



Buku Pembelajaran

# PEMROGRAMAN PERANGKAT BERGERAK

- Febri Damatraseta Fairuz • Septian Cahyadi
- Isnan Mulia • Suci Sutjipto • Thesya Marcella

## **KATA PENGANTAR**

Puji syukur ke hadirat Tuhan Yang Maha Kuasa, yang telah memberikan rahmat-Nya sehingga Buku Pembelajaran Pemrograman Perangkat Bergerak untuk mahasiswa/i Program Studi Teknologi Informasi Fakultas Informatika dan Pariwisata Institut Bisnis dan Informatika Kesatuan Bogor dapat diselesaikan dengan sebaik-baiknya.

Buku Pembelajaran ini dibuat sebagai panduan dalam melakukan kegiatan Pembelajaran Pemrograman Perangkat Bergerak yang merupakan kegiatan penunjang mata kuliah pada Program Studi Teknologi Informasi Fakultas Informatika dan Pariwisata Institut Bisnis dan Informatika Kesatuan Bogor. Buku ini diharapkan dapat membantu mahasiswa/i dalam mempersiapkan dan melaksanakan Pembelajaran dengan lebih baik, terarah, dan terencana. Pada setiap topik telah ditetapkan tujuan pelaksanaan Pembelajaran dan semua kegiatan yang harus dilakukan oleh mahasiswa/i serta teori singkat untuk memperdalam pemahaman mengenai materi yang dibahas.

Dalam pembuatan Buku Pembelajaran Pemrograman Perangkat Bergerak ini, penyusun menyadari masih jauh dari sempurna. Oleh karena itu, penyusun mengharapkan kritik dan saran yang membangun guna penyempurnaan buku ini di masa yang akan datang. Akhir kata, penyusun mengucapkan banyak terima kasih kepada semua pihak yang telah terlibat baik secara langsung maupun tidak langsung dalam pembuatan Buku ini. Semoga buku ini dapat memberikan manfaat dan menjadi panduan yang bermanfaat bagi mahasiswa/i dalam mempelajari dan mengembangkan aplikasi perangkat bergerak.

Penyusun

## DAFTAR ISI

KATA PENGANTAR .....	2
DAFTAR ISI.....	3
DAFTAR GAMBAR.....	5
BAB 1 INTRODUCTION .....	7
1.1 Tujuan Pembelajaran .....	7
1.2 Dasar Teori.....	7
1.2.1 Environment Development .....	7
1.2.2 Create Project.....	8
1.2.3 EXPO Cli .....	12
1.3 Latihan Pembelajaran.....	15
BAB 2 REACT FUNDAMENTAL.....	16
2.1 Tujuan Pembelajaran .....	16
2.2 Dasar Teori.....	16
2.2.1 Function Components (RFC) dan Class Components (RCC) .....	16
2.2.2 Pengunaan State Pada RFC dan RCC .....	20
2.2.3 Core Component UI dan JSX .....	27
2.3 Latihan Pembelajaran.....	45
BAB 3 LISTING DAN LIFE CYCLE MODE.....	47
3.1 Tujuan Pembelajaran .....	47
3.2 Dasar Teori.....	47
3.2.1 Install Library.....	47
3.2.2 StyleSheet – Daftar Teman UI.....	47
3.2.3 Listing .....	53
3.3 Latihan Pembelajaran.....	58
BAB 4 NAVIGATION BAR.....	59
4.1 Tujuan Pembelajaran .....	59
4.2 Dasar Teori.....	59
4.2.1 Install Library.....	59

4.2.2 Stack.....	60
4.2.3 Bottom Tabs Navigations .....	62
4.3 Latihan Pembelajaran.....	64
<b>BAB 5 LIFECYCLE METHOD REACT.....</b>	<b>65</b>
5.1 Tujuan Pembelajaran .....	65
5.2 Dasar Teori.....	65
5.2.1 Lifecycle Component API's .....	65
5.2.2 HOOK.....	68
5.2.3 Operations .....	70
5.3 Latihan Pembelajaran.....	83

## DAFTAR GAMBAR

Gambar 1. 1 Menu Edit System Environment	7
Gambar 1. 2 Environment Variable	8
Gambar 1. 3 Install & create project di cmd	9
Gambar 1. 4 Struktur folder project react	9
Gambar 1. 5 Running project React Native CLI	10
Gambar 1. 6 App starting on ios	11
Gambar 1. 7 komponen dasar pada react native	11
Gambar 1. 8 melihat versi expo	12
Gambar 1. 9 membuat projek expo pada terminal	12
Gambar 1. 10 Struktur projek React Native Expo	12
Gambar 1. 11 Menjalankan Expo	13
Gambar 2. 1 Menambahkan folder components	15
Gambar 2. 2 Contoh Penerapan Class pada React	16
Gambar 2. 3 Memanggil komponen class dalam bentuk JSX	17
Gambar 2. 4 Output React Class Component	17
Gambar 2. 5 Tipe data dan variabel	20
Gambar 2. 6 Contoh penerapan State pada RFC	20
Gambar 2. 7 Output penggunaan state variable pada RFC	20
Gambar 2. 8 Penerapan state variable pada RCC	21
Gambar 2. 9 Output penggunaan state variable pada RC	21
Gambar 2. 10 Penerapan state function pada RFC	22
Gambar 2. 11 Output penerapan state function pada RFC	22
Gambar 2. 12 Penerapan passing parameter pada RFC	23
Gambar 2. 13 Output penerapan state function untuk passing parameters	24
Gambar 2. 14 Penerapan fungsi pada RCC	25
Gambar 2. 15 Penggunaan container view	26
Gambar 2. 16 Penggunaan container SafeAreaView	26
Gambar 2. 17 Penggunaan container image background	27
Gambar 2. 18 Penggunaan container ScrollView	27
Gambar 2. 19 Penggunaan komponen UI Image	29
Gambar 2. 20 Output penggunaan komponen UI Image	29
Gambar 2. 21 Output komponen UI Forms	32
Gambar 2. 22 Penerapan stylesheet inline	33
Gambar 2. 23 Output penerapan stylesheet inline	33
Gambar 2. 24 Penerapan stylesheet embed	34
Gambar 2. 25 penerapan stylesheet embed	34
Gambar 2. 26 Penerapan flex pada stylesheet	35

Gambar 2. 27 Output penerapan Flex pada stylesheet	36
Gambar 2. 28 Flex Direction	36
Gambar 3. 1 Desain halaman daftar pertemanan	45
Gambar 3. 2 Break Down UI halaman daftar teman	45
Gambar 3. 3 Struktur folder	46
Gambar 3. 4 Sketsa Scripting MyFriend.js dengan class	46
Gambar 3. 5 Output skema awal	47
Gambar 3. 6 Script SearchBar dengan icon dan font icon	48
Gambar 3. 7 memanggil fungsi searchBar pada MyFriend.js	49
Gambar 3. 8 Output widget SearchBar	49
Gambar 3. 9 script UserItem.js	51
Gambar 3. 10 Script dan Output Penggunaan ScrollView	52
Gambar 3. 11 Script dan Output Penggunaan FlatList	53
Gambar 3. 12 Script dan Output Penerapan SectionList	54
Gambar 4. 1 Penggunaan Stack pada App.js	57
Gambar 4. 2 Function component untuk Page 1 & 2	58
Gambar 4. 3 Contoh transisi antar layar	59
Gambar 4. 4 Script penerapan perpindahan layar dengan Bottom Tabs Navigation	59
Gambar 4. 5 Contoh transisi dengan bottom tab navigations	59
Gambar 5. 1 Penggunaan Mounting	62
Gambar 5. 2 Penggunaan state lifecycle	63
Gambar 5. 3 Tampilan output state lifecycle	63
Gambar 5. 4 Penggunaan Hook	65
Gambar 5. 5 Tampilan penggunaan HOOK	66
Gambar 5. 6 Tampilan komponen Sign In	66
Gambar 5. 7 Skema dari RFC Sign In	68

# BAB 1 INTRODUCTION

## 1.1 Tujuan Pembelajaran

Tujuan pembelajaran pada bab ini adalah memberikan pemahaman dan keterampilan dasar kepada mahasiswa dalam menginstal, mengkonfigurasi lingkungan pengembangan React Native, serta mengenal proses pembuatan proyek baru. Diharapkan mahasiswa dapat:

1. Memahami langkah instalasi dan konfigurasi lingkungan pengembangan React Native.
2. Mengidentifikasi persyaratan dan dependensi yang diperlukan untuk mengembangkan aplikasi React Native.
3. Mengunduh dan menginstal perangkat lunak dan framework yang relevan untuk memulai proyek React Native.
4. Membuat proyek React Native baru menggunakan Expo atau React Native CLI.

## 1.2 Dasar Teori

### 1.2.1 Environment Development

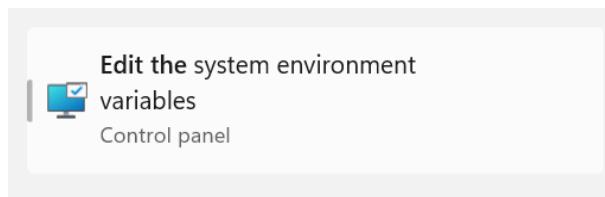
#### 1. Android Studio

##### Step 1 : Download File Android Studio

Android Studio merupakan lingkungan pengembangan untuk aplikasi React Native yang dapat diunduh melalui link <https://developer.android.com/studio/>. Pastikan memenuhi persyaratan sistem sebelum melakukan instalasi.

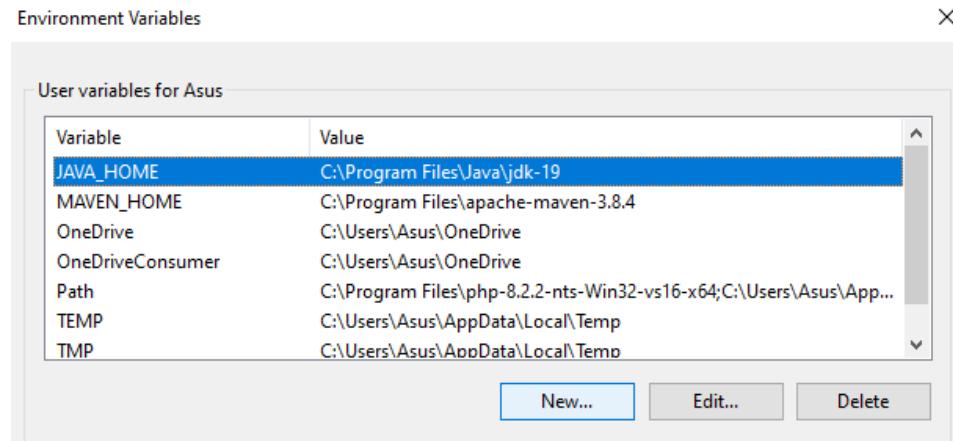
##### Step 2 : Mengatur Environment Variable Android pada OS Windows

Buka tab “search”, lalu buka “edit the system environment variables”



Gambar 1.1 Menu Edit System Environment

Klik “New..” .. dan buat variabel baru dengan nama "ANDROID\_HOME" dan isi dengan path di mana Anda menginstal Android SDK. Lalu klik OK.



