

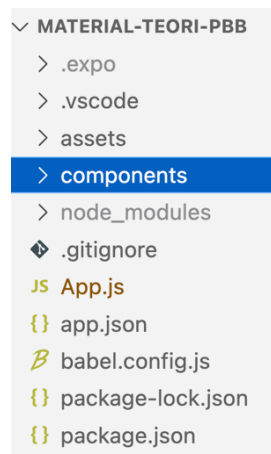


BAB II

REACT FUNDAMENTAL

1. Function Components (RFC) dan Class Components (RCC)

React Componen adalah bit kode yang independen dan dapat digunakan kembali. Komponen pada react memiliki tujuan yang sama dengan fungsi JavaScript, tetapi bekerja secara terpisah dan mengembalikan XHTML. Komponen react terdiri dalam dua jenis, komponen Class dan komponen Fungsi. Untuk membuat file-file komponen buatlah folder bernama components sejajar dengan file App.js.



Gambar 1. Menambahkan folder components

1.1. Class Component (RCC)

Komponen kelas harus menyertakan pernyataan *extends React.Component*. Pernyataan ini mengartikan bahwa class tersebut merupakan warisan untuk *React.Component*, dan memberikan akses komponen kepada fungsi *React.Component*. Komponen juga memerlukan metode *render()*, metode ini mengembalikan XHTML. Setiap component pada react selalu memiliki satu buah class. Berikut adalah contoh penerapan komponen CLASS pada react:



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

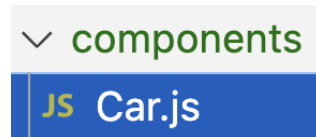
Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA



```
import { Text, View } from 'react-native'
import React, { Component } from 'react'

export class Car extends Component {
  render() {
    return (
      <View>
        <Text>Hi i'm a car</Text>
      </View>
    )
  }
}

export default Car
```

Import library pada JavaScript yang dapat digunakan pada project react. Library ini tersimpan pada folder node_module.

Render sebuah properties yang digunakan untuk menampilkan komponen yang telah terbentuk. Untuk dapat merender membutuhkan sebuah return statement untuk menampilkan ekspresi JSX

JSX JavaScript XML memungkinkan menuliskan syntax HTML kedalam React.

Export Default berfungsi untuk menginformasikan bahwa main programnya ada di komponen tersebut.

constructor() - Dipanggil selama fase konstruksi awal komponen React. Digunakan untuk mengatur status awal dan properti komponen.

Ubahlah script *App.js* untuk menguji file komponen yang telah anda buat dengan cara memanggil komponen class Car dalam bentuk JSX, komponen ini akan menjadi titik awal pada aplikasi anda.



```
JS App.js > ...
1  import Car from './components/Car';
2
3  export default function App() {
4    return (
5      <Car />
6    );
7  }
```

Setelah menambahkan *class Car* sebagai titik awal aplikasi maka tampilan pada simulator akan seperti berikut:



Gambar 1.1 Output React Class Component

1.2. Function Component (RFC)

RFC juga mengembalikan XHTML, dan memiliki sifat yang hampir sama dengan komponen Kelas, tetapi komponen Fungsi dapat ditulis menggunakan lebih sedikit kode, lebih mudah dipahami. Dalam membuat sebuah komponen fungsi pada react, dapat dibangun dalam dua bentuk RFC, berikut adalah contoh penerapan RFC dengan dua tipe scripting. Buatlah file bernama *Motorcycle.js* dan *Bicycle.js* didalam folder *components*, dan masukan script seperti dibawah ini:

Motorcycle.js	Bicycle.js
---------------	------------



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA

```
import { Text, View } from "react-native";
import React from "react";

const Motorcycle = () => {
  return (
    <View>
      <Text>Hi i'm a Motorcycle</Text>
    </View>
  );
};

export default Motorcycle;
```

```
import { View, Text } from "react-native";
import React from "react";

function Bicycle() {
  return (
    <View>
      <Text>Hi i'm a Bicycle</Text>
    </View>
  );
}

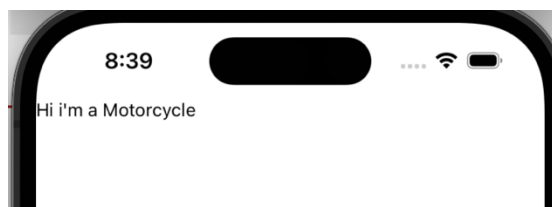
export default Bicycle;
```

