

## 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

##### 1.1. Instalasi 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
```

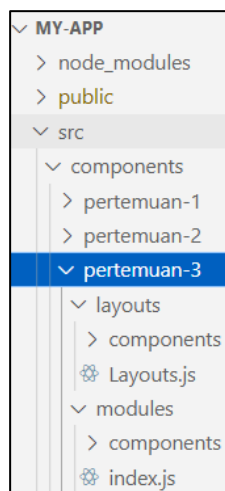
Syntax diatas akan menginstal tiga buah library sekaligus yaitu react-bootstrap, bootstrap, dan bootstrap-icons. Setelah berhasil menginstall library tersebut, bukalah file index.js dan panggil file library css milik bootstrap:

```
index.js > ...
import React from 'react';
import ReactDOM from 'react-dom/client';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap-icons/font/bootstrap-icons.css'
import reportWebVitals from './reportWebVitals';
```

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

##### 1.2. 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.



*Gambar 1.1. Struktur Project 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/>:

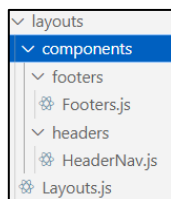
```
import React from 'react'
import Footers from './components/footers/Footers'
import HeaderNav from './components/headers/HeaderNav'

export default function Layouts(props) {
  return (
    <div id="main-layout">
      <HeaderNav />

      <main className='mt-20 py-10'>
        {props.children}
      </main>

      <Footers />
    </div>
  )
}
```

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`.



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*:

```
import React from 'react'

export default function Footers() {
  return (
    <footer className="container">
      <p>&copy; 2023 Febry D F, Inc. &middot;</p>
      <a href="https://wa.me/0813456789" target={"_blank"}>Contact</a></p>
    </footer>
  )
}
```

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

```
import React from 'react'

export default function HeaderNav() {
  const menuList = [{ id: 1, name: "Home", path: "/home", icon: "bi-house-door" },
    { id: 2, name: "Explore", path: "/explore", icon: "bi-compass" },
    { id: 3, name: "Messages", path: "/messages", icon: "bi-send" },
    { id: 4, name: "Log Out", path: "/log-out", icon: "bi-box-arrow-left" }];

  return (
    <header>
      <nav className="navbar navbar-expand-md fixed-top shadow bg-white">
        <div className="container">
          <a className="navbar-brand" href="#">
            