Gambar 1.2 Environment Variable

## 2. Java JDK

. Java JDK dibutuhkan untuk pengembangan aplikasi mobile menggunakan React Native. Anda dapat menginstall JAVA JDK dengan mengunduhnya melalui link:

<https://www.oracle.com/java/technologies/downloads/>

## 3. Node js

Node.js diperlukan untuk mengembangkan aplikasi React Native. Anda dapat mengunduh Node.js melalui <https://nodejs.org/download> kemudian install. Kemudian cek apakah sudah terinstall dengan benar dengan mengetik syntax berikut pada terminal

```
C:\Users\LENOVO>node -v  
v18.14.2
```

### 1.2.2 Create Project

#### A. Install React Native CLI and Create Project

Setelah berhasil menginstal dependency NodeJS:

1. Buka aplikasi terminal atau command prompt pada OS.
2. Masuk kedalam directory path untuk menyimpan Project React.
3. Setelah berhasil masuk kedalam directory path tersebut masukkan syntax ini untuk menginstall react-native dan membuat project.

```
npx react-native init MyFirstProjectReact
```

The terminal window shows the following sequence of events:

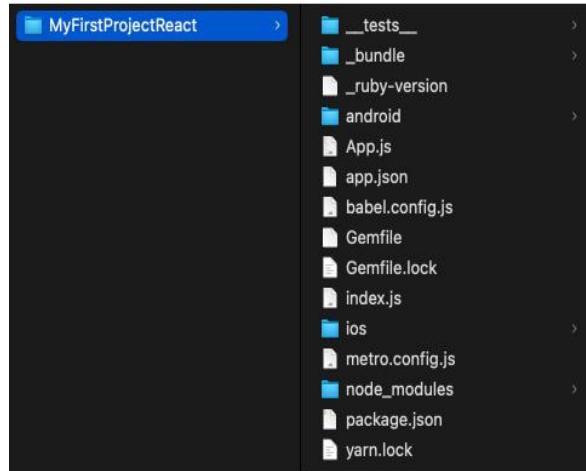
- A welcome message from the React Native CLI: "Welcome to React Native! Learn once, write anywhere".
- A list of completed steps:
  - ✓ Downloading template
  - ✓ Copying template
  - ✓ Processing template
  - ✓ Installing CocoaPods dependencies (this may take a few minutes)
- Instructions for Android:
  - Have an Android emulator running (quickest way to get started), or a device connected.
  - cd "/Users/febryfairuz/Documents/IBIK/2022-2023/Genap/Pem. Perangkat Bergerak/Workspace/MyFirstProjectReact" && npx react-native run-android
- Instructions for iOS:
  - cd "/Users/febryfairuz/Documents/IBIK/2022-2023/Genap/Pem. Perangkat Bergerak/Workspace/MyFirstProjectReact" && npx react-native run-ios
  - OR
  - Open MyFirstProjectReact/ios/MyFirstProjectReact.xcworkspace in Xcode or run "xed -b ios"
  - Hit the Run button
- Instructions for macOS:
  - See <https://aka.ms/ReactNativeGuideMacOS> for the latest up-to-date instructions.

Gambar 1. 3 Install & create project di cmd

4. Maka pada tampilan terminal akan mendownload dan menginstal react-native setelah terpasang selanjutnya akan membuat project bernama **MyFirstProjectReact**.

## B. Struktur Project React Native CLI

Pada gambar dibawah terdapat dua buah folder bernama android dan ios. Jika kalian buka pada masing-masing folder tersebut maka akan berisi project native dari os tersebut. Sedangkan sisanya adalah struktur milik ReactNative dimana bahasa pemrograman yang digunakan ialah Javascript.



Gambar 1. 4 Struktur folder project react

## C. Running Project React Native CLI

### 1. Starting Metro

Buka terminal dan masuk kedalam directory path Project React yang telah dibuat dan masukan syntax : npx react-native start

```
MyFirstProjectReact — node - npn exec react-native start THM04H-1vaM0d9f313...
ReactNativeClient[145]: react-native start
react-native: starting packager
Metro Bundler for Webpack
Packager is listening at http://localhost:19001
To reload the app press 'r'
To open developer menu press 'h'
```

Gambar 1. 5 Running project React Native CLI

Jangan tutup terminal ini, karena harus menjalankan metro, bundle JavaScript yang disertakan dengan React Native.

## 2. Starting Application on Android

Buka terminal baru dan masuk kedalam directory path Project React yang telah dibuat dan masukan syntax dibawah ini:

```
npx react-native run-android
```

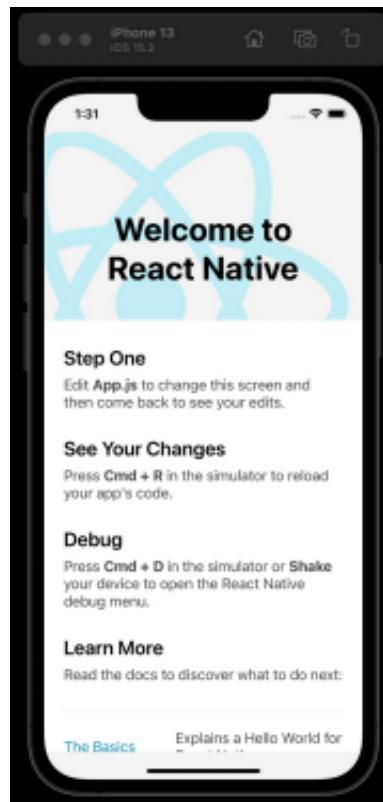
## 3. Starting Application on IOS

Untuk ios disarankan untuk menginstall watchman dan cocoapods terlebih dahulu. Buka terminal baru dan masuk kedalam directory path Project React yang telah dibuat dan masuk kedalam folder ios masukan syntax :

```
pod install
```

Lalu keluar dari folder ios dan ketik syntax:

```
npx react-native run-ios
```



Gambar 1. 6 App starting on ios

## D. Basic Components

- **View** Komponen paling mendasar untuk membangun UI
- **Text** Komponen untuk menampilkan teks.
- **Text Input** Komponen untuk memasukkan teks ke dalam aplikasi melalui keyboard.
- **StyleSheet** Menyediakan lapisan abstrak yang mirip dengan stylesheet CSS.

```
✓ import { StatusBar } from 'expo-status-bar';
✓ import { StyleSheet, Text, View } from 'react-native';

✓ export default function App() {
✓   return [
✓     <View style={styles.container}>
✓       <Text>Open up App.js to start working on your app!</Text>
✓       <StatusBar style="auto" />
✓     </View>
✓   ];
✓ }

✓ const styles = StyleSheet.create({
✓   container: {
✓     flex: 1,
✓     backgroundColor: '#fff',
✓     alignItems: 'center',
✓     justifyContent: 'center',
✓   },
✓});
```

Gambar 1. 7 komponen dasar pada react native

### 1.2.3 EXPO Cli

#### A. Install React Native EXPO CLI and Create Project

##### Step 1: Install Expo

Buka terminal dan masukkan syntax: npm install -g expo-cli

Jika sudah berhasil maka dapat dicek dengan masukkan syntax expo -V

```
C:\Users\Asus>expo -V
5.3.0
```

Gambar 1. 8 melihat versi expo

##### Step 2: Create Project Expo

Buka terminal dan masukkan syntax:

- expo init PEMBELAJARAN-PPB
- cd PEMBELAJARAN-PPB
- npm start

```

C:\Windows\system32\cmd.exe
D:\KULIAH\SEM 6\asdoss\ppb>expo init PPB-21-TI-KA-202310018
WARNING: expo-cli has not yet been tested against Node.js v19.6.0.
If you encounter any issues, please report them to https://github.com/expo/expo-cli/issues

expo-cli supports following Node.js versions:
* >=12.13.0 <13.0.0 (Maintenance LTS)
* >=14.0.0 <15.0.0 (Active LTS)
* >=15.0.0 <17.0.0 (Current Release)

There is a new version of expo-cli available (6.3.0).
You are currently using expo-cli 5.3.0
Install expo-cli globally using the package manager of your choice;
for example: npm install -g expo-cli to get the latest version

Choose a template: » blank           a minimal app as clean as an empty canvas
Downloaded template.
Using Yarn to install packages. Pass --npm to use npm instead.
Installed JavaScript dependencies.

Your project is ready!

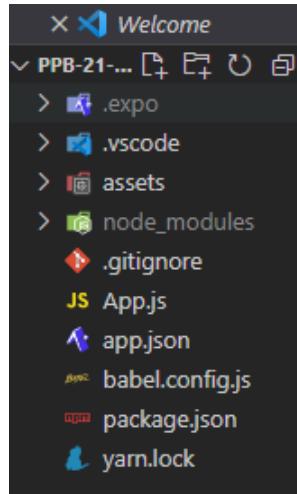
To run your project, navigate to the directory and run one of the following yarn commands.

cd PPB-21-TI-KA-202310018
yarn start # you can open iOS, Android, or web from here, or run them directly with the commands below.
yarn android
yarn ios # requires an iOS device or macOS for access to an iOS simulator

```

*Gambar 1. 9 membuat projek expo pada terminal*

## B. Struktur Project React Native EXPO



*Gambar 1. 10 Struktur projek React Native Expo*

Di direktori root, **App.js** adalah file yang memulai proyek. Lalu ada folder aset di mana **aset statis** dapat ditempatkan. Akhirnya, ada beberapa file konfigurasi dan manajemen dependensi seperti **app.json**, **babel.config.js**, **package.json**, dll.

### Step 3: Running Expo

**Note:** Jika menggunakan EXPO install dan daftar terlebih dahulu Expo go di smartphone masing-masing.

Setelah menjalankan “npm start” maka terminal akan menampilkan seperti dibawah ini:

```
npm start
Starting Metro Bundler

> Metro waiting on exp://192.168.119.164:19000
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
```

*Gambar 1. 11 Menjalankan Expo*

Jangan ditutup untuk terminal ini. Pada terminal tertulis address yang dapat kita akses melalui browser dengan cara menuliskan “<http://localhost:19002>”. Jika menggunakan EXPO go, terdapat bentuk BARCODE yang dapat kita scan untuk terhubung pada smartphone.

### **1.3 Latihan Pembelajaran**

Buatlah Project dan Buatlah tampilan sederhana untuk menampilkan MyProfile

## BAB 2 REACT FUNDAMENTAL

### 2.1 Tujuan Pembelajaran

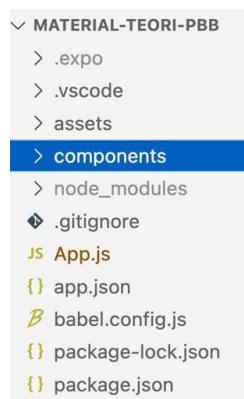
Tujuan pembelajaran pada bab ini adalah memberikan pemahaman tentang konsep dasar React dan komponen-komponen utamanya, terutama Function Components (RFC) dan Class Components (RCC). Diharapkan mahasiswa dapat:

1. Memahami perbedaan antara Function Components (RFC) dan Class Components (RCC).
2. Menguasai penggunaan State untuk mengelola data dan perubahan dalam aplikasi React, baik pada Function Components maupun Class Components.
3. Menguasai penggunaan Core Component UI dalam pembuatan tampilan antarmuka dengan JSX.
4. Mampu menggunakan berbagai jenis TouchableOpacity, TouchableHighlight, dan TouchableWithoutFeedback dalam menangani interaksi pengguna.
5. Memahami penggunaan Stylesheet dan mengatur layout dengan React.