Mengubah Titik Awal Aplikasi

```
JS App.js > ...
1 import Motorcycle from './components/Motorcycle';
2
3 export default function App() {
4   return (
5     <Motorcycle />
6   );
7 }
```

```
JS App.js > ...
1 import Bicycle from './components/Bicycle';
2
3 export default function App() {
4   return (
5     <Bicycle />
6   );
7 }
```

Output



1.3. Perbedaan antara RCC dengan RFC

Function Component (RFC)	Class Component (RRC)
Komponen fungsional hanyalah fungsi murni JavaScript biasa yang menerima props sebagai argumen dan mengembalikan elemen React (JSX).	Komponen kelas mengharuskan Anda untuk memperluas dari React. Komponen dan buat fungsi render yang mengembalikan elemen React.
Tidak ada metode render yang digunakan dalam komponen fungsional.	Itu harus memiliki metode render() yang mengembalikan JSX (yang secara sintaksis mirip dengan HTML)
Komponen fungsional berjalan dari atas ke bawah dan setelah fungsi dikembalikan, itu	Komponen kelas dibuat dan metode siklus hidup yang berbeda tetap hidup dan



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA

tidak dapat tetap hidup.	dijalankan dan dipanggil tergantung pada fase komponen kelas.
Juga dikenal sebagai komponen Stateless karena mereka hanya menerima data dan menampilkannya dalam beberapa bentuk, bahwa mereka terutama bertanggung jawab untuk merender UI.	Juga dikenal sebagai komponen Stateful karena mereka menerapkan logika dan keadaan.
Metode React Lifecycle (misalnya, componentDidMount) tidak dapat digunakan dalam komponen fungsional.	Metode React Lifecycle dapat digunakan di dalam komponen kelas (misalnya, componentDidMount).
Hooks dapat dengan mudah digunakan dalam komponen fungsional untuk membuatnya stateful. Contoh: <code>const [name, SetName]= React.useState('')</code>	Ini membutuhkan syntax yang berbeda di dalam komponen kelas untuk mengimplementasikan hooks. Contoh: <code>constructor(props){ super(props); this.state={name:''}</code>
Constructors tidak digunakan.	Constructors digunakan sesuai kebutuhan untuk menyimpan status.

2. Penggunaan State pada RFC dan RCC

State seperti penyimpanan data pribadi milik komponen. State berguna untuk menangani data yang berubah dari waktu ke waktu atau yang berasal dari interaksi pengguna. State akan menyiapkan sebuah memori komponen pada aplikasi.

2.1. Tipe data dan Variable

Tipe data pada Javascript ada tiga buah yaitu `const`, `let`, dan `var`. Berikut ini adalah contoh penggunaan variable terhadap tipe data tersebut:

Name	Scope	Desc
<code>const</code>	Block scope	Berisi nilai tetap dan tidak bisa diubah-ubah
<code>let</code>	Block scope	Nilai dapat diubah
<code>var</code>	Functional scope	Nilai dapat diubah dan diakses diluar block kecuali diluar function

Contoh:

```
const radian = 1  
console.log(radian)
```

```
let isActive = true  
console.log(isActive)
```

```
var total = 10.3  
console.log(total)
```

```
const bulan = 'mei'  
bulan = 'juni'  
console.log(bulan)
```

```
let name = 'Febry'  
console.log(name)
```

```
var obj = {title: 'Pem Perangkat  
Bergerak', id: 1}  
console.log(obj)
```

```
const arrayObj = [{title: 'Pem Perangkat Bergerak', id: 1}, {title: 'Pem Web', id: 2}]  
console.log(arrayObj)
```



2.2. State variable pada RFC

Berikut ini adalah contoh penerapan State pada RFC:

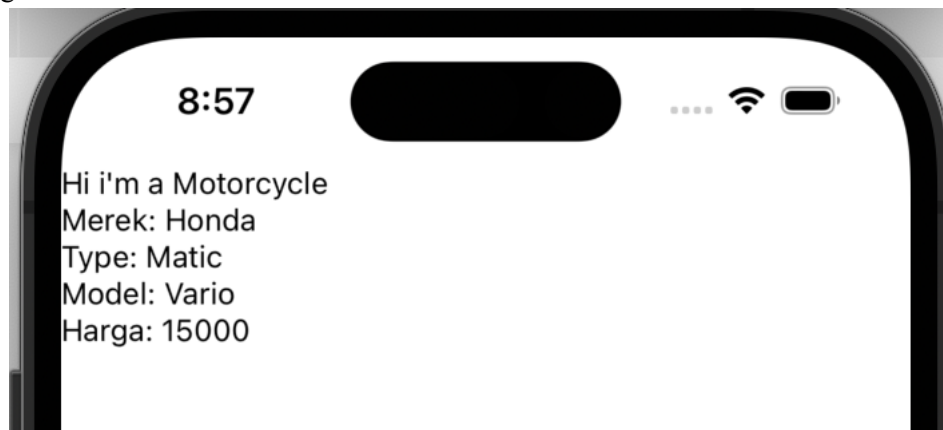
```
import { Text, View } from "react-native";
import React from "react";

var name = "Honda";
const Motorcycle = () => {
  return (
    <View>
      <Text>Hi i'm a Motorcycle</Text>
      <Text>Merek: {name}</Text>
      <Text>Type: {types.type}</Text>
      <Text>Model: {types.model}</Text>
      <Text>Harga: {types.harga}</Text>
    </View>
  );
};

export default Motorcycle;

const types = {type:"Matic", model:"Vario", harga:15000};
```

Pada script diatas jika ingin menampilkan state variable name dan types pada JSX maka harus menggunakan symbol Expression yaitu dengan menambahkan kurung kurawal {...}. Output dari script diatas sebagai berikut:



Gambar 2.2. Output penggunaan state variable pada RFC



2.3.State variable pada RCC

Penerapan state variable pada RCC berbeda dengan bentuk dari RFC. Bentuk penyimpanan data state di RCC lebih complex dan rumit. Berikut adalah contoh penerapan state variable pada RCC:

```
import React, { Component } from 'react'
import { Text, View } from 'react-native'

export class Car extends Component {
  constructor(props){
    super(props);
    this.state={
      merek:"Toyota",
      types:{type:"Matik", model:"Calya ADS"}
    }
  }

  render() {
    return (
      <View>
        <Text>Hi i'm a car</Text>
        <Text>Merek : {this.state.merek}</Text>
        <Text>Type : {this.state.types.type}</Text>
        <Text>Model : {this.state.types.model}</Text>
      </View>
    )
  }
}

export default Car
```

Pada script diatas sama halnya dengan RFC untuk memanggil sebuah state kedalam JSX harus menggunakan symbol Expression yaitu menggunakan kurung kurawal {...}, namun pada RCC pemanggilan state variable harus diawali dengan `this.state.nama_variable`. Berikut adalah contoh dari output script diatas:



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

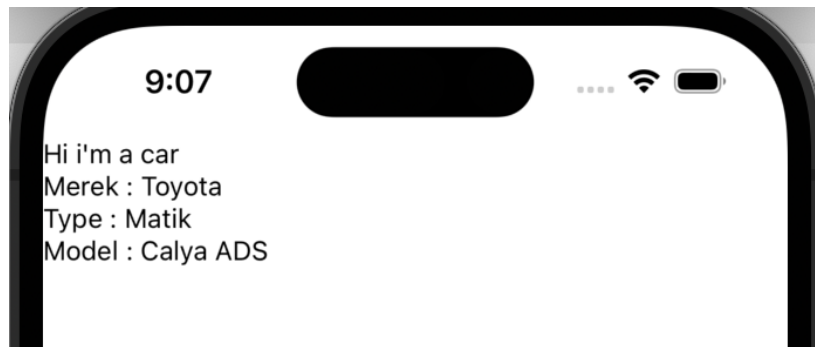
Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA



Gambar 2.3. Output penggunaan state variable pada RCC

2.4. State Function pada RFC

State function JavaScript merupakan blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi JavaScript dijalankan ketika "sesuatu" memanggilnya (memanggilnya). Berikut adalah contoh penggunaan state function pada RFC:

```
import { View, Text } from "react-native";
import React from "react";

function Bicycle() {
  return (
    <View>
      <Text>Hi i'm a Bicycle</Text>
      <TakeARide />
      {Place2Go()}
    </View>
  );
}

export default Bicycle;

const TakeARide = () => {
  return <Text>Let's go riding with me</Text>;
};

function Place2Go() {
  return <Text>We'r going to south west now, come on.</Text>;
}
```

Pada script diatas memiliki sebuah fungsi umum bernama *TakeARide()* dan *Place2Go()*, kedua fungsi tersebut dipanggil didalam JSX dengan dua cara yang berbeda. Untuk fungsi *TakeARide()* dipanggil dengan menggunakan tag XHTML, sedangkan untuk fungsi *Place2Go()* menggunakan *Expression*.



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA



Gambar 2.4.a. Ouput penerapan state function pada RFC

Biasanya dalam sebuah fungsi selalu ada sebuah aksi berupa pengiriman sebuah data atau informasi dalam bentuk variable. Aksi passing parameter yang dapat digunakan pada RFC sebagai berikut:

```
function Bicycle() {  
  const city = "south west";  
  const peoples = [{name:"Erdiana", fams:"Sister"},  
                   {name:"Emanuel", fams:"Brother"},]  
  return (  
    <View>  
      <Text>Hi i'm a Bicycle</Text>  
      <TakeARide peoples={peoples} />  
      {Place2Go(city)}  
    </View>  
  );  
}  
  
export default Bicycle;
```

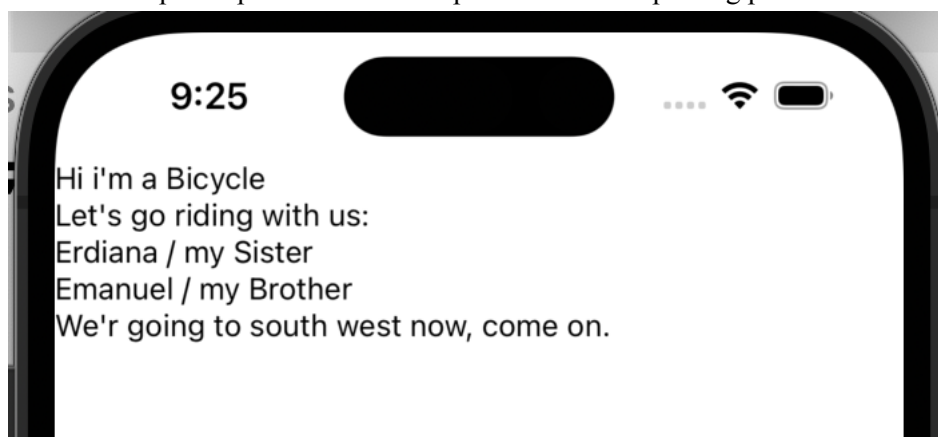


```
const TakeARide = ({peoples}) => {
  return (
    <View>
      <Text>Let's go riding with us:</Text>
      {peoples.map((v,index)=>(
        <View key={index}>
          <Text>{v.name} / my {v.fams}</Text>
        </View>
      ))}
    </View>
  )
};

function Place2Go(value) {
  return <Text>We'r going to {value} now, come on.</Text>;
}
```

Pada script diatas jika mengirimkan sebuah parameter kedalam bentuk XHTML seperti pada function <TakeARide />, parameter tersebut perlu disimpan dalam bentuk attribute. Dan ketika didalam fungsi TakeARide() nilai parameter harus dalam bentuk objek dan penamaannya harus sama dengan nama attribute yang dikirimkan.