          </a>
          <button className="navbar-toggler" type="button">
            <span className="navbar-toggler-icon"></span>
          </button>
          <div className="collapse navbar-collapse" id="navbarCollapse">
            <div className="d-flex w-100 justify-content-end">
              <div id="main-nav">
                <ul className="navbar-nav me-auto mb-2 mb-md-0">
                  {menuList.map((v, index) => (
                    <li className="nav-item me-1" key={index}>
                      <a className="nav-link text-hover-success px-3" href={v.path}>
                        <i className={"bi me-2 fs-5 text-dark"+v.icon}></i>
                        {v.name}
                      </a>
                    </li>
                  ))}
                </ul>
              </div>
            </div>
          </div>
        </div>
      </nav>
    </header>
  )
}
```

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

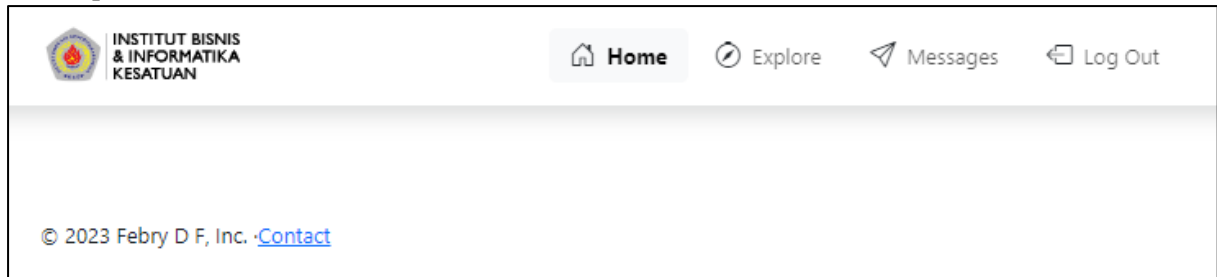
Untuk melakukan testing apakah komponen Layouts bekerja sesuai dengan rancangan web anda, jadikanlah komponen Layouts sebagai titik awal aplikasi anda pada file index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap-icons/font/bootstrap-icons.css';
import reportWebVitals from './reportWebVitals';
import Layouts from './components/pertemuan-3/layouts/Layouts';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Layouts />
  </React.StrictMode>
);

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



*Gambar 1.2.1. Output komponen Layouts*

### 1.3. 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.

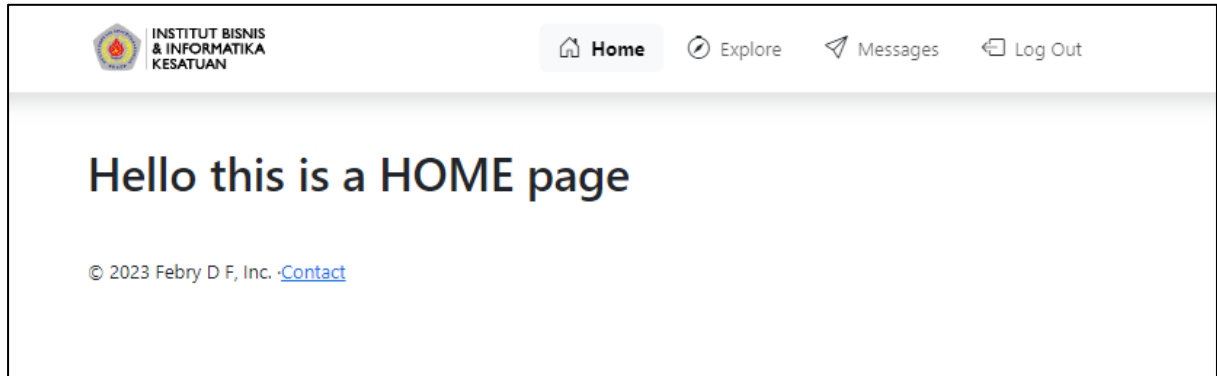
```
import React, { Component } from 'react'

export default class Home extends Component {
  render() {
    return (
      <div className='container'>
        <h1>Hello this is a HOME page</h1>
      </div>
    )
  }
}
```

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:

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Layouts>
      <Home />
    </Layouts>
  </React.StrictMode>
);
```

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:



*Gambar 1.3. Output dari inisialisasi composition*

## 2. 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.

### 2.1. 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`.

### 2.2. 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:

```
import React from 'react'
import { Route, Routes } from 'react-router-dom'
import Layouts from '../layouts/Layouts'
import Home from '../modules/components/Homes/Home'
import Explore from '../modules/components/Explore/Explore'
import Messages from '../modules/components/Messages/Messages'

export default function AppRoute() {
  return (
    <Routes>
      <Route index element={<Layouts><Home /></Layouts>} />
      <Route path='home' element={<Layouts><Home /></Layouts>} />
      <Route path='explore' element={<Layouts><Explore /></Layouts>} />
      <Route path='messages' element={<Layouts><Messages /></Layouts>} />
    </Routes>
  )
}
```

Pada script diatas, titik awal aplikasi merujuk kepada komponen Home hal ini ditentukan dengan attribute 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 bukanlah file index.js, lalu ubah script-nya menjadi:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap-icons/font/bootstrap-icons.css'
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
import AppRoute from './components/pertemuan-3/apps/AppRoute';

const { PUBLIC_URL } = process.env;
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter basename={PUBLIC_URL}>
      <AppRoute />
    </BrowserRouter>
  </React.StrictMode>
);
```

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:



*Gambar 2.2 Output navigasi dengan router*

### 2.3.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. Attribute 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':