### 2.2 Dasar Teori

#### 2.2.1 Function Components (RFC) dan Class Components (RCC)

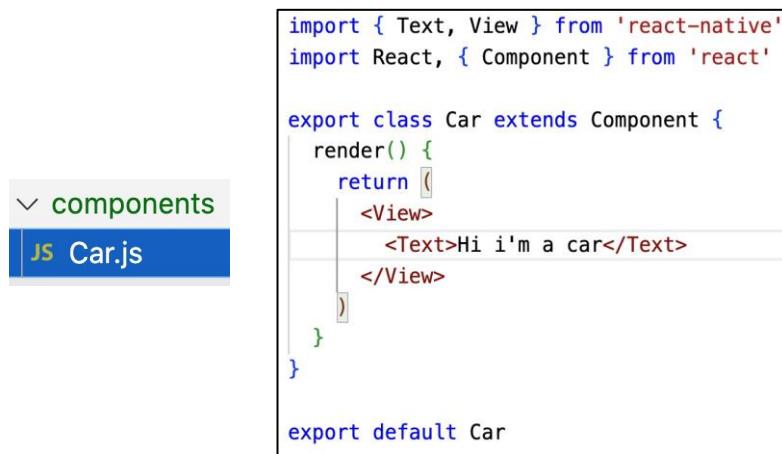
React Component adalah bit kode yang independen dan dapat digunakan kembali. Komponen pada react memiliki tujuan yang sama dengan fungsi JavaScript, tetapi bekerja secara terpisah dan mengembalikan XHTML. Komponen react terdiri dalam dua jenis, komponen Class dan komponen Fungsi. Untuk membuat file-file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



Gambar 2. 1 Menambahkan folder components

## A. 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 pada 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:



The image shows a code editor interface. On the left, there is a file tree with a 'components' folder expanded, showing a 'Car.js' file. The 'Car.js' file is selected and highlighted with a blue background. The code itself is as follows:

```
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
```

Gambar 2. 2 Contoh Penerapan Class pada React

**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 expresi 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 }
```

Gambar 2. 3 Memanggil komponen class dalam bentuk JSX

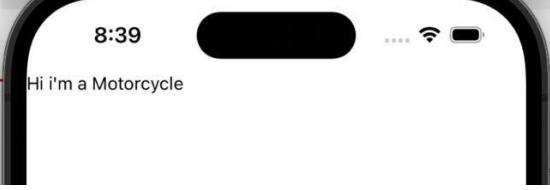
Setelah menambahkan *class Car* sebagai titik awal aplikasi maka tampilan pada simulator akan seperti berikut:



Gambar 2. 4 Output React Class Component

## B. 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 di dalam folder components, dan masukan script seperti dibawah ini:

Motorcycle.js	Bicycle.js
<pre>import { Text, View } from "react-native"; import React from "react";  const Motorcycle = () =&gt; {   return (     &lt;View&gt;       &lt;Text&gt;Hi i'm a Motorcycle&lt;/Text&gt;     &lt;/View&gt;   ); }  export default Motorcycle;</pre>	<pre>import { View, Text } from "react-native"; import React from "react";  function Bicycle() {   return (     &lt;View&gt;       &lt;Text&gt;Hi i'm a Bicycle&lt;/Text&gt;     &lt;/View&gt;   ); }  export default Bicycle;</pre>
<b>Mengubah Titik Awal Aplikasi</b>	
<pre>JS App.js &gt; ... 1 import Motorcycle from './components/Motorcycle'; 2 3 export default function App() { 4   return ( 5     &lt;Motorcycle /&gt; 6   ); 7 }</pre>	<pre>JS App.js &gt; ... 1 import Bicycle from './components/Bicycle'; 2 3 export default function App() { 4   return ( 5     &lt;Bicycle /&gt; 6   ); 7 }</pre>
<b>Output</b>	
	

### C. Perbedaan antara RCC dengan RFC

Function Component (RFC)	Class Component (RCC)
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 tidak dapat tetap hidup.	Komponen kelas dibuat dan metode siklus hidup yang berbeda tetap hidup dan 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	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: const [name, SetName]=React.useState("")	Ini membutuhkan syntax yang berbeda di dalam komponen kelas untuk mengimplementasikan hooks. Contoh: constructor(props){ super(props); this.state={name:""}}
Constructors tidak digunakan.	Constructors digunakan sesuai kebutuhan untuk menyimpan status.

### 2.2.2 Pengunaan 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.

#### Tipe data dan Variable

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

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

Contoh:

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

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

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

let isActive = true
console.log(isActive)

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

var total = 10.3
console.log(total)

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

Gambar 2. 5 Tipe data dan variabel

### A. State variable pada RFC

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

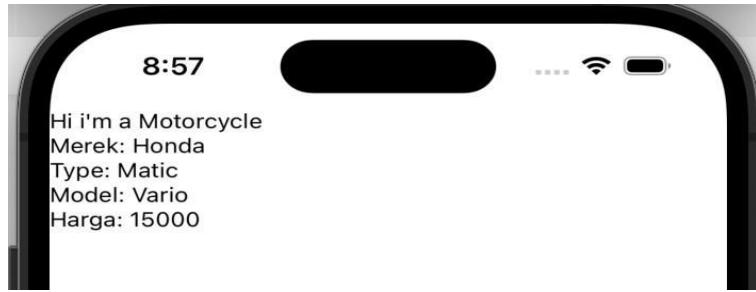
var name = "Honda";
const Motorcycle = () => {
  return (
    <View>
      <Text>Hi i'm a Motorcycle</Text>
      <Text>Merek: {name}</Text>
      <Text>Type: {types.type}</Text>
      <Text>Model: {types.model}</Text>
      <Text>Harga: {types.harga}</Text>
    </View>
  );
};

export default Motorcycle;

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

Gambar 2. 6 Contoh penerapan State pada RFC

untuk menampilkan state variable name dan *types* pada JSX gunakan symbol Expression dengan menambahkan kurung kurawal {...}. Output dari script diatas sebagai berikut:



Gambar 2. 7 Output penggunaan state variable pada RFC

## B. 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'

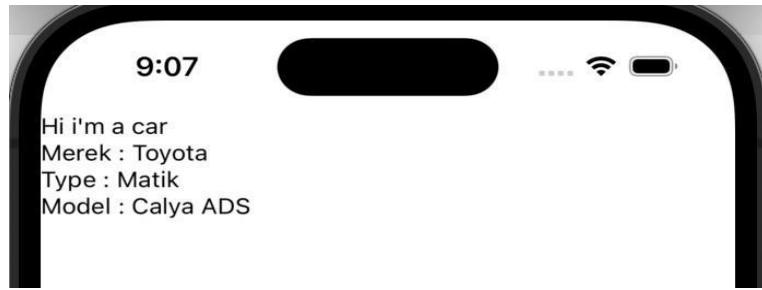
export class Car extends Component {
  constructor(props){
    super(props);
    this.state={
      merek:"Toyota",
      types:{type:"Matik", model:"Calya ADS"}
    }
  }
  render() {
    return (
      <View>
        <Text>Hi i'm a car</Text>
        <Text>Merek : {this.state.merek}</Text>
        <Text>Type : {this.state.types.type}</Text>
        <Text>Model : {this.state.types.model}</Text>
      </View>
    )
  }
}

export default Car
```

Gambar 2. 8 Penerapan state variable pada RCC

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:



Gambar 2. 9 Output penggunaan state variable pada RCC

### C. 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're going to south west now, come on.</Text>;
}
```

Gambar 2. 10 Penerapan state function pada RFC

Pada script di atas 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*.



Gambar 2. 11 Output penerapan state function pada RFC

Biasanya dalam sebuah fungsi selalu ada sebuah aksi berupa pengiriman sebuah data dalam bentuk variabel. 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>
  );
};

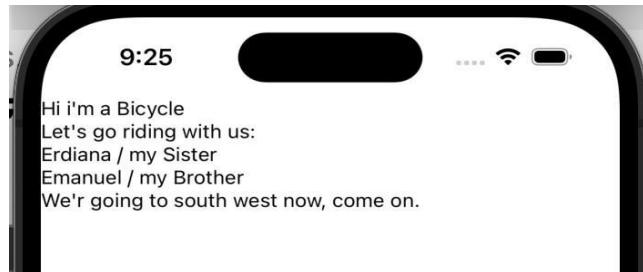
function Place2Go(value) {
  return <Text>We're going to {value} now, come on.</Text>;
}

```

Gambar 2. 12 Penerapan passing parameter pada RFC

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

Sedangkan pada fungsi Place2Go(), cukup mengirimkan nya ke dalam 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. 13 Output penerapan state function untuk passing parameters

#### D. State Function pada RCC

Berbeda halnya dengan RFC, penerapan sebuah fungsi di dalam 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;

```

Gambar 2. 14 Penerapan fungsi pada RCC

Dari script diatas untuk membuat sebuah fungsi di RCC, pertama kali yang perlu diperhatikan adalah menginisialisasi nama sebuah fungsi di dalam `constructor()`. Dengan menuliskan script seperti berikut `this.Come2Go = this.Come2Go.bind(this)`. Didalam RCC untuk membuat sebuah function harus berada dibawah constructor dan bentuknya pun 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()`.

### 2.2.3 Core Component UI dan JSX

#### A. 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>
  );
}
```

Gambar 2. 15 Penggunaan container 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>
  );
}

```

Gambar 2. 16 Penggunaan container SafeAreaView

#### c. ImageBackground

Sama halnya dengan container SafeAreaView, container ini hanya dapat digunakan sekali saja dalam inisialisasinya. Container ini diperlukan 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>
  );
}

```

Gambar 2. 17 Penggunaan container image background

#### 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>
);

```

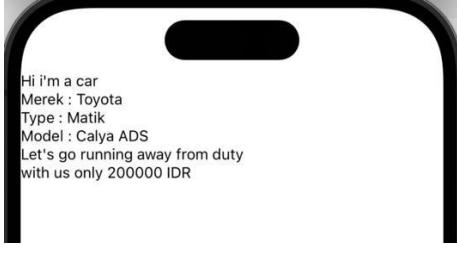
Gambar 2. 18 Penggunaan container ScrollView

## B. Basic UI

Basic UI pada React Native mencakup elemen-elemen dasar yang digunakan untuk membangun tampilan antarmuka aplikasi.

### 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. Status Bar terdapat pada library expo, yaitu `import { StatusBar } from "expo-status-bar"`.

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

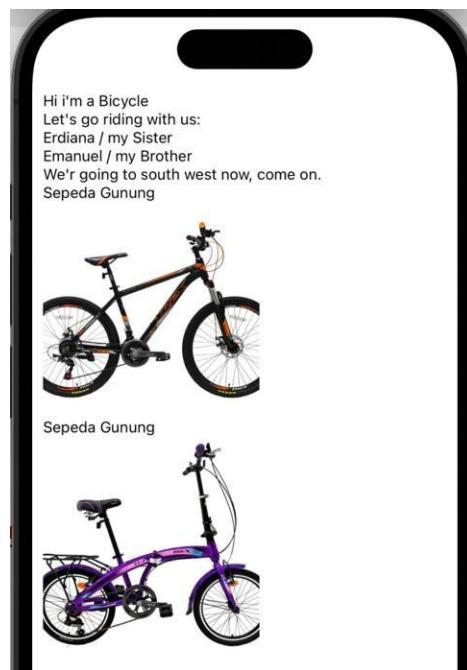
Berikutnya ialah JSX UI untuk menampilkan berbagai jenis gambar, termasuk gambar jaringan, sumber daya statis, gambar lokal sementara, dan gambar dari disk lokal, seperti rol kamera.

```
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>
);
```

Gambar 2. 19 Penggunaan komponen UI Image

Berikut adalah contoh output dari penggunaan komponen ui Image:



Gambar 2. 20 Output penggunaan komponen UI Image

### C. Forms UI

Forms UI pada React Native mencakup komponen-komponen yang digunakan untuk mengatur input data dari pengguna.

#### a. TextInput

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

Code	Output
------	--------

```

<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>

```

The screenshot shows a mobile application interface on an iPhone X. At the top, the status bar displays the time as 10:41. Below it is a circular profile picture of a person with dark hair and a beard. The main content area contains three input fields. The first field is labeled "Student ID:" and contains the value "212344545". The second field is labeled "Fullname:" and contains the value "Diaenudin Bin Avon". The third field is labeled "Address:" and contains the value "Jl. Perkutut No 40 A Block X 1 Jakarta".

## b. Buttons

Komponen tombol dasar yang seharusnya di render 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>
  );
};
```

- TouchableOpacity

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

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

- TouchableHighlight

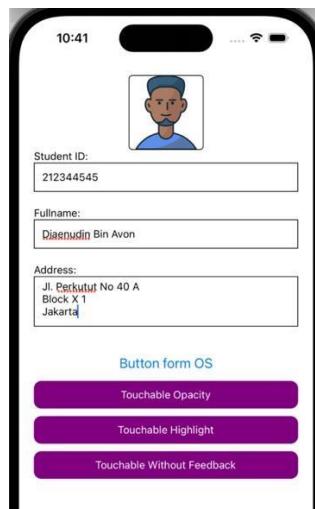
Pembungkus untuk membuat tampilan merespons sentuhan dengan benar. Saat ditekan, opasitas tampilan yang dibungkus akan berkurang, yang memungkinkan warna lapisan bawah terlihat, menggelapkan, atau mewarnai tampilan.

```
<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>
```



Gambar 2. 21 Output komponen UI Forms

## D. Stylesheet

StyleSheet adalah abstraksi yang mirip dengan CSS StyleSheets. React Native menyediakan sejumlah komponen dasar yang dapat digunakan secara langsung tetapi menurut tema aplikasi, kita harus menyesuaikan komponen kadang-kadang dan itulah mengapa kita menggunakan stylesheet. Intinya Stylesheet adalah sebuah abstraksi yang mirip dengan stylesheet css. Pada CSS code **backgroundcolor** digunakan untuk merubah *background color* sedangkan pada React Native menggunakan kode **backgroundColor**.