Sedangkan pada fungsi Place2Go(), cukup mengirimkannya kedalam isian expression Place2Go("value"). Namun jika yang dikirimkan dalam bentuk objek anda dapat menggunakan symbol bracket objek seperti pada fungsi TakeARide({variable}) atau dapat menggunakan *props*. Berikut adalah contoh penerapan state function pada RFC untuk passing parameters:



Gambar 2.4.b. Output penerapan state function untuk passing parameters

2.5. State Function pada RCC



Berbeda halnya dengan RFC, penerapan sebuah fungsi didalam RCC bisa dikatakan tidak sesimple di RFC. Berikut adalah contoh penerapan fungsi pada RCC:

```
export class Car extends Component {
  constructor(props) {
    super(props);
    this.Come2Go = this.Come2Go.bind(this);
    this.state = {
      merek: "Toyota",
      types: { type: "Matik", model: "Calya ADS" },
    };
  }

  Come2Go(value){
    return (
      <View>
        <Text>Let's go running away from duty</Text>
        <Text>with us only {value} IDR</Text>
      </View>
    )
  }

  render() {
    return (
      <View>
        <Text>Hi i'm a car</Text>
        <Text>Merek : {this.state.merek}</Text>
        <Text>Type : {this.state.types.type}</Text>
        <Text>Model : {this.state.types.model}</Text>
        {this.Come2Go(200000)}
      </View>
    );
  }
}

export default Car;
```

Dari script diatas untuk membuat sebuah fungsi di RCC, pertama kali yang perlu diperhatikan ialah menginisialisasi nama sebuah fungsi didalam `constructor()`. Dengan menuliskan script seperti berikut `this.Come2Go = this.Come2Go.bind(this)`. Didalam RCC untuk membuat sebuah function harus berada di bawah constructor dan bentuknyapun cukup dengan *NameOfFunction()*. Sedangkan untuk memanggil sebuah fungsi di RCC, sama halnya dengan state variable, kita perlu memanggilnya dengan bentuk Expression dengan diawali `this.NameOfFunction()`.

3. Core Component UI dalam JSX

3.1. Container UI



Container UI yang dapat digunakan untuk membangun sebuah output pada aplikasi terdiri dari View, SafeAreaView, ScrollView, dan ImageBackground. Penerapan container ini sama halnya seperti element block pada HTML yaitu <div>.

a. View

Container ini dapat digunakan lebih dari satu kali dalam JSX, ini membantu untuk membungkus tag JSX yang bersifat multiple element.

```
render() {  
  return (  
    <View>  
      <View>  
        <Text>Hi i'm a car</Text>  
        <Text>Merek : {this.state.merek}</Text>  
      </View>  
      <View>  
        <Text>Type : {this.state.types.type}</Text>  
        <Text>Model : {this.state.types.model}</Text>  
      </View>  
      {this.Come2Go(200000)}  
    </View>  
  );  
}
```

b. SafeAreaView

Container JSX dalam bentuk ini hanya digunakan hanya sekali dalam aplikasi, biasanya diinisialisasi didalam root atau titik awal pada component react. Tag JSX ini berguna untuk memposisikan layer UI mengikuti batas StatusBar. Jika tidak menggunakan JSX ini maka start awal layer akan berada di titik X dan Y sama dengan 0.

```
export default function App() {  
  return (  
    <SafeAreaView>  
      <Car />  
    </SafeAreaView>  
  );  
}
```

c. ImageBackground

Sama halnya dengan container SafeAreaView, container ini hanya dapat digunakan sekali saja dalam inisialisasinya. Container ini diperuntukan jika ingin memiliki background gambar pada aplikasinya.



```
export default function App() {
  return (
    <ImageBackground
      source={{
        url: "https://kis.ibik.ac.id/environment/ibik/images/background.jpg",
      }}
      resizeMode="cover"
      style={{ flex: 1 }}
    >
      <SafeAreaView>
        <Car />
      </SafeAreaView>
    </ImageBackground>
  );
}
```

d. ScrollView

Jika anda ingin memiliki layout dalam bentuk scrolling anda dapat menggunakan container ini sebagai bentuk JSX yang digunakan pada elemen yang memiliki data atau informasi yang panjang.

```
return (
  <ScrollView>
    <View>
      <Text>Hi i'm a car</Text>
      <Text>Merek : {this.state.merek}</Text>
    </View>
    <View>
      <Text>Type : {this.state.types.type}</Text>
      <Text>Model : {this.state.types.model}</Text>
    </View>
    {this.Come2Go(200000)}
  </ScrollView>
);
```