```
import React, { Component } from 'react'
import { Link } from 'react-router-dom'

export default class Home extends Component {
  render() {
    return (
      <div className='container'>
        <h1>Hello this is a HOME page</h1>
        <p className="text-center">
          Klik <Link to="/profile">here</Link> to access the profile page.
        </p>
      </div>
    )
  }
}
```

## 2.4. 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

```
{menuList.map((v, index) => (
  <li className="nav-item me-1" key={index}>
    <NavLink className="nav-link text-hover-success px-3" to={v.path}>
      <i className={"bi me-2 fs-5 text-dark "+v.icon}></i>
      {v.name}
    </NavLink>
  </li>
))}
```

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

## 2.5. Nested Route

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

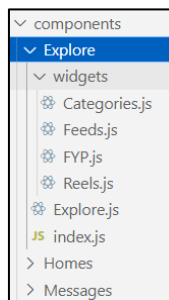
```
https://www.pw1.com/profile
https://www.pw1.com/profile/biodata
https://www.pw1.com/profile/setting
```

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:

```
export default function AppRoute() {
  return (
    <Routes>
      <Route index element={<Layouts><Home /></Layouts>} />
      <Route path='home' element={<Layouts><Home /></Layouts>} />
      <Route path='explore' element={<Layouts><Explore /></Layouts>} >
        <Route path="feeds" element={<Feeds />} />
        <Route path="reels" element={<Reels />} />
        <Route path="fyp" element={<FYP />} />
      </Route>
      <Route path='messages' element={<Layouts><Messages /></Layouts>} />
      <Route path='profile' element={<Layouts><Profile /></Layouts>} />
    </Routes>
  )
}
```

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

Sub domain	Komponen
/feeds	Feeds
/reels	Reels
/fyp	FYP



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.

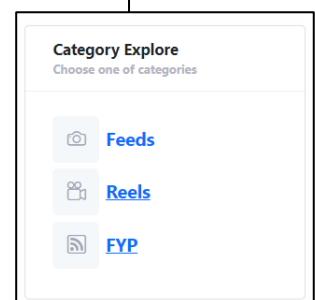
Dalam folder widgets, memiliki 4 buah file RFC yang diperuntukan untuk merender berdasarkan sub end-point explorer yang telah didefinisikan sebelumnya.

## Categories.js

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

export default function Categories() {
  const categories = [
    { id: 1, name: "Feeds", path: "/feeds", icon: "bi-camera" },
    { id: 2, name: "Reels", path: "/reels", icon: "bi-camera-reels" },
    { id: 3, name: "FYP", path: "/fyp", icon: "bi-rss" }];



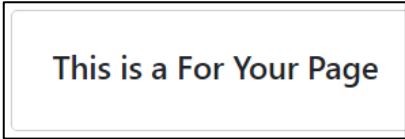
  return (
    <div id='categories' className='card border'>
      <div className="card-header">
        <h3 className="card-title align-items-start flex-column">
          <span className="card-label fw-bolder text-dark">Category Explore</span>
          <span className="text-gray-400 mt-1 fw-bold fs-6">Choose one of categories</span>
        </h3>
      </div>
      <div className="card-body">
        <ul class="nav flex-column">
          {categories.map((v, index) => (
            <li class="nav-item mb-2" key={index}>
              <NavLink class="nav-link text-dark text-hover-primary" aria-current="page" to={"/explore"+v.path}>
                <div className="symbol symbol-50px">
                  <span className="symbol-label">
                    <i class={["fs-4 bi " + v.icon]}></i>
                  </span>
                </div>
                <span class="fs-4 ms-2 fw-bold">{v.name}</span>
              </NavLink>
            </li>
          ))}
        </ul>
      </div>
    </div>
  )
}
```