Terdapat 3 cara untuk membuat style pada tampilan aplikasi, antara lain :

- Inline

Kita bisa membuat style secara langsung pada component yang ingin kita beri style, untuk membuat style secara inline kita menggunakan props yang sudah disediakan . Akan tetapi

cara ini tidak disarankan , karena akan mengotori kode . Dengan metode Inline kita tidak perlu import component StyleSheet dari react native.

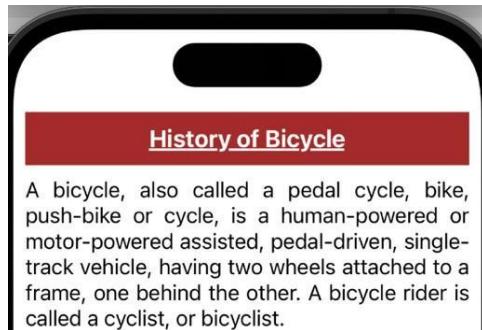
```
import { View, Text, StyleSheet } from "react-native";
import React from "react";

const ItemTyphograph = () => {
  return (
    <View>
      <Text
        style={{
          fontSize: 20,
          fontWeight: "bold",
          backgroundColor:"brown",
          color:"#fff",
          textAlign:"center",
          textDecorationLine: "underline",
          padding:10,
          marginBottom:10
        }}
      >
        History of Bicycle
      </Text>
      <Text style={styles.paragraph}>
        A bicycle, also called a pedal cycle, bike, push-bike or cycle, is a human-powered or motor-powered assisted, pedal-driven, single-track vehicle, having two wheels attached to a frame, one behind the other. A bicycle rider is called a cyclist, or bicyclist.
      </Text>
    </View>
  );
};

export { ItemTyphograph };
```

Gambar 2. 22 Penerapan stylesheet inline

Untuk membuat stylesheet secara inline pada script diatas kita dapat memasang attribute style pada tag element JSX. Seperti pada `<Text style={{key:value}}>...</Text>`. Property lengkap dari stylesheet react native dapat dilihat di <https://reactnative.dev/docs/text#style>. Berikut adalah hasil output dari script diatas:



Gambar 2. 23 Output penerapan stylesheet inline

- Embed

Kita bisa juga membuat style tanpa mengotori kode salah satunya dengan cara embed. Embed adalah cara membuat style di dalam satu file , tetapi tidak secara langsung di dalam component. Kita bisa menggunakan built in component yang sudah disediakan oleh React Native yaitu dengan StyleSheet. Pada penerapan kali ini kita membutuhkan library dari react-native untuk menggunakan StyleSheet seperti `import { StyleSheet } from "react-native"`. Masih pada file yang sama yaitu `ItemTyphograph()`. Pada JSX Text kedua yang berupa paragraph, disini akan ditambahkan sebuah style embedded.

```

const ItemTyphograph = () => {
  return (
    <View>
      <Text>...
      </Text>

      <Text style={styles.paragraph}>
        A bicycle, also called a
        <Text style={{...styles.paragraph, color:"red"}}>pedal cycle</Text>,
        <Text style={{...styles.paragraph, fontWeight:"bold"}>bike</Text>,
        <Text style={{...styles.paragraph, fontStyle:"italic"}>push-bike or cycle</Text>,
        is a human-powered or motor-powered assisted, pedal-driven, single-track vehicle, h
      </Text>
    </View>
  );
};

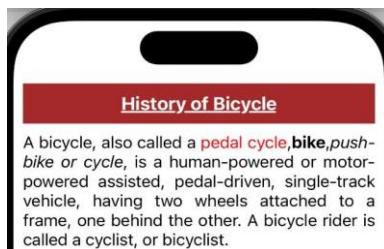
export { ItemTyphograph };

const styles = StyleSheet.create({
  paragraph :{
    fontSize: 18, textAlign:"justify"
  }
})

```

Gambar 2. 24 Penerapan stylesheet embed

Pada script diatas untuk menggunakan style embedded kita perlu mendeklarasikannya kedalam bentuk variables. Disini variable stylesheet disimpan dengan nama styles, dimana variable ini memiliki bentuk objek dari library StyleSheet. Key paragraf disini menyimpan 1 bentuk objek, yang berisi property fontSize dan textAlign. Dan key paragraf akan digunakan untuk elemen JSX Text (sesuai pada gambar kotak). Pada script diatas, tulisan pedal cycle, bike, dan push-bike or cycle, memiliki custom styling, dimana jika kita ingin mengupdate state tanpa perlu mengubah inti value pada state, kita dapat menggunakan metode *Spread Operator*, yaitu dengan cara seperti berikut: `{...object, value}` Berikut adalah contoh hasil dari output script diatas:



Gambar 2. 25 penerapan stylesheet embed

- External

Kita bisa juga membuat style tanpa mengotori kode dengan cara selain embed yaitu dengan cara external. External adalah cara membuat style tidak dalam satu file melainkan pada file terpisah. Contoh, pada source code sebelumnya variable styles dapat kita simpan dalam bentuk file js baru dan memanggil file tersebut di ItemTyphograph().

- Stylesheet untuk Layout

Berikut ini terdapat beberapa property pada desain UI:

- Flex akan menentukan bagaimana item akan "mengisi" ruang yang tersedia di sepanjang sumbu utama layar. Space akan dibagi menurut properti flex masing-masing elemen.

```

import { StatusBar } from "expo-status-bar";
import { SafeAreaView, StyleSheet, View } from "react-native";

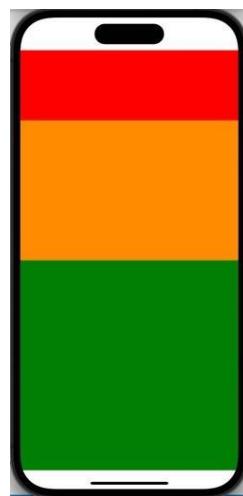
export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <StatusBar hidden={true} />
      <View style={{ flex: 1, backgroundColor: "red" }} />
      <View style={{ flex: 2, backgroundColor: "darkorange" }} />
      <View style={{ flex: 3, backgroundColor: "green" }} />
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  paragraph: {
    fontSize: 18,
    textAlign: "justify",
  },
  container: {
    flex: 1
  },
});

```

Gambar 2. 26 penerapan flex pada stylesheet

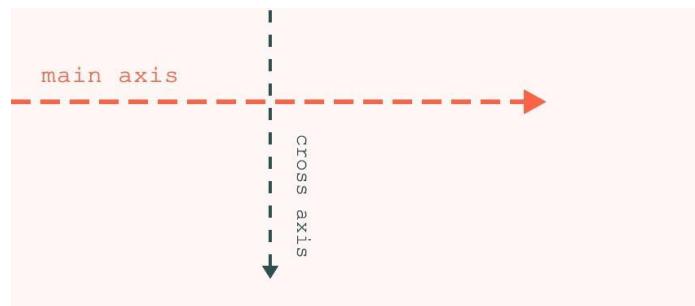
Dalam contoh berikut, tampilan merah, oranye, dan hijau adalah semua anak dalam tampilan kontainer yang memiliki set flex: 1. Tampilan merah menggunakan flex: 1 , tampilan oranye menggunakan flex: 2, dan tampilan hijau menggunakan flex: 3 .  $1+2+3 = 6$ , artinya tampilan merah akan mendapatkan  $1/6$  ruang, oranye  $2/6$  ruang, dan hijau  $3/6$  ruang.



Gambar 2. 27Output penerapan Flex pada stylesheet

## Flex Direction

FlexDirection mengatur arah Main Axis. Dimana nilai default dari FlexDirection ialah column, yang artinya Cross Axis akan berada di sumbu vertikal, bergerak dari atas ke bawah.



Gambar 2. 28 Flex Direction

**column** - (nilai default) mengatur posisi node-node berada dari atas ke bawah. Jika pembungkus diaktifkan, baris berikutnya akan dimulai di sebelah kanan item pertama di bagian atas area.

**row** – mengatur posisi node-node dari kiri ke kanan. Jika pembungkus diaktifkan, baris berikutnya akan dimulai di bawah item pertama di sebelah kiri wadah.

**column-reverse** - mengatur posisi node-node dari bawah ke atas. Jika pembungkus diaktifkan, baris berikutnya akan dimulai di sebelah kanan item pertama di bagian bawah wadah.

**row-reverse** - mengatur posisi node-node dari kanan ke kiri. Jika pembungkus diaktifkan, baris berikutnya akan dimulai di bawah item pertama di sebelah kanan wadah.

Berikut adalah contoh script awal tanpa menggunakan flexdirection pada stylesheet:

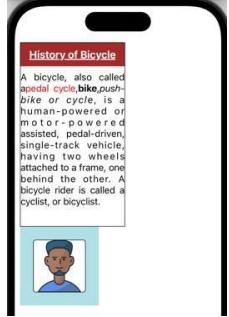
```

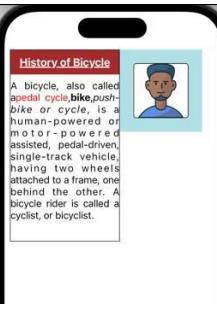
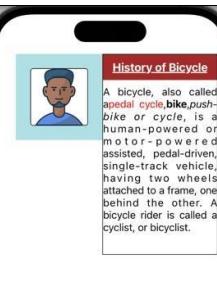
import { StatusBar } from "expo-status-bar";
import { SafeAreaView, StyleSheet, View } from "react-native";
import { ItemImage, ItemTyphograph } from "./components/stylesheets/Items";

export default function App() {
  return (
    <SafeAreaView style={{ flex: 1 }}>
      <StatusBar hidden={true} />
      <View style={styles.container}>
        <ItemTyphograph />
        <ItemImage />
      </View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  paragraph: {
    fontSize: 18,
    textAlign: "justify",
  },
  container: {
    padding: 10,
    flex: 1,
  },
});

```

Flex Direction	Output
<pre> container: [   padding: 10,   flex: 1,   flexDirection:"column" ], </pre>	

<pre>container: {   padding: 10,   flex: 1,   flexDirection:"column-reverse" },</pre>	
<pre>container: {   padding: 10,   flex: 1,   flexDirection:"row" },</pre>	
<pre>container: {   padding: 10,   flex: 1,   flexDirection:"row-reverse" },</pre>	

Dari script di atas kita akan menambahkan property flexDirection pada key container, sehingga dapat melihat bagaimana node-node didalamnya ItemTypograph dan ItemImage diatur sesuai bentuk dari flexDirection.

