Penerapan React JS Pada Pengembangan FrontEnd Aplikasi Startup Ubaform

by Doe John

Submission date: 09-Jun-2021 07:39PM (UTC+0700)
Submission ID: 1603432882
File name: React_JS_Pada_Pengembangan_FrontEnd_Aplikasi_Startup_Ubaform.pdf (1.62M)
Word count: 4436
Character count: 29337

Penerapan React JS Pada Pengembangan *FrontEnd* Aplikasi Startup Ubaform

***Abstract*— Perkembangan web untuk sistem pembelajaran daring pada era pandemi sudah sangat populer. Tidak sedikit muncul nya *startup* baru pada bidang pendidikan yang menyediakan berbagai layanan pembelajaran maupun layanan akademik lainya. Sebagai seorang *developer* sekaligus perintis startup, pengembangan sistem harus dilaksanakan dengan cepat. Pengembangan sistem yang kompleks membutuhkan teknologi yang mendukung proses pengembangan seperti *library atau framework* untuk membantu meningkatkan kinerja dan efisiensi waktu. Pengembangan startup ubaform berbasis *single page application* (SPA). Penerapan *single page application* (SPA) akan meningkatkan kinerja serta pengurangan waktu dan biaya infrastruktur. Oleh karena itu dalam pengembangan *front-end startup* ubaform yang berbasis *single page application* (SPA) menerapkan *library* javascript yaitu ReactJS. Penerapan ReactJS dalam pengembangan aplikasi *startup* ubaform memberikan kecepatan rendering aplikasi yang cepat sehingga pengaksesan aplikasi menjadi lebih cepat, memberikan kemudahan kepada *developer* dalam membuat tampilan yang interaktif dan reusability kode yang mana React mengizinkan *developer* untuk memanfaatkan kembali component yang sudah dikembangkan ke dalam aplikasi lainnya dengan memakai fungsi yang sama.**

Keywords*— *Javascript Library; Single Page Application; ReactJS;

I. PENDAHULUAN

Dalam pengembangan sebuah teknologi tentunya banyak hal yang harus dipersiapkan sehingga teknologi yang dikembangkan bisa digunakan sebagaimana seharusnya. Pengembangan yang terencana secara sistematik akan meminimalisir kesalahan-kesalahan yang mungkin akan terjadi saat proses pengembangan. Dalam pengembangan startup ubaform perancangan terencana mengenai teknologi yang akan digunakan telah dipertimbangkan secara matang. Selain untuk mengurangi kesalahan, merancang pengembangan yang matang bisa membantu menganalisis kelemahan teknologi pengembangan yang sudah ada kemudian memperbaikinya kekurangan yang ada dengan menggunakan teknologi pengembangan yang lebih baru.

Banyak teknologi yang telah ada yang bisa digunakan dalam pengembangan sebuah startup, dalam konteks ini startup ubaform mengembangakan sebuah teknologi web *Software as a service*. *Software as a service* (SaaS) ini adalah model pengembangan perangkat lunak yang memungkinkan pelanggan untuk menggunakan aplikasi yang dihosting sebagai layanan SaaS di Internet. SaaS adalah cara bagi bisnis untuk mendapatkan manfaat yang sama seperti perangkat lunak komersial dengan biaya lebih rendah karena mereka tidak perlu menginstal dan menjalankan aplikasi di komputer mereka. SaaS mengurangi beban pemeliharaan dan dukungan perangkat lunak. Pengguna melepaskan kendali atas versi dan persyaratan perangkat lunak[1].

Pengembangan sebuah web dapat dilakukan dengan banyak teknologi, ubaform sendiri merancang teknologi yang ada dalam sistem dengan mengikuti tren pengembangan web saat ini juga mempertimbangkan tujuan pembuatan web adalah untuk layanan pengerjaan kuis secara daring, pembuatan formulir dan survei yang membutuhkan kecepatan dalam pengaksesan data. Tren pengembangan sebuah web saat ini yang banyak digunakan salah satunya adalah *Single Page Application* (SPA). Hal utama yang menjadi pertimbangan memilih *Single Page Application* (SPA) dibanding dengan *Multi Page Application* (MPA) adalah faktor kecepatan atau performa. Menurut Google dan beberapa penelitian lainnya, waktu buka atau akses yang lambat mempengaruhi kepuasan pengguna, sehingga dapat menghilangkan pelanggan dan mengurangi keinginan mereka untuk kembali ke layanan tersebut. Oleh karena itu pada saat SPA dimuat, jumlah data yang berjalan antara *client-side* dengan *server-side* sangat kecil sehingga waktu pemuatan halaman diminimalkan[2].

II. KAJIAN PUSTAKA

Pada karya ilmiah ini menggunakan beberapa penelitian yang digunakan sebagai landasan penulisan atas kasus yang serupa.

2.1 *Single Page Application* (SPA)

Single Page Application (SPA) merupakan aplikasi yang berjalan di browser dan tidak memerlukan pemuatan ulang halaman saat digunakan. Ini berarti bahwa pengguna tidak berpindah halaman saat *user* mengirim setiap permintaan ke server. Menerapkan *single page application* memiliki dua manfaat utama: mengurangi penggunaan bandwidth jaringan dan mempercepat penjelajahan. SPA ini didukung secara luas oleh pustaka JavaScript yang mengurangi bandwidth jaringan. Data yang diterima dari server dalam format JSON *(JavaScript Object Notation)* dan dapat dilihat secara asinkron menggunakan JavaScript, sehingga penjelajahan menjadi lebih cepat[3].

2.2 React JS

React adalah *open-source library* JavaScript deklaratif, efisien dan fleksibel untuk membangun antarmuka pengguna. React memungkinkan untuk membuat *user interface* yang kompleks dengan set kode kecil yang terisolasi yang disebut "*component*." React js ini digunakan untuk menangani lapisan tampilan dalam aplikasi satu halaman dan pengembangan *mobile application*. React js dikelola oleh facebook, instagram dan komunitas pengembang dan korporasi. React berusaha untuk memberikan kecepatan, kesederhanaan dan skalabilitas. Beberapa fitur yang paling mencolok adalah JSX, Komponen Stateful, Model Objek Dokumen Virtual.

