namun hanya merujuk pada satu halaman yaitu halaman titik awal (komponen class Home). Untuk membuat sebuah web yang statik maka diperlukan sebuah routing untuk memapping *end-point* pada masing-masing navigasi. Pada react js kita dapat menggunakan react-router-dom untuk melakukan mapping pada setiap *end-point* sehingga kita dapat mengatur komponen-komponen mana saja yang akan dituju.

1. **Installasi react-router-dom**

Bukalah terminal project anda dan tambahkan syntax dibawah ini untuk menginstallasi library react-router-dom:

|  |
| --- |
| npm install react-router-dom |

Setelah berhasil menambahkan library react-router-dom maka pada file package.json akan menambahkan dependencies react-router-dom.

1. **Menggunakan Router**

Buatlah file bernama AppRoute.js pada folder ./apps/AppRoutes.js dan masukan script dibawah ini untuk menentukan end-point pada masing-masing komponen:

Text

Description automatically generated

Pada script diatas, titik awal aplikasi merujuk kepada komponen Home hal ini ditentukan dengan atribute index, yang menandakan bahwa ketika mengakses end-point “/” maka akan menuju komponen Home. Jika kita menuju end-point /home maka akan mengakses komponen Home, sedangkan untuk end-point /explore akan mengakses komponen Explore dan untuk mengakses end-point /messages maka akan diarahkan ke komponen Messages.

Setelah membuat file untuk mengatur end-point pada aplikasi bukalah file index.js, lalu ubah script-nya menjadi:

Text

Description automatically generated

Setelah merubah index.js silakan anda coba jalankan projeknya dan apakah navigasi menu anda dapat berpindah end-point sesuai dengan komponen yang dituju. Berikut adalah contoh hasil dari penggunaan react-router-dom:

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

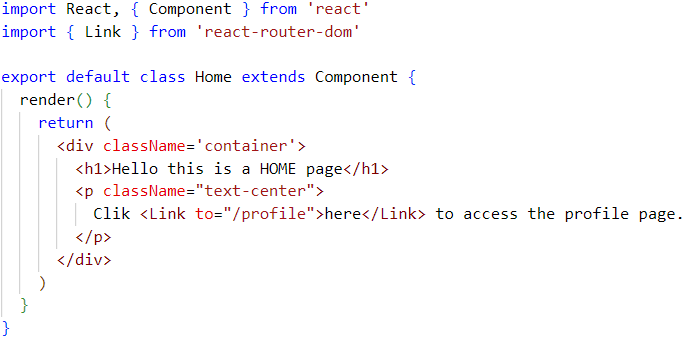
Graphical user interface, application

Description automatically generated

***Gambar 2.2*** *Output navigasi dengan router*

1. **Link**

<Link> adalah elemen yang memungkinkan pengguna menavigasi ke halaman lain dengan mengklik atau mengetuknya. Di react-router-dom, <Link> merender elemen <a> yang dapat diakses dengan href asli menunjuk ke sumber daya yang ditautkannya. Ini berarti hal-hal seperti mengklik kanan <Link> dapat berfungsi dengan baik. Atribute to pada link dapat anda gunakan untuk merujuk pada end-point yang telah didaftarkan. Berikut adalah contoh penggunaan Link pada JSX dengan menggunakan library { Link } from 'react-router-dom':



1. **NavLink**

<NavLink> adalah jenis khusus dari <Link> yang mengetahui apakah itu "aktif" atau "tertunda". Ini berguna saat membuat menu navigasi, tabulasi atau collapse item. Pada contoh ini kita dapat mengganti bentuk <a> pada komponen HeaderNav, berikut adalah contoh dari penggunaan NavLink pada JSX yang berada pada library { NavLink } from 'react-router-dom':

**HeaderNav.js**

Text

Description automatically generated

Jalankan project anda dan lihat perbedaan antara penggunaan <a>, <Link> dan <NavLink>.