### Justify Content

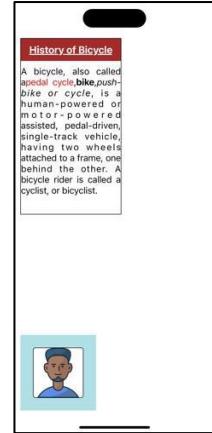
JustifyContent menjelaskan cara menyusun node-node di dalam poros utama pembungkusnya. Misalnya, menggunakan properti ini untuk memusatkan node secara horizontal di dalam pembungkus dengan flexDirection diatur ke row atau vertikal di dalam pembungkus dengan flexDirection diatur ke column.

Justify Content	Output
<pre data-bbox="262 244 817 730">container: {   padding: 10,   flex: 1,   justifyContent:"center" },</pre>	 <p>A bicycle, also called a pedal cycle, bike, push-bike or cycle, is a human-powered or motor-powered, single-track vehicle, having two wheels attached to a frame, one behind the other. A bicycle rider is called a cyclist, or bicyclist.</p> <p></p>
<pre data-bbox="262 730 817 1216">container: {   padding: 10,   flex: 1,   justifyContent:"flex-end" },</pre>	 <p>A bicycle, also called a pedal cycle, bike, push-bike or cycle, is a human-powered or motor-powered, single-track vehicle, having two wheels attached to a frame, one behind the other. A bicycle rider is called a cyclist, or bicyclist.</p> <p></p>
<pre data-bbox="262 1216 817 1733">container: {   padding: 10,   flex: 1,   justifyContent:"flex-start" },</pre>	 <p>A bicycle, also called a pedal cycle, bike, push-bike or cycle, is a human-powered or motor-powered, single-track vehicle, having two wheels attached to a frame, one behind the other. A bicycle rider is called a cyclist, or bicyclist.</p> <p></p>

```

container: {
  padding: 10,
  flex: 1,
  justifyContent:"space-between"
},

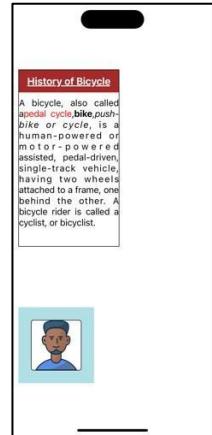
```



```

container: {
  padding: 10,
  flex: 1,
  justifyContent:"space-around"
},

```



- Align Items alignItems menjelaskan cara menyelaraskan node-node di sepanjang Cross Axis pembungkusnya. Ini sangat mirip dengan justifyContent tetapi alih-alih diterapkan ke Main Axis, alignItems diterapkan ke Cross Axis.

Align Items	Output
<pre> container: {   padding: 10,   flex: 1,   alignItems:"baseline" }, </pre>	<p>A screenshot of a mobile application interface. It shows a white card with the "History of Bicycle" header and a detailed text description. The text is aligned at the baseline, meaning the bottom edge of the text block is aligned with the bottom edge of the card's content area. At the bottom of the card is a small blue square containing a black and white profile picture of a person.</p>

```
container: {  
  padding: 10,  
  flex: 1,  
  alignItems:"center"  
},
```



```
container: {  
  padding: 10,  
  flex: 1,  
  alignItems:"flex-end"  
},
```

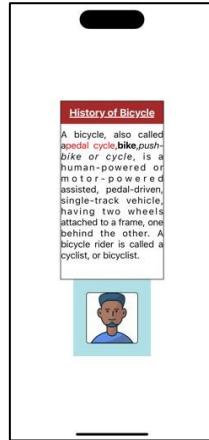


```
container: {  
  padding: 10,  
  flex: 1,  
  alignItems:"flex-start"  
},
```



## 2.3 Latihan Pembelajaran

1. Buatlah script untuk memposisikan node-node berada di tengah-tengah layar monitor seperti berikut:



2. Dalam membuat script nomor 1, buatlah sebuah RCC sebagai titik awal aplikasi dan untuk node node nya menggunakan RFC (kotak tulisan history bicycle & kotak gambar). Panggil kedua RFC tersebut sebagai bagian dari node RCC yang akan anda buat.
3. Buatlah sebuah splash screen sederhana seperti gambar dibawah ini dan buatlah dengan menggunakan StyleSheet Inline dan Embedded.



4. Buatlah sebuah halaman untuk menampilkan informasi data diri anda, namun data mengenai informasi anda disimpan dalam bentuk sebuah Objek sebagai berikut:

```
const MyBio = {  
    identity:{  
        npm:212310056 ,  
        firstname: "MUHAMMAD",  
        middlename:"ZIDAN",  
        lastname:"AKBAR",  
    },  
    educations:[  
        {id:1, school:"SDN 1 Kota Bogor"},  
        {id:2, school:"SMPN 1 Kota Bogor"},  
        {id:3, school:"SMAN 1 Kota Bogor"},  
    ],  
    mobile_phone: 0812345679,  
  
    email:"212310056@student.ibik.ac.id",  
    gender:'M',    tall_body:175,  
    weight_body:64.5  
}
```

Implementasikanlah variable diatas dengan menggunakan state variable RCC dan state variable RFC. Dan berikan output dari masing-masing component.

## BAB 3 LISTING DAN LIFE CYCLE MODE

### 3.1 Tujuan Pembelajaran

Tujuan pembelajaran pada bab ini adalah memberikan pemahaman tentang cara mengelola daftar elemen dalam aplikasi React Native dan memahami siklus hidup (life cycle) dari komponen React Native. Diharapkan mahasiswa dapat:

1. Memahami penggunaan Listing dalam React Native dengan fokus pada ScrollView, FlatList, dan SectionList.
2. Menguasai penggunaan ScrollView untuk membuat daftar elemen yang dapat di-scroll.
3. Menguasai penggunaan FlatList untuk mengatur daftar elemen dengan jumlah besar atau dinamis secara efisien.
4. Memahami penggunaan life cycle mode pada komponen React Native, seperti componentDidMount, componentDidUpdate, dan componentWillUnmount.

### 3.2 Dasar Teori

#### 3.2.1 Install Library

Pada materi kali ini memerlukan beberapa dependencies yang perlu digunakan, yaitu salah satunya ialah penggunaan icon pada aplikasi react native expo. Pada project kali ini akan memanfaatkan dependencies icon dari <https://oblador.github.io/react-native-vector-icons/> oleh karenanya bukalah terminal pada project anda dan masukan syntax dibawah ini untuk menginstall library icon:

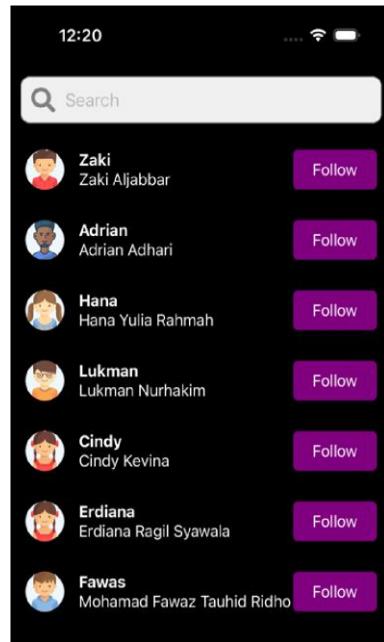
```
npm i react-native-vector-icons
```

Setelah berhasil menambahkan library icon tersebut silahkan anda jalankan expo metro servernya dan ikuti tahapan-tahapan perintah dibawah ini dengan baik.

#### 3.2.2 StyleSheet – Daftar Teman UI

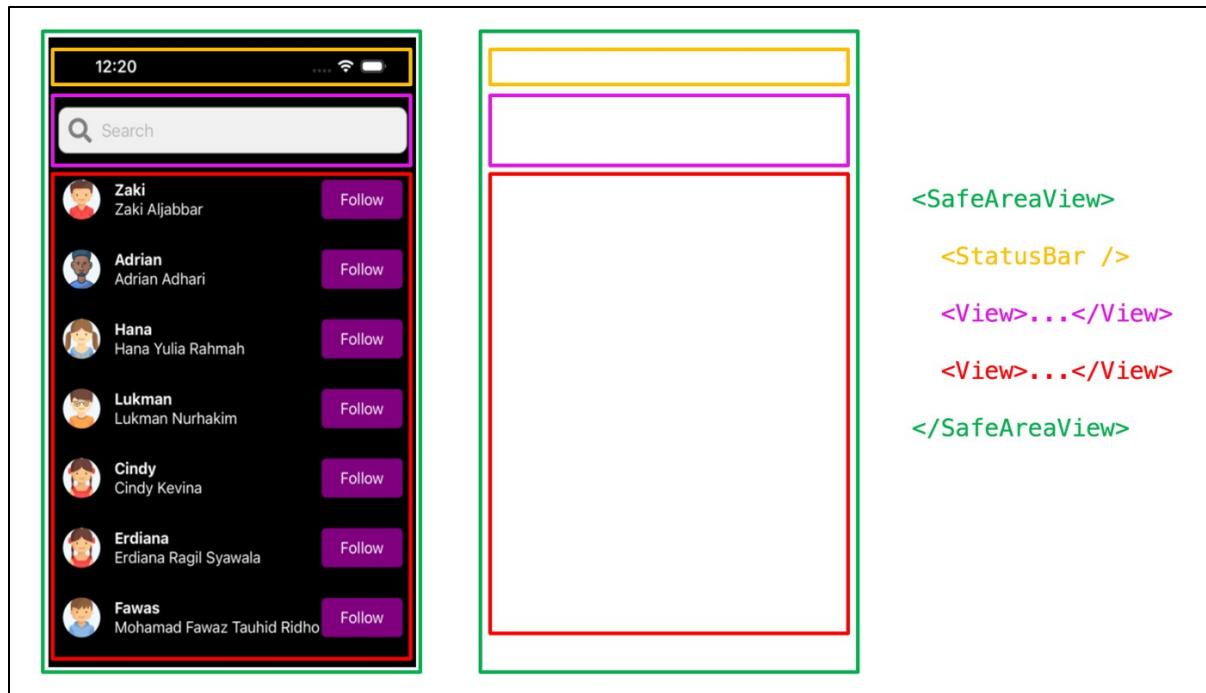
Pada bagian ini akan membahas mengenai penggunaan Listing dan Life Cycle method pada react native expo cli. Namun sebelum ke pembahasan materi, hal yang perlu dilakukan

adalah membuat layouting pada aplikasi. Contoh kasus kali ini adalah membuat sebuah halaman daftar pertemanan. Jika merujuk pada desain dibawah ini:



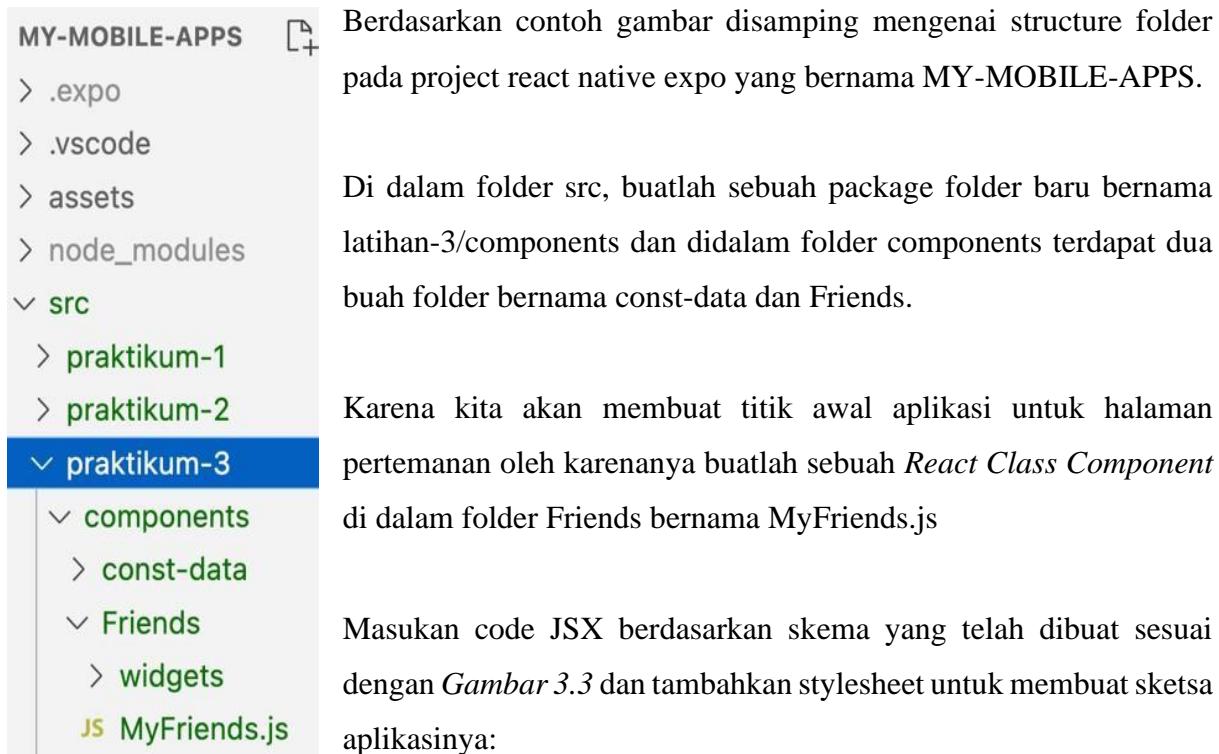
Gambar 3. 1 Desain halaman daftar pertemanan