Seperti yang telah disinggung sebelumnya penggunaan React JS dalam pengembangan membuat proses pembuatan *user interface* interaktif menjadi lebih mudah. Ketika membuat sebuah tampilan menggunakan React JS untuk setiap *state* di aplikasi yang dikembangkan React JS akan secara efisien memperbarui dan merender setiap perubahan yang terjadi hanya pada *state* tersebut. Penulisan *syntax* React JS yang deklaratif membuat alur kode menjadi lebih mudah terprediksi dan mudah untuk dilakukan *debug.* React JS berbasis *component* yang mana dapat membuat beberapa *component* yang terenkapsulasi sehingga mengatur *state*-nya sendiri, kemudian Komponen tersebut digabungkan untuk membentuk *user interfaces* yang lebih kompleks. Hal ini juga memudahkan *developer* ketika melakukan *maintenance* ketika terjadi hal yang tidak sesuai karena developer akan langsung menuju component yang bermasalah tanpa mengganggu component lainya sehingga selain mudah juga lebih cepat. Selain itu komponen-komponen yang telah ditulis bisa digunakan kembali ketika *developer* membutuhkan hal ini

akan menghemat waktu dan efisiensi kode agar tidak ditulis secara berulang-ulang.

2.3 JSX

JSX adalah sintaks seperti XML atau HTML yang digunakan oleh React untuk memperluas ECMAScript sehingga teks seperti XML atau HTML dapat digunakan berdampingan dengan kode JavaScript atau React. React memiliki logika rendering secara inheren digabungkan dengan logika UI lainnya, seperti bagaimana event ditangani, bagaimana status dapat berubah dari waktu ke waktu, dan bagaimana data dapat disiapkan untuk ditampilkan. JSX direkomendasikan untuk digunakan bersama dengan react dengan tujuan untuk menjelaskan bagaimana tampilan UI sehingga memungkinkan react untuk menampilkan pesan dan peringatan yang lebih berguna bagi *developer*. JSX memudahkan *developer* karena kode antara markup (HTML) dapat digabungkan langsung dengan logika (Javascript) sehingga penulisan sintaksnya lebih sederhana dan mudah daripada penulisan keduanya secara terpisah. Contoh penulisan syntax JSX :

```
const name = 'Budi';
const element = <h1>Halo, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

2.3 *Node package manager* (NPM)

NPM adalah *Node Package Manager* yang populer digunakan secara luas oleh pengembang JavaScript untuk berbagi tool, menginstal modul, dan mengelola dependensi. NPM dapat dilihat sebagai repositori untuk proyek *open source* bagi siapa saja yang ingin menggunakan dan mempublikasikan kode-kode tertentu. Hal Yang dapat dilakukan NPM adalah dapat menginstal dan meng-uninstal package, serta mengelola versi dan dependensi yang diperlukan untuk menjalankan proyek. NPM ada bersamaan dengan saat instalasi Node JS. Penggunaan NPM dalam proyek ubaform akan membantu dalam menginstal beberapa *package* yang dibutuhkan dalam pengembangan sehingga mempermudah pengembang serta menghemat waktu pengembangan startup.

III. METODOLOGI

Pada tahap ini dilakukan metodologi penerapan React JS antara lain tahapan perancangan dan implementasi Detail tahapan sebagai berikut:

A. *Perancangan React* JS pada *front end* ubaform

Tahap perancangan merupakan tahap yang sangat penting dalam pengembangan perangkat lunak dan menjadi dasar dalam proses pengembangan yang dilakukan. Dalam perancangan tampilan ubaform selain menggunakan React JS juga menggunakan beberapa *package* yang mendukung pengembangan serta kompleksitas dari tampilan ubaform yang dikembangkan. Penggunaan *package* dalam *library* React JS sudah sangat umum dilakukan oleh *developer*.

B. *Implementasi* React JS pada *front end* ubaform

Tahap selanjutnya adalah implementasi, tahap ini menjelaskan bagaimana implementasi dari React JS dan beberapa *package* bawaan yang digunakan dalam pengembangan. Selain itu juga menjelaskan bagaimana menggabungkan beberapa *component state* sehingga membentuk tampilan yang kompleks serta menjelaskan *page* penggunaan react router yang digunakan untuk *routing* ke beberapa *page* aplikasi.

IV. HASIL DAN PEMBAHASAN

Bab ini menjelaskan setiap langkah yang dijelaskan pada bab sebelumnya dengan lebih jelas.

A. *Perancangan React* JS pada *front end* ubaform

Seperti yang dijelaskan dalam metodologi, tahap ini akan menjelaskan hasil dan pembahasan dari perancangan sistem ubaform menggunakan React JS dan beberapa *package* dalam pengembangan *startup* ubaform. Dalam *paper* ini tidak akan dibahas semua rancangan tampilan yang akan dikembangkan mengingat batasan halaman yang terbatas. Adapun beberapa rancangan tampilan yang akan dibahas dan dikembangkan antara lain :

1. *Landing Page*
2. *Login Page* dan *Register Page*
3. *Dashboard Page*

Pada perancangan pembuatan *front end* ubaform hal yang akan kita lakukan adalah dengan melakukan *Create React App* (CRA) untuk menginisiasi project React. Setelah merancang inisiasi react *app* selanjutnya adalah menentukan *package* apa saja yang akan kita butuhkan dalam proses pengembangan *front end* ubaform. Dalam mengembangkan React App dengan bantuan *package* dari NPM sangat memudahkan dan mempercepat pengembangan. Hal ini dikarenakan dengan *package* yang ada pada NPM bisa langsung digunakan sesuai dengan kebutuhan. *Package* biasanya berisi kode siap pakai yang ditulis oleh *developer* lain dan bisa digunakan secara gratis (*open source code*) pada project yang menggunakan library javascript seperti react JS.

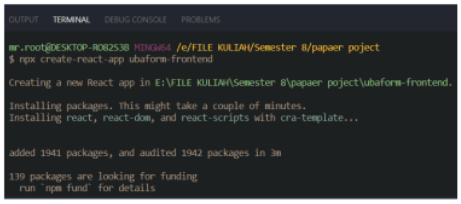
Berikut beberapa package yang dibutuhkan dalam pengembangan startup ubaform :

1. *Package* `react-router-dom` untuk membuat kita dapat berpindah halaman dari halaman satu ke halaman lainya. Perpindahan halaman di sini tentu saja masih menganut *Single page application* yang mana perpindahan halaman terjadi pada satu page tanpa melakukan loading atau pemuatan ulang halaman..
2. *Package* `material UI`. Penggunaan Material UI membantu dalam mempercepat proses pembuatan user interface. `material UI` memiliki beberapa package berbeda antara lain `@material-ui/core`, package Material UI core tersedia berbagai component seperti *Button, Card, Form* dan component lainya. `@material-ui/icons` khusus menyediakan Icon dan `@material-ui/lab` menyediakan component yang tidak ada pada `@material-ui/core`.
3. *Package* `zxcvbn` yang berfungsi untuk membantu melakukan pengecekan terhadap kekuatan kombinasi sandi yang diinput oleh user.
4. *Package* `react-multi-carousel` yang berfungsi untuk membuat tampilan card bergerak (*carousel card*).
5. *package* `syncfusion/ej2-react-navigations` package yang berfungsi dalam membantu pengembangan dashboard terutama pada navigasi sidebar untuk user interface ubaform.
6. *package* `react-icons` yang berfungsi menyediakan icon-icon yang digunakan dalam pembuatan *user interfaces* jika icon pada `@material-ui/icons` kurang atau tidak tersedia.

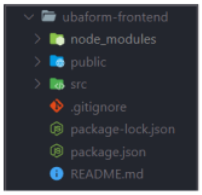
B. *Implementasi* React JS pada *front end* ubaform

Selanjutnya adalah implementasi, pada tahap ini implementasi React JS dimulai dengan *Create React App* (CRA) dengan menuliskan `npx create-react-app ubaform-frontend` pada terminal. Syntax tersebut akan

menginisiasi project React. Syntax `npx` adalah package runner tool yang tersedia bersamaan dengan npm 5.2 atau versi 5.2 keatas. Kemudian `create-react-app` mengindikasi maksud atau tujuannya, yaitu untuk membuat *react app.* Selanjutnya diikuti dengan nama project react app yang dibuat dalam konteks ini adalah `ubaform-frontend`. Ketika CRA berlangsung akan ada banyak sekali package yang akan diinstal. diantara package tersebut adalah react package, react-dom package, react-scripts package dan banyak lagi. Seperti yang terlihat pada gambar berikut.



Setelah *Create React App* terjadi maka akan ada struktur folder seperti pada gambar berikut



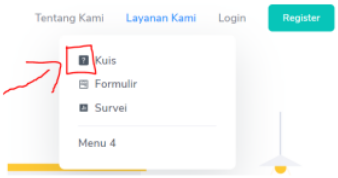
Setelah project react-app terinstall selanjutnya adalah membuat user interfaces dari ubaform. Komponen penyusun user interface terdiri dari beberapa *component*. Komponen ini akan kita buat dalam folder `src` dengan folder tersendiri bernama `components`. Pembuatan Komponen berdasarkan fungsionalitas masing-masing dari user interfaces, seperti *component* header, footer, sidebar dan lain sebagainya. Nantinya Komponen-komponen kecil ini akan disatukan menjadi sebuah komponen besar sehingga membentuk user interface yang kompleks.

Pembuatan *user Interfaces* dimulai dengan membuat *landing page*. Pembuatan *landing page* dimulai dengan membuat folder `landingpage` yang terdiri dari file JSX dan CSS. Pada file JSX inisiasi penggunaan package dilakukan. pada landing page package yang digunakan adalah *package* `react-icons`, `react-router-dom`. Untuk menginisiasi penggunaan *package* dilakukan dengan menuliskan `import` **`{nama package}`** `from` **`sumber package.`** Dalam konteks ini adalah melakukan `import` kedua *package* `react-icons` dan `react-router-dom`. Berikut contoh gambar inisiasi package pada *file* JSX/JS.