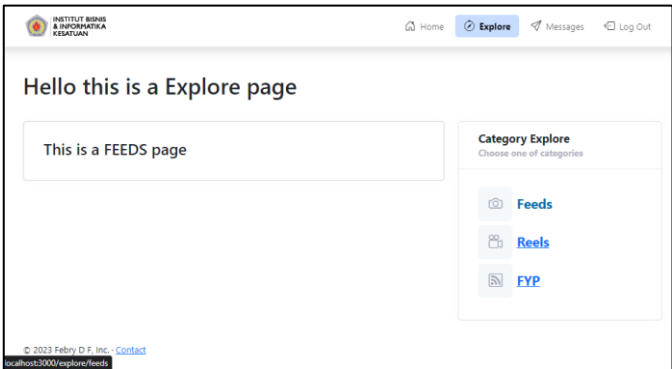


Pada script RFC Categories, file tersebut hanya menampilkan sebuah navigasi yang berisikan daftar-daftar sub end-poin yang dimiliki oleh Explorer.

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

Komponen RFC	Output
<b>Feeds.js</b> <pre>export default function Feeds() {   return (     &lt;div className='card ' id='feeds'&gt;       &lt;div className="card-body"&gt;         &lt;h3&gt;This is a FEEDS page&lt;/h3&gt;       &lt;/div&gt;     &lt;/div&gt;   ) }</pre>	
<b>Reels.js</b> <pre>export default function Reels() {   return (     &lt;div className='card ' id='reels'&gt;       &lt;div className="card-body"&gt;         &lt;h3&gt;This is a REELS page&lt;/h3&gt;       &lt;/div&gt;     &lt;/div&gt;   ) }</pre>	
<b>FYP.js</b> <pre>export default function FYP() {   return (     &lt;div className='card ' id='fyp'&gt;       &lt;div className="card-body"&gt;         &lt;h3&gt;This is a For Your Page&lt;/h3&gt;       &lt;/div&gt;     &lt;/div&gt;   ) }</pre>	

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

Explorer.js	Output
<pre>import React, { Component } from 'react' import { Outlet } from 'react-router-dom' import Categories from './widgets/Categories'  export default class Explore extends Component {   render() {     return (       &lt;div className='container'&gt;         &lt;h1&gt;Hello this is a Explore page&lt;/h1&gt;          &lt;div className="mt-10"&gt;           &lt;div className="row"&gt;             &lt;div className="col-lg-8"&gt;               &lt;Outlet /&gt;             &lt;/div&gt;             &lt;div className="col-lg-4"&gt;               &lt;Categories /&gt;             &lt;/div&gt;           &lt;/div&gt;         &lt;/div&gt;       &lt;/div&gt;     )   } }</pre>	

### 2.5. Navigate

Elemen <Navigate> mengubah lokasi saat ini saat dirender. Ini adalah pembungkus komponen di sekitar `useNavigate`, dan menerima semua argumen yang sama dengan props. Elemen ini biasanya diperuntukan untuk redirect kedalam lokasi end-point.

```
<Navigate to="/home" replace={true} />
```

### 3. Latihan Praktikum

1. Pada project sebelumnya buatlah sebuah package baru didalam folder components dengan nama *latihan-3-1*.
2. 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  $\pm 10$  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 memiliki  $\pm 10$  komponen maka cukup hanya 1 kali mendefinisikannya, maka  $\pm 10$  komponen tersebut sudah diketahui bahwa mereka adalah komponen turunan dari `Layouts`.
3. 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.
4. Buatlah sebuah komponen baru yang menampilkan halaman Error 404. Buatlah sebuah kondisi jika end-point yang tidak didaftarkan atau tidak tersedia dalam `BrowserRouter` maka akan menampilkan halaman Error 404.

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 repository tersebut. Buatlah 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*