Berdasarkan desain diatas dapat kita break down UI untuk mendapatkan skema konsep element pada react native seperti berikut:



Gambar 3. 2 Break Down UI halaman daftar teman

Setelah melakukan Analisa terhadap desain aplikasi, selanjutnya kita tentukan sebuah komponen react dalam bentuk class untuk menjadi titik awal aplikasi. Pada project react native expo cli buatlah sebuah package baru bernama latihan -3 di dalam folder `./src/`.



*Gambar 3. 3 Struktur folder pada project react native*

```

import { SafeAreaView, StyleSheet, View } from 'react-native'
import React, { Component } from 'react'
import { StatusBar } from 'expo-status-bar';

export class MyFriends extends Component {
  render() {
    return (
      <SafeAreaView style={styles.container}>
        <StatusBar hidden={false} style="light" />
        <View style={styles.header}></View>
        <View style={styles.body}></View>
      </SafeAreaView>
    )
  }
}

export default MyFriends

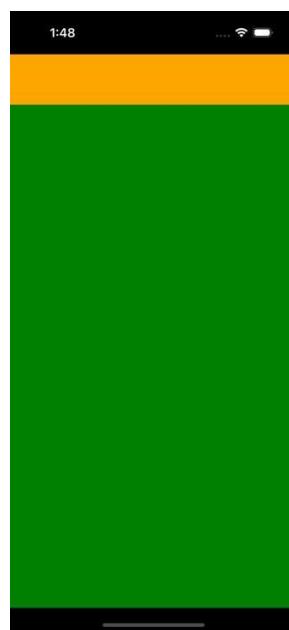
const styles = StyleSheet.create({
  container: { flex: 1, backgroundColor: "black" },
  header:{ flex:1, justifyContent:"center", paddingHorizontal:10, backgroundColor: "orange" },
  body:{ flex: 10, backgroundColor: "green" }
});

```

*Gambar 3. 4 Sketsa Scripting MyFriend.js dengan class*

Pada sketsa scripting diatas, dimana titik awal aplikasi untuk halaman pertemanan ini dibuat dengan komponen class. Komponen ini menggunakan container Safe Area View sebagai pembungkus pertama, didalamnya terdapat 3 buah elemen JSX yaitu ada Status Bar dan container View per tema untuk bagian header dan View kedua diperuntukan bagian body.

Disetiap element emmet JSX memiliki property style yang telah didefinisikan sesuai dengan nama-nama key pada variable styles, yaitu ada key container, header dan body. Dari hasil scripting tersebut maka output yang ditampilkan akan seperti berikut ini:



Gambar 3. 5 Output skema awal

Di Dalam folder Friends, buatlah sebuah folder bernama widgets. Dimana folder ini berisi file-file mentah komponen function react. Widget pertama yang akan dibuat adalah bagian header pada desain halaman pertemanan. Simpanlah *React Function Component* dengan nama Search Bar.js di dalam folder widgets milik Friends.

## SearchBar.js

```
import { View, StyleSheet, TextInput } from "react-native";
import React from "react";
import FontAwesome5 from "react-native-vector-icons/FontAwesome5";

const SearchBar = () => {
  return (
    <View style={styles.search_box}>
      <FontAwesome5 name={"search"} size={25} color="grey" />
      <TextInput style={styles.search_input} placeholder="Search" />
    </View>
  );
}

export default SearchBar;

const styles = StyleSheet.create({
  search_box: {
    padding: 10, flexDirection: "row", borderWidth: 1,
    borderColor: "grey", borderRadius: 10, backgroundColor: "#f0f0f0",
  },
  search_input: { fontSize: 18, width: "90%", color: "white", marginLeft: 10 },
});
```

Gambar 3. 6 Script Search Bar dengan icon dan font icon

Pada script diatas terdapat syntax untuk menggunakan icon dari library yang telah diinstall yaitu reactnative-vector-icons. Contoh script diatas untuk penggunaan font icon menggunakan type fontawesome 5 dimana anda dapat mencari nama-nama iconnya di <https://fontawesome.com/v5/search>

```
<FontAwesome5 name={"search"} size={25} color="grey" />
```

Disini pada emmet `FontAwesome5` property `name` menggunakan value `search` untuk mendapatkan bentuk icon seperti kaca pembesar. Lalu properties `size` diperuntukan untuk mengatur ukuran iconnya, sedangkan property `color` digunakan untuk memberikan warna pada icon.

Setelah menambahkan script diatas silakan anda panggil RFC SearchBar didalam RCC MyFriends sesuai dengan posisi penempatan layoutnya.

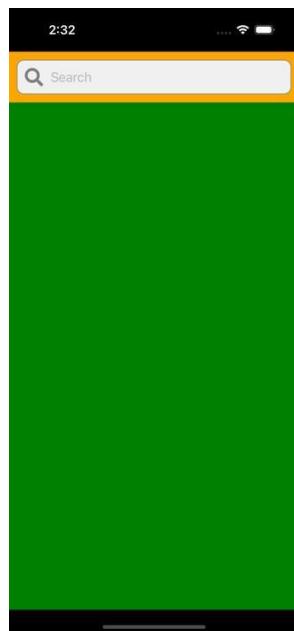
### MyFriends.js

```
export class MyFriends extends Component {
  render() {
    return (
      <SafeAreaView style={styles.container}>
        <StatusBar hidden={false} style="light" />
        <View style={styles.header}>
          <SearchBar />
        </View>
        <View style={styles.body}></View>
      </SafeAreaView>
    )
  }
}
```

Memanggil fungsi SearchBar dan menempatkannya pada layer-Res

Gambar 3. 7 memanggil fungsi searchBar pada MyFriend.js

Maka output dari menampilkan script di atas sebagai berikut:



Gambar 3. 8 Output widget SearchBar

Untuk layer kedua pada script MyFriends.js, View dengan key body akan berisi contoh penggunaan Listing seperti *ScrollView*, *FlatList* dan *SectionList*. Materi tersebut akan dijelaskan pada bagian selanjutnya.

### 3.2.3 Listing

Penggunaan konsep listing pada react native terbagi menjadi tiga buah tipe, pertama secara manual mengandalkan emmed *ScrollView*, kedua ialah *FlatList* dan terakhir adalah *SectionList*. Pada contoh kali ini penggunaan listing selalu memanfaatkan data dalam bentuk array object. Berikut ini adalah contoh data buatan yang akan digunakan untuk mengimplementasi ketiga bentuk listing:

```
const Users = [  
  { id: 1, name: "Zaki", fullname: "Zaki Aljabbar" gender: "M" },  
  { id: 2, name: "Adrian", fullname: "Adrian Adhari", gender: "M" },  
  { id: 3, name: "Hana", fullname: "Hana Yulia Rahmah", gender: "F" },  
  { id: 4, name: "Lukman", fullname: "Lukman Nurhakim", gender: "M" }, {  
    id: 5, name: "Cindy", fullname: "Cindy Kevina", gender: "F" }  
];
```

Sedangkan untuk rendering item listing berdasarkan objek diatas, pada contoh bagian ini akan membuat sebuah komponen baru bernama *UserItem*. Fungsi ini yang nantinya digunakan untuk merender JSX dengan ketiga tipe listing tersebut.

Sebelumnya, silakan anda siapkan folder assets untuk menyimpan file-file media seperti gambar. Folder assets ini sejajar dengan folder src pada project anda. Pada script dibawah ini pada emmet JSX Image memanggil sebuah file bernama icon-girl-1.png dimana file ini berada di folder assets/icons/.

## UserItems.js

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

const UserItem = ({ item }) => {
  return (
    <View style={styles.search_container}>
      <View style={styles.search_account}>
        <Image source={require("../../../../../assets/icons/icon-girl-1.png")} style={styles.story_ava} />
        <View>
          <Text style={{ ...styles.story_name, fontWeight: "bold" }}>
            {item.name}
          </Text>
          <Text style={styles.story_name}>{item.fullname}</Text>
        </View>
      </View>
      <View>
        <TouchableOpacity activeOpacity={0.6} style={styles.btn_follow}>
          <Text style={styles.story_name}>Follow</Text>
        </TouchableOpacity>
      </View>
    </View>
  );
};

const styles = StyleSheet.create({
  search_container: {
    flexDirection: "row", justifyContent: "space-between",
    alignItems: "center", padding: 15,
  },
  search_account: {
    flexDirection: "row", justifyContent: "flex-start", alignItems: "center",
  },
  story_ava: {
    width: 40, height: 40, borderRadius: 100,
    backgroundColor: "#f0f8ff", marginRight: 15,
  },
  story_name: { fontSize: 16, color: "white", textAlign: "left" },
  btn_follow: {
    backgroundColor: "purple", paddingHorizontal: 20,
    paddingVertical: 10, borderRadius: 5,
  },
});

export default UserItem;
```

Gambar 3. 9 script UserItem.js

## A. ScrollView

ScrollView merupakan elemen scroll secara umum yang dapat berisi banyak komponen dan tampilan. Item yang dapat digulir bisa berbeda-beda, dan dapat menggeser baik secara *vertikal* maupun *horizontal* (dengan memasang properti horizontal).

Script	Output
<pre> import { ScrollView } from 'react-native' import React from 'react' import UserItem from './UserItem'  const ExpScrollView = ({Users}) =&gt; {   return (     &lt;ScrollView&gt;       {Users.map((v,index)=&gt;(         &lt;UserItem item={v} key={index} /&gt;       ))}     &lt;/ScrollView&gt;   ) }  export default ExpScrollView </pre> <pre> import { Users } from '../const-data/ObjDummies'; import ExpScrollView from './widgets/ExpScrollView';  export class MyFriends extends Component {   render() {     return (       &lt;SafeAreaView style={styles.container}&gt;         &lt;StatusBar hidden={false} style="light" /&gt;         &lt;View style={styles.header}&gt;           &lt;SearchBar /&gt;         &lt;/View&gt;         &lt;View style={styles.body}&gt;           &lt;ExpScrollView Users={Users} /&gt;         &lt;/View&gt;       &lt;/SafeAreaView&gt;     )   } } </pre>	

Gambar 3. 10 Script dan Output Penggunaan ScrollView

## B. FlatList

Penggunaan FlatList tidak jauh berbeda dengan ScrollView, pada emmed ini juga memiliki sebuah properti yang dapat digunakan untuk membentuk scroll dalam orientasi Horizontal maupun Vertikal. Cukup menambahkan property bernama *horizontal={true/false}*.

Namun yang berbeda ialah adanya property renderItem, property tersebut diperuntukkan untuk mengirim sebuah object ke dalam bentuk render JSX tanpa menggunakan looping array object datanya. Berikut adalah contoh script dari penerapan emmed FlatList:

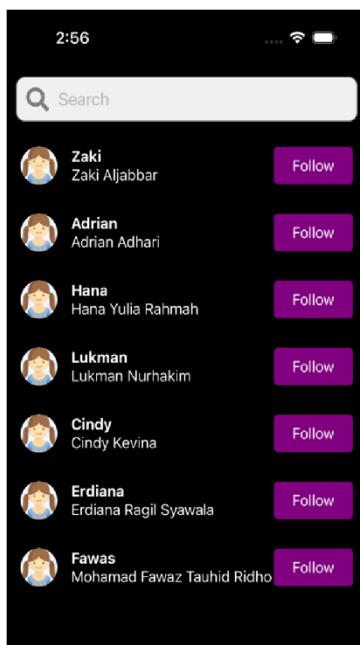
```