```
3  import { BsQuestionSquareFill } from "react-icons/bs";
4  import {FaWpforms } from "react-icons/fa";
5  import {RiBarChartBoxFill} from 'react-icons/ri'
6  import { Link } from "react-router-dom";
```

Inisiasi package `react-icons` dilakukan sesuai dengan dokumentasi dari situs resmi react *icon*, yatu nama *package* harus sesuai dengan nama icon.

Selanjutnya setelah inisiasi *package* adalah tahap penggunaan *package*. Penggunaan *package* dilakukan dengan menulis *syntax* `<BsQuestionSquareFill/>` untuk penggunaan icon BsQuestionSquareFill pada *file* JSX yang ada dalam *functional component* atau *class component.* Hasil penerapan bisa dilihat pada gambar berikut



Untuk `react-router-dom` sebagaimana dijelaskan sebelumnya bahwa *package* ini digunakan untuk memungkinkan user dalam berpindah halaman, dari halaman yang satu ke halaman lainya, namun dalam konteks *Single Page Application.* Artinya ketika user ingin mengakses salah satu halaman dalam sebuah *page* maka tidak perlu dilakukan pemuatan ulang halaman. ada beberapa *component* pada *package* `react-router-dom` yang mana masing-masing *component* memiliki fungsionalitas yang berbeda-beda. Namun pada *landing page* kita akan menggunakan beberapa diantaranya, `BrowserRouter`, `Link` dan `Route`. `BrowserRouter` digunakan untuk membungkus *root component*. Pada *front end* ubaform BrowserRouter dikonfigurasikan pada file tersendiri yang bernama `MainRouter.js` untuk memudahkan melakukan manajemen *routing*. Pada file `MainRouter.js` `BrowserRouter` membungkus beberapa Route. Route *component* memiliki parameter `path` dan `component`. `Path` pada `route` digunakan seperti sebuah objek URL untuk menampilkan *component* tertentu, sedangkan `component` pada `Route` adalah apa yang akan ditampilkan nantinya. Berikut contoh penggunaan `BrowserRouter` dan `Route` dari *package* `react-router-dom`

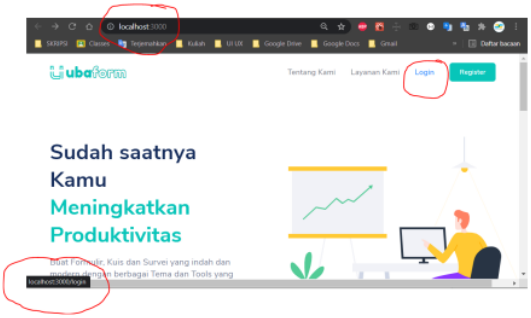
```
11  function MainRouter() {
12    return (
13      <div>
14        <BrowserRouter>
15          <Route path="/" exact component={Landing} />
16          <Route path="/login" component={Login} />
17          <Route path="/register" component={Register} />
18          <Route path="/dashboard" exact={true} component={Dashboard} />
19          <Route path="/userprofile" component={MainUserProfile} />
20          <Route path="/builder" component={MainBuilder} />
21          <Route path="/play" component={QuizPlay} />
22        </BrowserRouter>
23      </div>
24    );
25  }
```

Pada *landing page* ubaform penggunaan *package* `react-router-dom` hanya digunakan untuk berpindah halaman sehingga inisiasi yang ditulis hanya `Link` saja. Biasanya pada HTML biasa penulisan link dilakukan dengan *tag* `<a></a>` akan tetapi pada React JS yang menggunakan format penulisan JSX tag nya berubah menjadi `<Link></Link>` yang mana berasal dari package `react-router-dom`. Penulisan syntax `Link` dilakukan dengan dengan menambahkan parameter `to` sebagai pengganti `href` pada *tag* `<a></a>` yang mengindikasikan alamat atau URL dari Link yang di maksud atau menjadi tujuan. Agar halaman bisa berpindah definisi pada parameter `to` harus sama dengan parameter yang ada pada Route sehingga ketika Link di klik maka Route yang memiliki URL yang sama akan menampilkan *component* yang ada pada route parameter. Berikut contoh gambar penerapan penggunaan Link.

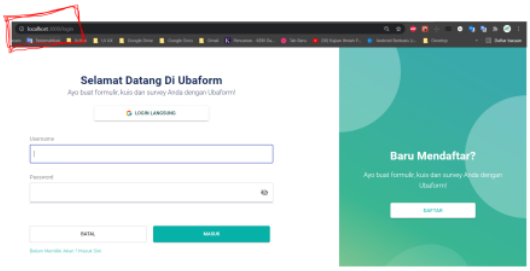
```
68    <li className="nav-item ">
69      <Link to="/login" className="nav-link">
70        Login
71      </Link>
72    </li>
```

Pada gambar tersebut mengindikasikan bahwa URL yang akan dituju adalah "`/login`" yang mana ketika ketika user mengklik navigasi item login maka page akan berganti dari landing page yang URL default nya "`localhost:3000`"

menuju login page yang URL nya "localhost:3000/login" . Seperti yang terdapat pada gambar berikut



Pada Gambar diatas ketika navigasi *login* di hover atau kursor berada diatas navigasi login maka akan muncul URL tujuan dari navigasi *login* tersebut. Setelah *login* navigasi di klik maka tampilan berubah ke login page beserta URL nya tanpa melakukan pemuatan ulang halaman. Seperti yang terdapat pada contoh berikut



Selanjutnya penerapan React JS pada pengembangan *Login page*. *Login page* memiliki beberapa *component* penyusun diantaranya login form, validasi kekuatan *password* dan *button*. Untuk membuat *login form*, package yang digunakan adalah `@material-ui/core`. Dimulai dengan melakukan inisiasi Komponen penyusun *form* yang tersedia dalam package `@material-ui/core` antara lain :

`FormGroup`, `InputLabel`, `OutlinedInput`, `InputAdornment`, `IconButton`. Untuk melakukan import *component* sekaligus dari *package* dilakukan dengan membungkus nama *component* menggunakan kurung kurawal dan dipisahkan dengan menggunakan tanda koma. seperti pada gambar berikut :