1. **Nested Route**

Nested route diperuntukan bagi web application yang ingin memiliki sub end-point, contohnya ketika mengakes end-point profile biasanya akan memiliki beberapa end-point umum seperti:

|  |
| --- |
| https://www.pwl.com/profile  https://www.pwl.com/profile/biodata  https://www.pwl.com/profile/setting |

Untuk membuat nested route di react kita dapat menambahkan code seperti dibawah ini, contoh yang akan dibuat nested routed pada bagain ini yaitu untuk end-point explore:

Text

Description automatically generated

Pada script diatas dapat dilihat bahwasannya end-point /explore memiliki tiga buah sub domain yaitu:

|  |  |
| --- | --- |
| **Sub domain** | **Komponen** |
| */feeds* | Feeds |
| */reels* | Reels |
| */fyp* | FYP |

Graphical user interface, text, application

Description automatically generated

Komponen Explore perlu menambahkan emmet Outlet untuk menciptakan only single page agar dapat merender komponen dari *sub end-point*.

Jika melihat struktur folder Explorer disamping, folder Explorer memiliki satu buah folder bernama widgets. Dimana folder ini isinya adalah file mentah komponen react.

Dalam folder widgets, memiliki 4 buah file RFC yang diperuntukan untuk merender berdasarkan sub end-point expolorer yang telah didefinisikan sebelumnya.

**Categories.js**

Graphical user interface, application

Description automatically generatedText

Description automatically generated

Pada script RFC Categories, file tersebut hanya menampilkan sebuah navigasi yang berisikan daftar-daftar sub end-poin yang dimiliki oleh Explorer.

Sedangkan script untuk komponen sub end-poin hanya menampilkan sebuah text sederhana berdasarkan judul dari masing-masing end-poin.

|  |  |
| --- | --- |
| **Komponen RFC** | **Output** |
| **Feeds.js** |  |
| **Reels.js** |  |
| **FYP.js** |  |

Dan yang terakhir ialah pada komponen main route Explorer perlu menambahkan emmet Outlet agar dapat merender komponen sub end-point:

|  |  |
| --- | --- |
| **Explorer.js** | **Output** |
| Timeline  Description automatically generated |  |

1. **Latihan Praktikum**
2. Pada project sebelumnya buatlah sebuah package baru didalam folder components dengan nama *latihan-3-1*.
3. Pada bagian ***2.2 Menggunakan Router***, terlihat pada komponen AppRoute setiap end-point perlu diinisialisasi atau didefinisikan sebagai turunan komponen Layouts. Jika anda memiliki lebih dari 10 end-point maka anda perlu mendefinisikan ±10 komponen sebagai bentuk turunan komponen Layout, teknik ini dapat sangat merepotkan bagi para developer. Cobalah anda buat teknik model algoritma yang lebih sederhana dari file AppRoute tersebut. Contoh, jika memilki ±10 komponen maka cukup hanya 1 kali mendefinisikannya, maka ±10 komponen tersebut sudah diketahui bahwa mereka adalah komponen turunan dari Layouts.
4. Buatlah sebuah end-point baru untuk Login dengan isian form email dan password (bukan turunan komponen Layouts). Jika data email dan password salah maka akan menampilkan pesan error, sedangkan jika benar maka akan meredirect ke halaman HOME.

Pengumpulan tugas Latihan praktikum dikumpulkan kedalam GITHUB masing-masing mahasiswa berdasarkan repository yang telah dibuat PWL-TI-20-PA-NPM. File source code disimpan sesuai nama project-praktikum dan masukan kedalam repositori tersebut. Buatkanlah file dokumen dalam bentuk file pdf yang berisi Screen Capture dari hasil program yang telah dikerjakan. Simpan dalam file PDF tersebut kedalam project tersebut.

Tambahkan Collaborator management access pada repository anda ke *@FebryFairuz*