import { FlatList } from 'react-native'
import React from 'react'
import UserItem from './UserItem'

const ExpFlatList = ({Users}) => {
  return (
    <FlatList data={Users} renderItem={({item}) => <UserItem item={item} />} />
  )
}

export default ExpFlatList

```



Gambar 3. 11 Script dan Output Penggunaan FlatList

### C. SectionList

Berbeda dengan *ScrollView* atau *FlatList*, emmed ini tidak bisa memiliki orientasi dalam bentuk Horizontal. Dan pengiriman data valuenya pun memiliki konsep yang berbeda dari kedua contoh listing sebelumnya.

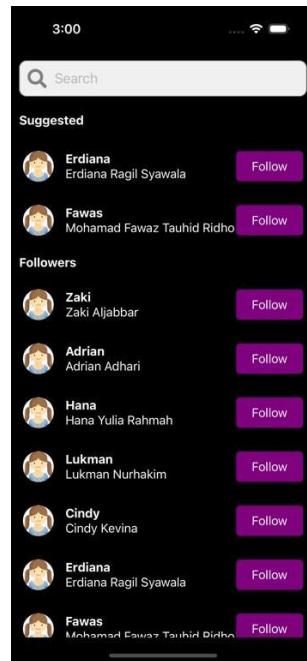
```

import { Text, SectionList } from "react-native";
import React from "react";
import UserItem from "./UserItem";

const ExpSectionList = ({Users}) => {
  const data = [{ title: "Suggested", data: Users },
    { title: "Followers", data: Users }];
  return (
    <SectionList sections={data}
      renderItem={({item}) => <UserItem item={item} /> }
      renderSectionHeader={({section: {title}}) => (
        <Text style={{ color:"white"}}>{title}</Text>
      )} />
    );
};

export default ExpSectionList;

```



Gambar 3. 12 Script dan Output Penerapan SectionList

Jika diperhatikan implementasi terhadap ketiga tipe listing, dari sisi script memiliki perbedaan dari cara merender atau menerima sebuah data hingga ditampilkan. Namun dari sisi output, ketiga list tersebut memiliki kesamaan, yang berbeda hanyalah listing *SectionList*, dimana emmet ini memerlukan *renderSectionHeader* untuk menampilkan judul dari *object* datanya.

### **3.3 Latihan Pembelajaran**

1. Pada *react function component* UserItem, buatlah sebuah kondisi jika *gender* dari setiap objeknya adalah “M” maka menampilkan gambar *icon boy*, namun sebaliknya menampilkan gambar *icon girl*.
2. Buatlah package baru dengan nama Latihan-3 pada project latihan react expo cli anda.
3. Cloning lah aplikasi dibawah ini dengan baik, gunakanlah Listing ScrollView, FlatList, dan SectionList.

## BAB 4 NAVIGATION BAR

### 4.1 Tujuan Pembelajaran

Tujuan pembelajaran pada bab ini adalah memberikan pemahaman tentang konsep dasar navigasi dalam pengembangan aplikasi React Native. Diharapkan mahasiswa dapat:

1. Memahami konsep dasar dan pentingnya menggunakan navigasi dalam pengembangan aplikasi React Native untuk mengatur tampilan dan alur aplikasi secara efisien.
2. Menguasai cara menginstall library tambahan yang diperlukan untuk implementasi navigasi di aplikasi React Native.
3. Memahami penggunaan Stack Navigation untuk mengatur navigasi antar halaman (screen) dalam aplikasi dengan efisien.
4. Menguasai penggunaan Bottom Tabs Navigation untuk membuat navigasi berbasis tab di bagian bawah layar aplikasi.

### 4.2 Dasar Teori

#### 4.2.1 Install Library

Pada materi kali ini membutuhkan beberapa library yang perlu diinstall untuk melakukan proses perpindahan halaman dari satu komponen ke komponen lain. Library dibawah ini merupakan library dasar untuk melakukan proses pindah komponen, silakan anda tambahkan library tersebut kedalam project react expo cli anda:

```
npm install @react-navigation/native @react-navigation/native-stack  
npx expo install react-native-gesture-handler  
npx expo install react-native-screens react-native-safe-area-context
```

Setelah berhasil menambahkan library tersebut silakan anda jalankan Expo Server anda dengan benar.

#### 4.2.2 Stack

Stack navigasi menyediakan cara bagi aplikasi anda untuk melakukan transisi antar layar, setiap komponen yang didaftarkan oleh Stack sifatnya akan bertumpuk layar. Jika memanggil komponen satu ke komponen lain maka komponen berikutnya akan berada di atas layar komponen sebelumnya. Berikut ini adalah contoh penggunaan STACK dengan React Function Component.

Bukalah file App.js anda lalu tambahkan script diatas untuk mendaftarkan komponen-komponen yang menggunakan STACK:

##### App.js

```
import { NavigationContainer } from "@react-navigation/native";
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import { Text, TouchableOpacity, View } from "react-native";

export default function App() {

  const Stack = createNativeStackNavigator();

  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name="Page1"
          component={Page1}
          options={{ title: "Welcome" }}
        />
        <Stack.Screen name="Page2" component={Page2} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Gambar 4. 1 Penggunaan Stack pada App.js

Pada file App.js diatas pertama anda harus memanggil file library `@react-navigation/native` dan `@react-navigation/native-stack` . Buatlah sebuah variable bernama Stack dimana variable ini memiliki value `createNativeStackNavigator()`. Variable inilah yang nantinya digunakan untuk mendaftarkan setiap Komponen React yang telah dibuat.

**Note:**

`<NavigationContainer>` à pembungkus untuk mendefinisikan sebuah tumpukan layer navigasi yang berasal berbagai komponen react.

`<Stack.Navigator>` à pembungkus untuk menginisialisasi komponen-komponen layar apa saja yang dapat digunakan. Ini sama halnya dengan mendefinisikan sebuah root end-point pada sebuah website. `<Stack.Screen>` à inisialisasi komponen react (RFC/RCC) sebagai bagian dari layar yang akan dibuat tumpukan antar muka.

Pada `Stack.Screen` memiliki beberapa properties umum yang dapat digunakan, yaitu:

Properties	Value
name	Diperlukan untuk penanaman root pada layar
component	Memanggil file komponen react (RCC/RFC)
options	Berisi konfigurasi untuk title layer atau untuk aksi gesture dan masih banyak lagi. <code>{title:"", gestureEnabled :boolean}</code>

Berikut ini adalah contoh react function component untuk Page1 dan Page2 untuk menerapkan proses transisi antar layar:



```

const Page1 = ({navigation}) => {
  return (
    <TouchableOpacity onPress={() => navigation.navigate('Page2', {name: 'Febry'})} >
      <View style={{ backgroundColor:"purple", padding:10, borderRadius:10, margin:10 }}>
        <Text style={{ color:"white", textAlign:"center" }}>Click here to see me</Text>
      </View>
    </TouchableOpacity>
  );
}

```

Parameter `navigation` ditambahkan jika anda ingin melakukan transisi antar layar

Gambar 4. 2 Function component untuk Page 1 & 2

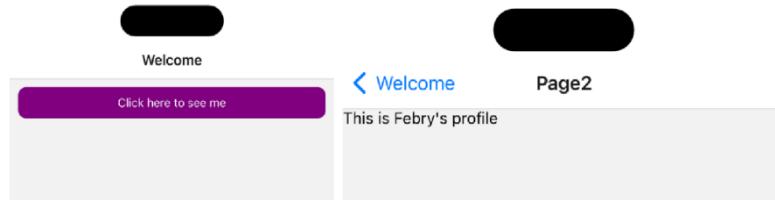
Pada RFC Page1 terdapat sebuah JSX untuk membuat sebuah tombol, jika tombol tersebut ditekan (onPress) maka akan mengeksekusi layar Stack.Screen bernama Page2. Jika anda ingin mengirimkan sebuah parameter ke dalam navigasi anda dapat menginisialisasinya dengan format sebuah object:

```
navigation.navigate('name_stack', {object_passing_parameter});
```

Sedangkan pada RFC Page2 dibawah ini hanya menampilkan JSX dalam bentuk texting dan mencoba untuk mengambil nilai parameter yang telah dikirim di Page1 dengan menggunakan parameter bernama route.

```
const Page2 = ({navigation, route}) => {
  return <Text>This is {route.params.name}'s profile</Text>;
};
```

Parameter route digunakan jika anda ingin mengambil passing parameter yang telah dikirimkan, biasanya parameter tersebut akan disimpan kedalam key bernama params dan diiringi dengan nama key yang telah dikirim.



Gambar 4. 3 Contoh transisi antar layar

#### 4.2.3 Bottom Tabs Navigations

Jika anda ingin menggunakan tab bottom navigasi anda perlu menginstall library dengan:

```
npm install @react-navigation/bottom-tabs
```

Berikut ini adalah contoh perpindahan antar layar menggunakan Bottom Tabs Navigations:

```

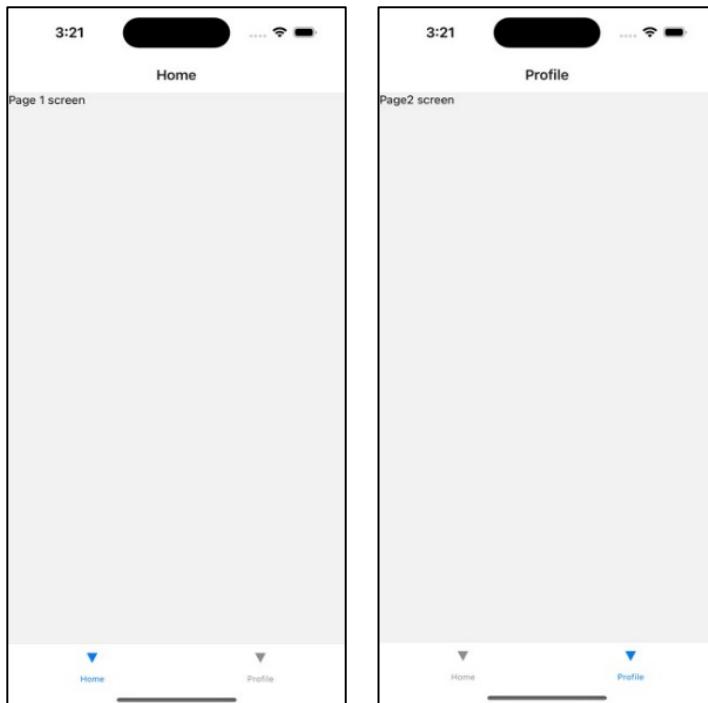
import React from "react";
import { Text, View } from "react-native";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";

export default function App() {
  const Tab = createBottomTabNavigator();
  return (
    <Tab.Navigator>
      <Tab.Screen name="Home" component={Page1} />
      <Tab.Screen name="Profile" component={Page2} />
    </Tab.Navigator>
  );
}