```
import {
  Divider,
  FormGroup,
  InputLabel,
  OutlinedInput,
  IconButton,
  InputAdornment
} from "@material-ui/core";
```

Kemudian untuk implementasi dapat dilihat pada gambar berikut

```
<FormGroup>
  <InputLabel className={classes.labelInput}>Username </InputLabel>
  <OutlinedInput
    variant="outlined"
    margin="normal"
    required
    fullWidth
    id="username"
    name="username"
    autoComplete="username"
    autoFocus
    style={{ marginBottom: "35px" }}
  />
  <InputLabel className={classes.labelInput}>Password </InputLabel>
  <OutlinedInput
    id="outlined-adornment-password"
    type={values.showPassword ? "text" : "password"}
    value={values.password}
    onChange={(e) => setPassword(e.target.value)}
    endAdornment={
      <InputAdornment position="end">
        <IconButton
          aria-label="toggle password visibility"
          onClick={handleClickShowPassword}
          onMouseDown={handleMouseDownPassword}
          edge="end"
        >
          {values.showPassword ? <Visibility /> : <VisibilityOff />}
        </IconButton>
      </InputAdornment>
    }
  />
</FormGroup>
```

Potongan kode pada gambar tersebut akan menghasilkan *user interface* sebagai berikut

Pada *login form* bagian *password field* terdapat penggunaan react hooks. Hooks memungkinkan *developer* untuk menggunakan *state* dan fitur React lainnya tanpa harus membuat kelas. penggunaan react hooks dalam konteks ini digunakan untuk membuat inputan *password* dapat dilihat atau disembunyikan dengan *trigger eyes icon* pada akhir *password field*. untuk penggunaan hooks dilakukan dengan melakukan inisiasi dengan menuliskan syntax

```
import React, { useState } from 'react';
```

kemudian melakukan deklarasi variabel. Ada dua deklarasi variabel, yang pertama untuk password dengan kondisi awal berupa *string* kosong dan yang kedua untuk values *state* dengan kondisi awal variabel `showPassword` adalah *false.*

```
const [password, setPassword] = useState("");
const [values, setValues] = useState({
  showPassword: false,
});
```

Selanjutnya adalah menginisiasi variabel ke dalam `OutlinedInput` bagian *password* seperti pada gambar berikut :

```
<OutlinedInput
  id="outlined-adornment-password"
  type={values.showPassword ? "text" : "password"}
  value={values.password}
  onChange={(e) => setPassword(e.target.value)}
```

Seperti yang terlihat pada gambar bahwa *attribute* type `{values.showPassword?"text" : "password"}` memiliki nilai dengan kondisional `text` dan `password`. Pada tahap ini jika kondisi awal pada deklarasi sebelumnya adalah false, maka kondisi pada inisiasi kondisional dari *attribute type* adalah `password` dan begitupun sebaliknya. untuk mengganti kondisional *attribute type* diperlukan *event handler* sebagai proses lanjutan setelah *trigger icon eyes* di klik. Pembuatan *event handler* dilakukan dengan konsep memperbarui kondisi awal dari values `showPassword` yang awalnya *false* menjadi *true* dengan cara seperti yang terlihat pada gambar kode berikut