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

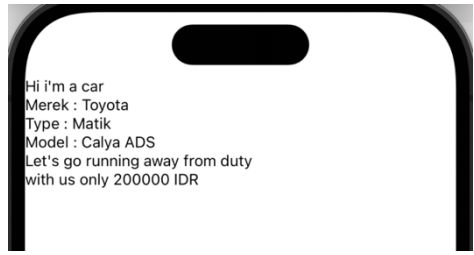
Kelas

: TI-21-PA

3.2. Basic UI

a. StatusBar

Elemen JSX `<StatusBar />` ini untuk mengontrol status aplikasi seperti zona status, biasanya berada pada bagian atas layar, yang menampilkan waktu, Wi-Fi dan informasi jaringan seluler, level baterai, dan/atau ikon status lainnya.

Code	Output
<pre>export default function App() { return (<SafeAreaView> <StatusBar hidden={true} /> <Car /> </SafeAreaView>); }</pre>	

Berikut adalah attribute yang dapat digunakan untuk element StatusBar:

Attribute	Value
backgroundColor	Color, default 'black'
hidden	boolean

b. Text

Sebuah element JSX untuk menampilkan sebuah tulisan kedalam aplikasi.

```
<Text>Hi i'm a car</Text>
```

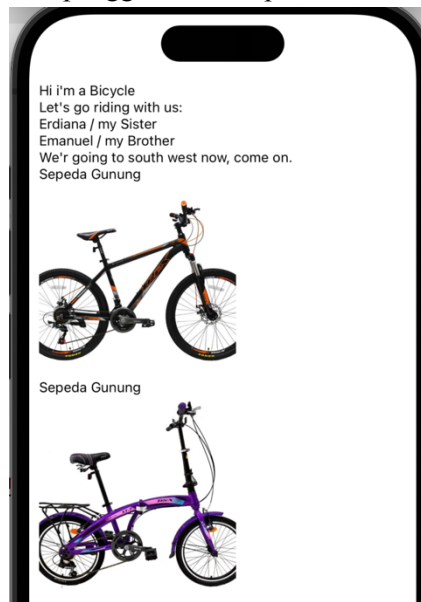
c. Image

Berikutnya ialah JSX UI untuk menampilkan berbagai jenis gambar, termasuk gambar jaringan, sumber daya statis, gambar lokal sementara, dan gambar dari disk lokal, seperti rol kamera.

```
return (
  <View style={{padding:10}}>
    <Text>Hi i'm a Bicycle</Text>
    <TakeARide peoples={peoples} />
    {Place2Go(city)}
    <View>
      <Text>Sepeda Gunung</Text>
      <Image
        source={{
          uri: "https://trexsporting.com/images/products/11-KbmXViHodZ.jpg",
        }}
        style={{width:200, height:200}}
      />
    </View>

    <View>
      <Text>Sepeda Gunung</Text>
      <Image
        source={require("../assets/icons/sepeda-lipat.jpeg")}
        style={{width:200, height:200}}
      />
    </View>
  </View>
);
}
```

Berikut adalah contoh output dari penggunaan komponen ui Image:



Gambar 3.2.c. Output penggunaan komponen UI Image

3.3. Forms UI

a. TextInput

Komponen dasar untuk memasukkan teks ke dalam aplikasi melalui keyboard. Komponen ini memiliki property konfigurasi untuk beberapa fitur, seperti koreksi otomatis, kapitalisasi otomatis, teks placeholder, dan jenis keyboard yang berbeda, seperti keypad numerik.



Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA

Code	Output
<pre><View> <Text>Student ID:</Text> <TextInput style={styles.inputText} placeholder="Enter your NPM" keyboardType="numeric" /> </View> <View> <Text>Fullname:</Text> <TextInput style={styles.inputText} placeholder="Enter your name here" /> </View> <View> <Text>Address:</Text> <View> <TextInput editable multiline numberOfLines={4} maxLength={40} /> </View> </View></pre>	

b. Buttons

Komponen tombol dasar yang seharusnya dirender dengan baik di platform apa pun. Mendukung tingkat penyesuaian minimal.

```
<Button
  title="Button form OS"
/>
```

Sebelum mencoba beberapa bentuk dari Touchable buatlah sebuah fungsi umum seperti dibawah ini:

```
const buttonAct = (title) => {
  return (
    <View
      style={{
        backgroundColor: "purple",
        borderRadius: 10,
        padding: 10,
        alignItems: "center",
        marginVertical: 5,
      }}
    >
      <Text style={{ color: "white" }}>{title}</Text>
    </View>
  );
};
```




Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

IBIK

Kelas

: TI-21-PA

- **TouchableOpacity**

Pembungkus untuk membuat tampilan merespons sentuhan dengan benar. Saat ditekan, opasitas tampilan terbungkus berkurang, meredupkannya.

```
<TouchableOpacity
  activeOpacity={0.6}
>
  {buttonAct("Touchable Opacity")}
</TouchableOpacity>
```

- **TouchableHighlight**

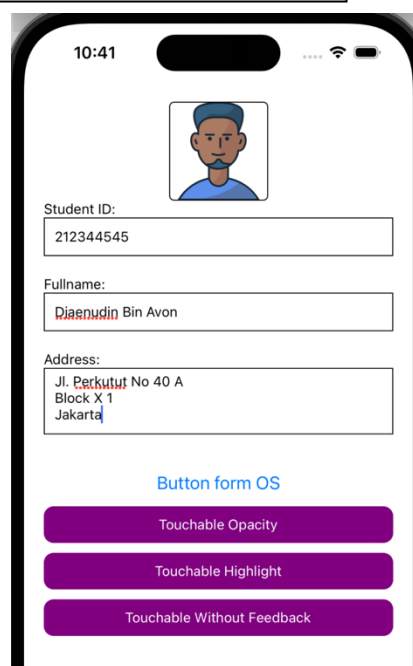
Pembungkus untuk membuat tampilan merespons sentuhan dengan benar. Saat ditekan, opasitas tampilan yang dibungkus akan berkurang, yang memungkinkan warna lapisan bawah terlihat, menggelapkan, atau mewarnai tampilan.

```
<TouchableHighlight
  activeOpacity={0.6}
>
  {buttonAct("Touchable Highlight")}
</TouchableHighlight>
```

- **TouchableWithoutFeedback**

Jangan gunakan kecuali Anda memiliki alasan yang sangat bagus. Semua elemen yang merespons pers harus memiliki umpan balik visual saat disentuh.

```
<TouchableWithoutFeedback
  activeOpacity={0.6}
>
  {buttonAct("Touchable Without Feedback")}
</TouchableWithoutFeedback>
```





Matakuliah/Code

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3

Dosen

: Irvan Rizky Ariansyah / Raiyana Jan Winata

I B I K

Kelas

: TI-21-PA

Gambar 3.3. Output komponen UI Forms

3.4. Stylesheet

4. Latihan Praktikum

Pengumpulan tugas Latihan praktikum dikumpulkan kedalam GITHUB masing-masing mahasiswa berdasarkan repository yang telah dibuat PPB-TI-21-[PA/KA]-NPM. File source code disimpan sesuai nama project-praktikum dan masukan kedalam repositori 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 kepada:

@FebryFairuz dan (@IrvanRizkyAriansyah atau @raiyanawinata)

Disusun Oleh		Disetujui Oleh
<u>Irvan Rizky Ariansyah</u> Penyusun I	<u>Raiyana Jan Winata</u> Penyusun II	<u>Febri Damatraseta Fairuz, S.T., M.Kom</u> Dosen Kordinator