```
const handleClickShowPassword = () => {
  setValues({ showPassword: !values.showPassword });
};
```

Jadi `setValues` disini akan memperbarui nilai `values` menjadi kebalikan dari nilai awal artinya dari *false* ke *true* dan sebaliknya.

Kemudian pada *login form* bagian *password field* terdapat pengecekan kekuatan dari inputan *password* oleh *user* dengan indikator-indikator khusus. Untuk membuat indikator *password*, *package* yang digunakan adalah `zxcvbn`. Dengan adanya password indikator diharapkan *user* akan lebih memperhatikan kompleksitas pembuatan *password* sehingga tidak terjadi hal yang tidak diinginkan karena kurang kuatnya *password* yang digunakan.

Hal pertama adalah membuat file passwordIndicator.jsx kemudian melakukan inisiasi untuk menggunakan package yaitu dengan menulis syntax `import zxcvbn from "zxcvbn"` Setelah itu membagi kondisi atau status indikator *password* menjadi enam bagian. Keenam indikator ini akan ditandai dengan keterangan kata dan keterangan warna sebagaimana yang tercantum dalam tabel berikut:

Keterangan Kata	Keterangan Warna
Tidak ada inputan	Tidak ada Warna
Sangat Lemah	#828282
Lemah	#EA1111
Rentan	#FFAD00
Bagus	#9BC158
Kuat	#00B500

Dengan membuat dua keterangan seperti pada tabel berarti juga membuat dua *function* dengan kondisi masing-masing. Kondisional pada kedua fungsi sama-sama menggunakan *switch-case* dengan parameter kondisi adalah inputan user melalui field password. Sebelumnya deklarasi Hooks variabel `password` memiliki nilai awal berupa string kosong. String kosong ini nantinya akan diperbarui dengan `setPassword` menggunakan *onChange attribute value* pada *component* `OutlinedInput` seperti pada gambar kode berikut

```
    value={values.password}
    onChange={(e) => setPassword(e.target.value)}
```

Jadi setiap inputan pada *password field* akan memperbarui nilai `password` yang awalnya berupa string kosong. Kemudian nilai `password` inilah yang akan dioperkan ke *component PasswordIndicator.jsx* menggunakan *props*. Nilai operan ini akan diakumulasi oleh package `zxcvbn` yang mana terdapat empat kondisi yang akan merubah indikator. Kondisi Lemah, Rentan, Bagus dan Kuat. Nilai sebelumnya diakumulasi dari total 100/4 akan menghasilkan 25% untuk setiap kondisi. Lebih jelasnya ada pada gambar kode berikut

```
const passwordIndicator = ({ password }) => {
  const testResult = zxcvbn(password);
  const num = (testResult.score * 100) / 4;
```

Jadi artinya ketika user menginput *password* maka yang terjadi adalah kombinasi karakter yang diinput akan diakumulasi dengan empat nilai yang mewakili empat kondisi sebelumnya. untuk setiap kombinasi akan ada penambahan 25% sehingga semakin banyak kombinasi karakter akan semakin kuat indikator password yang dihasilkan. Hal ini bisa terlihat pada saat melakukan `console.log` seperti pada gambar berikut

```
0                    PasswordIndicator.js:7
25                   PasswordIndicator.js:7
50                   PasswordIndicator.js:7
75                   PasswordIndicator.js:7
100                  PasswordIndicator.js:7
>
```

Selanjutnya adalah mengatur agar penambahan tersebut dapat digunakan dalam kondisional *switch-case* sebelumnya sehingga dapat merubah keterangan indikator baik keterangan warna maupun keterangan kata. Dengan menjadikan nilai sebelumnya parameter maka akan ada lima kondisi, akan tetapi hanya empat kondisi yang dapat merubah indikator. Berikut gambar potongan kode kondisional *switch-case*

```
const createPassLabel = () => {
  switch (testResult.score) {
    case 0:
      return "Sangat Lemah";
    case 1:
      return "Lemah";
    case 2:
      return "Rentan";
    case 3:
      return "Bagus";
    case 4:
      return "Kuat";
    default:
      return "none";
  }
};
```

```
const functionProgressColor = () => {
  switch (testResult.score) {
    case 0:
      return "#828282";
    case 1:
      return "#EA1111";
    case 2:
      return "#FFAD00";
    case 3:
      return "#9BC158";
    case 4:
      return "#00B500";
    default:
      return "none";
  }
};
```

Jadi parameter switch-case adalah `testResult.score` Kalkulasi yang terjadi adalah setiap penambahan 25% akan menghasilkan nilai 1 `testResult.score` sehingga masuk pada case 1 dan merubah indikator begitupun seterusnya. Selanjutnya adalah dengan mengimplementasikan nilai tersebut ke dalam style CSS untuk mengatur lebar dan warna dari progress bar sebagai media indikator. Berikut Potongan kode implementasinya

```
  const changePasswordColor = () => ({
    width: `${num}%`,
    backgroundColor: functionProgressColor(),
    height: "7px",
  });
```

```
<div className="progress" style={{ height: "7px" }}>
  <div className="progress-bar" style={changePasswordColor()} />
</div>
<p style={{ color: functionProgressColor() }}>{createPassLabel()}</p>
```

Pada implementasi ini yang terjadi adalah sebuah *progress bar* dengan *style* CSS dinamis yang diatur berdasarkan nilai hasil akumulasi dan kondisional sebelumnya sehingga menghasilkan indikator *password* yang terlihat seperti pada gambar berikut.

Selanjutnya pengembangan masuk kepada tahap pembuatan *dashboard* untuk *user*. *Dashboard* ubaform ini memiliki beberapa *component* utama yang terdiri dari *Header, Sidebar,* dan *Main Content*. Untuk pembuatan *component* dashboard di menggunakan *package* `@material-ui/core`, `@material-ui/icons/` dan `react-router-dom`.

Pada *header* penggunaan *package* `@material-ui/core` untuk membuat *user avatar, dropdown menu, makeStyle*. Pada *header dashboard* dilakukan kostumisasi *style* CSS pada *component button* dengan *component* bawaan dari *package* `@material-ui/core` `makeStyle` dimulai dengan melakukan `import` *component* dari *package*.

```
import  makeStyles from "@material-ui/core/styles";
```

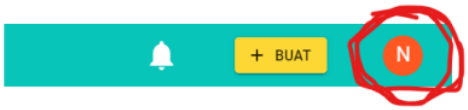
Selanjutnya membuat function dengan nama `useStyles` dan diinisiasi dengan *component* `makeStyle` seperti yang terlihat pada gambar berikut

```
const useStyles = makeStyles((theme) => ({
  root: {
    display: "flex",
    "& > *": {
      margin: theme.spacing(1),
    },
  },
  orange: {
    color: theme.palette.getContrastText(deepOrange[500]),
    backgroundColor: deepOrange[500],
    cursor: "pointer",
  },
```

Di dalam function `useStyle` dibuat variabel-variabel sesuai kebutuhan dengan *value style* CSS masing-masing. Kemudian pada *main function Header* dilakukan inisiasi lagi dengan membuat variabel bernama `classes`. variabel `classes` diinisiasi dengan function `useStyle`. untuk penggunaan *style* tersebut bisa dilakukan dengan menggunakan `className={classes.namaStyle}` seperti pada gambar berikut

```
<Avatar onClick={handleClickAvatar} className={classes.orange}>
  N
</Avatar>
```

Pada konteks ini penggunaan *custom style* CSS dilakukan pada *component user avatar* dengan tujuan membuat warna *avatar* menjadi warna orange.



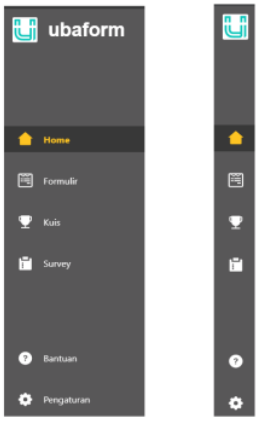
Selanjutnya Pada *Sidebar Dashboard* untuk *package* yang digunakan adalah `syncfusion/ej2-react-navigations`. *Package* ini membantu pembuatan sidebar dengan responsivitas yang baik. Penggunaan *package* dimulai dengan inisiasi *component* yang digunakan.

```
import { SidebarComponent } from "@syncfusion/ej2-react-navigations";
```

Kemudian untuk penggunaan `SidebarComponent` dilakukan dengan menulis *syntax* sebagai berikut

```
<SidebarComponent
  id="dockSidebar"
  ref={(Sidebar) => (this.dockBar = Sidebar)}
  enableDock
  dockSize="62px"
  width="220px"
  className="mr-5"
>
```

Pada `SidebarComponent` terdapat dua tampilan, yaitu tampilan *sidebar* dengan *icon* dan label lalu yang kedua adalah tampilan sidebar icon saja. Selain itu ada beberapa *attribute* didalamnya, attribute `dockSize` berfungsi untuk mengatur lebar sidebar sehingga lebar tampilan hanya menampilkan icon saja, `width` berfungsi untuk mengatur lebar maksimal dari sidebar yang mana menampilkan icon dengan labelnya. untuk lebih jelasnya pada gambar berikut



Untuk melakukan transformasi diantara keduanya membutuhkan *trigger*. Pada *sidebar dashboard* ubaform pembuatan *sidebar* menggunakan *class component.* Sehingga penggunaan akan sedikit berbeda dengan *functional component* yang menggunakan Hooks seperti yang sudah kita bahas sebelumnya. Untuk itu ketika menggunakan trigger `onClick={this.toggleClick}` akan ada `this` sebelum nama event handler.`this` disini akan merujuk pada *event handler.* Pada event handler ini juga menggunakan *arrow function* agar tidak melakukan binding lagi pada *constructor.*

```
// Toggle(Open/Close) the Sidebar
toggleClick = () => {
  this.dockBar.toggle();
}
```

Kemudian selanjutnya pada `SidebarComponent` diinisiasi `ref` yang bertujuan untuk mengakses elemen react. dalam konteks ini elemen react yang dibuat dalam *render method* adalah `SidebarComponent`. Pada konteks ini `ref` merujuk pada `this.dockbar.toggle()` yang ada pada *event handler* sehingga bisa membuat *sidebar* melakukan transformasi lebar dari yang hanya menampilkan *icon* bisa menjadi *sidebar* yang menampilkan *icon* beserta labelnya.

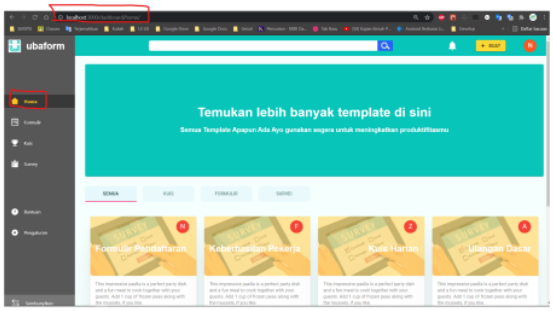
Setelah itu membuat navigasi pada sidebar dengan menggunakan `react-router-dom`. Navigasi yang dibuat pada sidebar dashboard ubaform menggunakan *component* `NavLink` untuk menentukan URL. Penggunaan `NavLink` dalam konteks ini lebih cocok dibanding menggunakan `Link` hal ini dikarenakan pada `NavLink` ketika diakses akan langsung memberi indikator aktif pada `NavLink` tersebut.

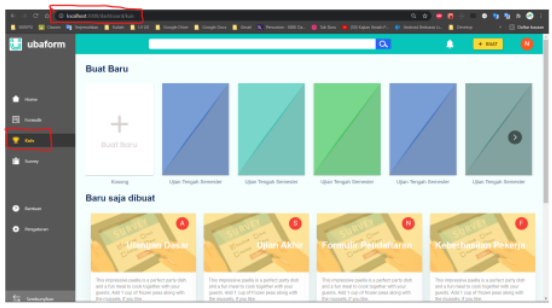
```
<NavLink
  to="/dashboard/home"
  className="nav_link"
  id="show-name"
>
  <RiHome2Fill size={25} />
  <span className="nav_name">Home</span>
</NavLink>
```

```
<Switch>
  <Route path="/dashboard/home">
    <DashboardHome />
  </Route>
  <Route path="/dashboard/formulir">
    <FormulirDashboard />
  </Route>
  <Route path="/dashboard/kuis">
    <KuisDashboard />
  </Route>
  <Route path="/dashboard/survey">
    <SurveyDashboard />
  </Route>
  <Route path="/user">{/* <UserProfile /> */}</Route>
</Switch>
```

Selanjutnya untuk mengubah atau berpindah *page* menggunakan `Route`. Perpindahan *page* ini terjadi pada *main content dashboard.* Ada beberapa Navigasi yang dibuat antara lain Home, Formulir, Kuis, Survey, Bantuan dan

Pengaturan. Seperti yang terlihat pada kode ada penggunaan *component* `Switch`. `Switch` akan memeriksa URL dari `NavLink` yang benar-benar cocok dengan `path` dari `route` tersebut dan hanya akan menampilkan satu dari beberapa *component*. Jadi jika `Switch` menemukan `path` yang cocok dengan `NavLink`, maka Switch akan mengabaikan `path` yang lainnya. Selain itu pada navigasi *sidebar* ini penggunaan *package* `react-icons` diimplementasikan untuk setiap *list item* navigasi. Penggunaan package `react-icons` pada konteks ini untuk mendapatkan *icon* yang cocok yang tidak terdapat pada package `@material-ui/icons`. Seperti pada hasil di bawah ini





Terlihat pada gambar diatas, bahwa ketika pertama kali *user* diarahkan dari *login page* ke *dashboard*, *dashboard* akan melakukan *redirect* ke halaman */dashboard/home*. Komponen Redirect ini juga bawaan dari package `react-router-dom`. Untuk implementasinya sebagai berikut

```
<Redirect from="/dashboard" to="/dashboard/home" />
```

Kemudian ketika user ingin pindah dari *dashboard home* ke kuis *page*, *user* akan mengklik navigasi yang ada pada sidebar dan tampilan yang berubah hanya pada *main content dashboard* saja sedangkan untuk sidebar dan header tidak mengalami perubahan.

Selanjutnya Pembahasan akan masuk ke *main content dashboard*. Pada *main content dashboard* untuk *dashboard home* menampilkan *Card* dengan *text* (Jumbotron) dan *list template* yang bisa digunakan dalam membuat Kuis, Formulir dan Survey. *List* template ini ditampilkan dalam bentuk *card*. *List card* ditampilkan berdasarkan kategori Semua, Kuis, Formulir dan Survey. Penggunaan *Tabs* untuk menampilkan kategori *template* dikembangkan dengan Komponen *Tabs* dari *package* `@material-ui/core` dan sebagai navigasi menggunakan *package* `react-router-dom`. Seperti sebelumnya, buat file baru dengan nama `TabsCategory.jsx` kemudian inisiasi kedua package.

```
import Tabs from "@material-ui/core/Tabs";
import Tab from "@material-ui/core/Tab";
import {
  Switch,
  Route,
  NavLink,
  BrowserRouter,
} from "react-router-dom";
```

Selanjutnya menggabungkan file `TabsCategory.jsx` dengan *component* utama `Dashboard.jsx`. Penggabungan *component* dapat dilakukan dengan menggunakan syntax `import` **`namaKomponen`** `from` **`"direktori"`**. Kemudian untuk setiap kategori memiliki componentnya masing-masing sehingga perlu dibuat *file* berdasarkan jumlah dan nama kategori, komponen-komponen ini juga akan di *import* ke *file* `TabsCategory.jsx`. Sedikit berbeda dari implementasi sebelumnya, pada `TabsCategory.jsx` kali ini kita perlu membuat sebuah array yang berisi URL sebagai nilai untuk menginisiasi attribute to, seperti yang dapat dilihat sebagai berikut

```
const allTabs = [
  "/dashboard/home/",
  "/dashboard/home/quiz",
  "/dashboard/home/form",
  "/dashboard/home/survey",
];
```

Lalu untuk implementasi, pertama membungkus Route menggunakan *component* BrowserRouter. Di dalam BrowserRouter akan ada *component* Tabs.

```
<BrowserRouter>
  <div className="wrap-tabs-cate">
    <Route
      path="/dashboard/home/"
      render={({ location }) => (
        <Fragment>
          <Tabs className="nav-tabs-cate" value={location.pathname}
            <Tab
              label="Semua"
              value="/dashboard/home/"
              component={NavLink}
              to={allTabs[0]}
              className="nav-tab-title"
            />
```

Karena TabsCategory berada pada halaman *dashboard* bagian *home* yang URL nya `/dashboard/home` maka `path` dari `Route` diinisiasikan dengan `/dashboard/home`. Untuk *attribute* `render` dibuat *function* yang berisi *props location*. *props location* ini akan menentukan lokasi `Tabs` yang aktif sesuai dengan *pathname* yang diklik. Oleh karena itu *value* pada *component* `Tabs` ditulis `{location.pathname}`, ini akan menampilkan indikator aktif pada `Tab` *item*.

Kemudian di dalam *component* Tabs ada beberapa *component* `Tab` yang bisa dikatakan menu *item*. *Attribute* pada *item* `Tab` *component* `to` yang diinisiasikan dengan *array* sebelumnya sesuai *index item* `Tab` masing-masing. Lalu untuk *attribute component* mengindikasikan bahwa bentuk *component* untuk menuliskan URL adalah *component* `NavLink`. Lalu agar `Tabs` bisa berganti konten yang ditampilkan sesuai nama `Tab` maka digunakan *component* `Route` yang dibungkus *component* `switch` seperti berikut

```
<Switch>
  <Route
    path={allTabs[3]}
    render={() => (
      <div>
        <SurveyTemplate />
      </div>
    )}
  />
```

`Path` pada `Route` ini juga disesuaikan dengan `path` Pada `Tab` `NavLink` sebelumnya. Lalu untuk *attribute* `render`

disini mengindikasikan sebagai konten yang akan ditampilkan. Sehingga hasilnya sebagai berikut

Selanjutnya penerapan React pada dashboard bagian kuis *page*, pada bagian ini dilakukan pengembangan *user interface* untuk pembuatan kuis beserta pilihan *template* kuis. *User* akan dapat membuat kuis dengan pilihan pembuatan dari nol (*blank*) atau juga langsung dapat memilih *template* yang sesuai dengan kebutuhan *user*. *User interface* pada fitur ini akan dibuat *slider card (carousel card)* seperti tampilan pada gambar berikut

Dapat dilihat pada fitur "Buat Baru", tampilan *slider card* dimulai dengan pilihan buat baru dan setelahnya ada beberapa pilihan *template*. Untuk pengembangan tampilan *slider card* menggunakan *package* `react-multi-carousel`. Penggunaan package diawali dengan melakukan `import` seperti berikut

```
import Carousel from "react-multi-carousel";
```

Selanjutnya untuk membuat *slider card responsive* dilakukan dengan membuat `function` dengan nama `responsive`. pada `function responsive` berisi beberapa variabel yang mewakili beberapa bentuk responsivitas antara lain variabel `mobile`, variabel `tablet`, variabel `desktop`, variabel `superLargeDekstop`. Setiap variabel memiliki atribut yang sama dengan nilai yang berbeda-beda. Kedua atribut antara lain adalah `items` dan `breakpoints`, atribut *items* berfungsi untuk mengatur jumlah tampilan *card* yang akan ditampilkan sedangkan untuk `breakpoints` berfungsi untuk mengatur batas lebar minimum dan maximum dari *card* yang ditampilkan.

```
const responsive = {
  superLargeDesktop: {
    breakpoint: { max: 4000, min: 3000 },
    items: 6,
  },
  desktop: {
    breakpoint: { max: 3000, min: 1024 },
    items: 6,
  },
  tablet: {
    breakpoint: { max: 1024, min: 464 },
    items: 2,
  },
  mobile: {
    breakpoint: { max: 464, min: 0 },
    items: 1,
  },
};
```

Kemudian untuk penerapan dengan menuliskan *syntax* sebagai berikut

```
function QuisDashboard() {
  const classes = useStyles();
  return (
    <div className={classes.root}>
      <h3>Buat Baru</h3>
      <Carousel
        responsive={responsive}
        focusOnSelect={true}
        itemClass="carousel-item-padding-900-px"
        className="mt-4"
      >
        <div>
          <Card className="mr-4">
            <img src={newTempImage} />
          </Card>
```

Dapat dilihat bahwasanya `<Corausel/>` memiliki banyak atribut salah satu nya adalah atribut *responsive* sebagai atribut untuk menerapkan `function responsive` yang telah dibuat sebelumnya.

Hasil pembahasan React JS beserta beberapa package yang digunakan sudah sesuai dengan apa yang dipaparkan pada tahap perencanaan dan penerapan. Penerapan React JS selain dapat membantu developer dalam proses pengembangan dengan bantuan antara lain penggunaan package serta penggunaan berbasis *component* yang mempermudah mempercepat efisiensi waktu dalam pengembangan maupun proses debugging. Akan tetapi pada saat ini belum ada implementasi terkait pengujian terhadap *component* yang dibuat. Hal ini perlu diimplementasikan karena dikhawatirkan adanya unit *component* yang dibuat memiliki bug yang tidak terlihat sehingga merubah data dan sebagainya

V. KESIMPULAN DAN SARAN

Berdasarkan hasil yang didapat dari tahap perencanaan dan implementasi dapat disimpulkan bahwa penerapan React JS dalam pengembangan Front end *startup* ubaform yang berbasis *Single Page Application* (SPA) mampu memberikan kemudahan bagi developer serta efisiensi waktu karena penggunaan *package* sehingga developer tidak perlu melakukan *coding* dari nol. Selain itu React JS datang dengan *Reusable component* dan dapat dibuat secara terpisah menjadi beberapa *component* kecil dapat membantu *developer* mengetahui letak *error* terjadi sehingga *debugging* menjadi lebih cepat dan dengan memisahkan setiap *component* menjadi unit-unit kecil ketika dilakukan *debug* tidak akan mengganggu *component* lainya.

Adapun saran untuk meningkatkan pengembangan front end pada *startup* ubaform antara lain diperlukan untuk melakukan *unit testing* pada setiap *component* untuk mengetahui resiko terjadi nya *error* sehingga ketika aplikasi ubaform sudah kompleks nantinya semakin mudah lagi untuk dilakukan *maintenance.*

DAFTAR PUSTAKA

[1] Kulkarni, G., 2012. Cloud Computing-Software as Service. International Journal of Cloud Computing and Services Science (IJ-CLOSER), 1(1).

[2] Alves, R., 2021. What Is a Single Page Application (SPA)?. [online] Outsystems.com. Available at: <https://www.outsystems.com/blog/posts/single-page-application/?utm_source=google&utm_medium=cpc&utm_campaign=Awareness_G_GBL_Search&utm_term=single%20page%20application&utm_content=awareness&gclid=Cj0KCQjwnueFBhChARIsAPu3YkRAQoapRib3-ZPfiXwmxVPQhjMUUtIBKL13-XKaIq5HBGHPkc9QU9caAia8EALw_wcB> [Accessed 7 June 2021]..

[3] HAM, H., 2021. Teknologi Single Page Application (SPA). [online] School of Computer Science. Available at: <https://socs.binus.ac.id/2018/12/06/teknologi-single-page-application-spa/> [Accessed 3 June 2021].

Penerapan React JS Pada Pengembangan FrontEnd Aplikasi Startup Ubaform

ORIGINALITY REPORT

4%	4%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ejournal.unikama.ac.id Internet Source	1%
2	medium.com Internet Source	1%
3	id.reactjs.org Internet Source	<1%
4	myassignmenthelp.com Internet Source	<1%
5	www.hostinger.co.id Internet Source	<1%
6	alifahdiena.wordpress.com Internet Source	<1%
7	adoc.pub Internet Source	<1%
8	core.ac.uk Internet Source	<1%
9	myberitadanpendidikan.blogspot.com Internet Source	<1%

10	rifq221091.wordpress.com Internet Source	<1 %
11	dspace.vutbr.cz Internet Source	<1 %
12	jurnalid.wordpress.com Internet Source	<1 %
13	pt.scribd.com Internet Source	<1 %
14	www.mdpi.com Internet Source	<1 %
15	www.slideshare.net Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		