```

*Gambar 4. 4 Script penerapan perpindahan layar dengan Bottom Tabs Navigation*

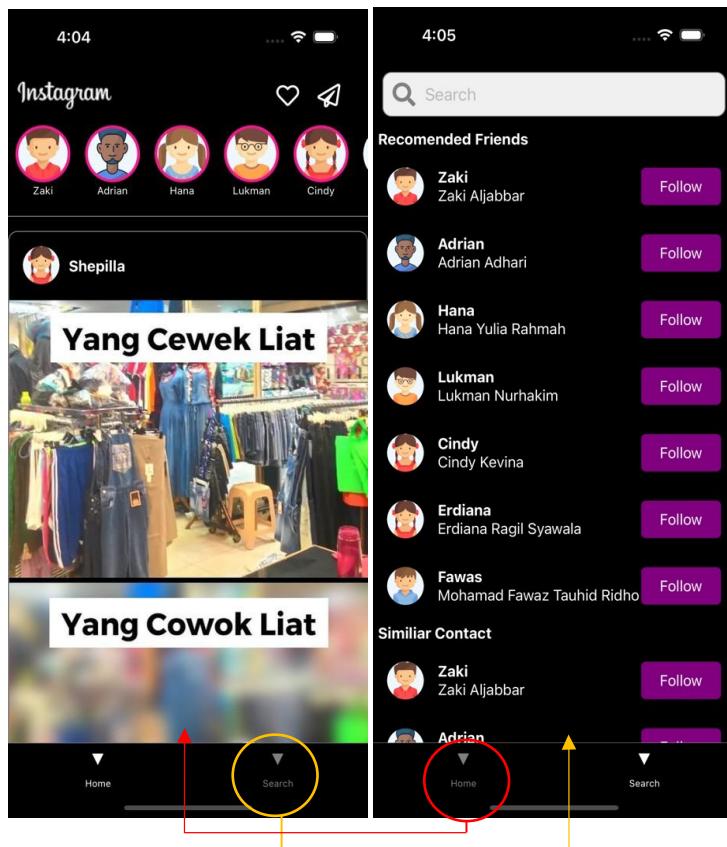
Pada penggunaan perpindahan antar layar kali ini kurang lebih memiliki skema yang sama dengan penggunaan STACK. Namun dengan TAB sebagai inisialisasi layar komponennya.



*Gambar 4. 5 Contoh transisi dengan bottom tab navigations*

### 4.3 Latihan Pembelajaran

1. Buatlah package baru bernama latihan-4.
2. Pada buku mengenai Listing, silakan anda masukan halaman-halaman komponen react yang telah dibuat seperti komponen HOME dan MyFriends dengan menggunakan Navigation Bottom Tab seperti contoh gambar dibawah ini:



3. Buatlah sebuah komponen react dimana berisi tampilan halaman SIGN IN, form isian berupa EMAIL dan PASSWORD. Jika mengklik tombol SIGN IN maka akan mengalihkan layer ke HOME seperti gambar pada soal Latihan 2.

Gunakan navigation STACK untuk komponen LOGIN agar dapat melakukan transisi halaman dimana halaman komponen tersebut bukan bagian dari Bottom Tab Navigation. Halaman form SIGN IN hanya sebatas UI tanpa adanya validasi data yang dimasukan.

## BAB 5 LIFECYCLE METHOD REACT

### 5.1 Tujuan Pembelajaran

Tujuan Pembelajaran pada bab ini adalah memberikan pemahaman tentang siklus hidup komponen (lifecycle component) dalam React serta penggunaan Hook untuk mengelola state dan operasi lainnya dalam aplikasi React. Dengan pemahaman ini, mahasiswa diharapkan dapat:

1. Memahami bagaimana siklus hidup komponen bekerja dalam React dan kapan masing-masing fase siklus terjadi.
2. Menguasai penggunaan Hook, khususnya useState, untuk mengelola state dalam komponen React dengan lebih mudah dan efisien.
3. Menggunakan lifecycle component API's untuk mengatur logika aplikasi yang memerlukan tindakan pada saat komponen dipasang (mounting) atau diperbarui (updating).
4. Memahami dan mampu melakukan operasi seperti selection, repetition, dan operasi aritmatika untuk memanipulasi data dalam aplikasi React.

### 5.2 Dasar Teori

#### 5.2.1 Lifecycle Component API's

Lifecycle component API's biasa dipakai untuk komponen react pada CLASS. Pada lifecycle class memiliki tiga buah fase utama yaitu, **Mounting**, **Updating**, dan **Unmounting**.

##### A. Mounting

Mounting merupakan Teknik untuk memasukan elemen kedalam bentuk DOM. Pada RCC memiliki 4 buah komponen yang perlu digunakan untuk melakukan Mounting:

- *constructor()*

Metode *constructor()* diinisialisasi pada awal pada saat komponen dimuat, dan ini adalah wadah untuk menyiapkan STATE awal dan beberapa nilai awal lainnya.

- *getDerivedStateFromProps()*

Metode ini biasanya dipanggil sebelum melakukan merender elemen di DOM. Metode ini baik digunakan untuk mengatur nilai value yang telah dikirimkan melalui props.

- *render()*

Metode *render()* merupakan metode yang benar-benar menampilkan HTML ke DOM atau yang dikenal dengan nama JSX.

- *componentDidMount()*

Metode *componentDidMount()* dipanggil setelah komponen dirender. Di sinilah metode ini menjalankan pernyataan yang mengharuskan komponen ditempatkan pada DOM.

Berikut ini adalah contoh mounting untuk menangkap nilai dari sebuah inputan elemen JSX dengan menggunakan TextInput:

```
import { Dimensions, Image, SafeAreaView, StyleSheet, Text, TextInput, View, } from "react-native";
import React, { Component } from "react";

export class FormRCC extends Component {
  constructor(props){
    super(props);
    this.state={
      title:"IBI Kesatuan",
      subTitle:"Bogor Indonesia"
    }
  }
  render() {
    return (
      <SafeAreaView>
        <View>
          <Image source={require("../../../../../assets/icons/icon-ibik.png")} />
          <View>
            <Text>{this.state.title}</Text>
            <Text>{this.state.subTitle}</Text>
          </View>
        </View>
      </SafeAreaView>
    )
  }
}
```

Inisialisasi key pada state lifecycle  
bernama title, dan subTitle

Cara menampilkan state  
lifecycle pada JSX

Gambar 5. 1 Penggunaan Mounting

```

<View>
    <View>
        <Text>Change Logo</Text>
    </View>
    <View>
        <Text>Title</Text>
        <TextInput placeholder="Enter title here"
            defaultValue={this.state.title}
            onChangeText={(text)=>this.setState({title:text})} />
    </View>
    <View>
        <Text>Sub Title</Text>
        <TextInput placeholder="Enter sub title here"
            defaultValue={this.state.subTitle}
            onChangeText={(text)=>this.setState({subTitle:text})} />
    </View>
</View>
</SafeAreaView>
};

}

export default FormRCC;

```

mengambil nilai value pada TextInput dengan menggunakan properties onChangeText, dan untuk mengupdate pada state gunakan this.setState({namakey:value})

Gambar 5. 2 Penggunaan state lifecycle

Dari script diatas maka output nya akan sebagai berikut:



Gambar 5. 3 Tampilan output state lifecycle

Silakan anda ganti value pada isian form diatas untuk mengganti judul dan sub judul pada header aplikasi.

### **5.2.2 HOOK**

Sedangkan untuk lifecycle pada React Function Component (RFC) menggunakan HOOK. Konsep dari HOOK ini jauh lebih simple dari React Class Component. Dengan menggunakan HOOK, state pada komponen dapat diakses dengan fitur react lainnya.

Dalam penggunaan HOOK ada tiga hal yang perlu diperhatikan, yaitu:

- a. Hook hanya dapat dipanggil di dalam komponen RFC.
- b. Hook hanya dapat dipanggil pada level teratas dari sebuah komponen.
- c. Hook tidak bisa berisi logika bersyarat

#### A. useState

React *useState* Hook memungkinkan untuk mengelola data pada state didalam komponen fungsi. State pada umumnya mengacu ke data atau properti yang perlu ditracking dalam aplikasi. Untuk menggunakan useState anda dapat menggunakan library dari react dengan cara mengimport `import { useState } from "react";` Berikut ini adalah contoh dari penggunaan HOOK namun dengan contoh kasus yang sama yaitu mengubah header aplikasi untuk title dan subtitle:

```

import { View, Text, StyleSheet, SafeAreaView, Image, TextInput } from "react-native";
import React, { useState } from "react";

const FormRFC = () => {
  const [title, setTitle] = useState("IBI Kesatuan");
  const [subTitle, setSubTitle] = useState("Bogor Indonesia");
  return (
    <SafeAreaView>
      <View>
        <Image source={require("../../../../../assets/icons/icon-ibik.png")} />
        <View>
          <Text>{title}</Text>
          <Text>{subTitle}</Text>
        </View>
      </View>
      <View>
        <View>
          <Text>Change Logo</Text>
        </View>
        <View>
          <Text>Title</Text>
          <TextInput
            placeholder="Enter title here"
            defaultValue={title}
            onChangeText={(text) => setTitle(text)}
          />
        </View>
        <View>
          <Text>Sub Title</Text>
          <TextInput
            placeholder="Enter sub title here"
            defaultValue={subTitle}
            onChangeText={(text) => setSubTitle(text)}
          />
        </View>
      </View>
    </SafeAreaView>
  );
}

export default FormRFC;

```

The diagram illustrates the flow of data and state management in the `FormRFC` component using the `useState` hook.

- Format penulisan useState()**: A callout box points to the line `[getter, setter]=useState(value)`, explaining the syntax where `getter` and `setter` are generated by the hook.
- Inisialisasi title dan subtitle dengan Menggunakan HOOK useState**: A callout box points to the initial state assignments `const [title, setTitle] = useState("IBI Kesatuan");` and `const [subTitle, setSubTitle] = useState("Bogor Indonesia");`.
- Menampilkan nilai getter dari useState**: A callout box points to the two `<Text>{title}</Text>` and `<Text>{subTitle}</Text>` components, indicating they display the current values of `title` and `subTitle`.
- Mengisini nilai baru pada setter title atau subtitle dengan properties onChangeText()**: Two callout boxes point to the `onChangedText` properties of the `<TextInput>` components. One points to the `onChangedText={(text) => setTitle(text)}` in the `<Text>Title</Text>` section, and another points to the `onChangedText={(text) => setSubTitle(text)}` in the `<Text>Sub Title</Text>` section, both explaining how they update the state.

Gambar 5. 4 Penggunaan Hook

Dari script diatas maka outputnya tidak jauh berbeda:

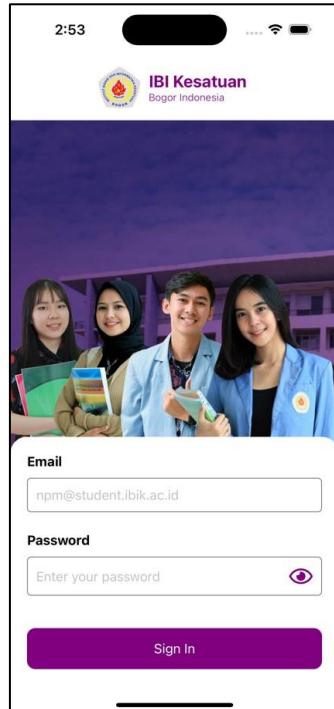


Gambar 5. 5 Tampilan penggunaan HOOK

### 5.2.3 Operations

#### A. Selection

Pada contoh kali ini akan menggunakan tugas latihan latihan sebelumnya, terdapat sebuah form Sign In yang berisikan Email dan Password. Berikut ini adalah contoh tampilan komponen Sign In dari materi sebelumnya:



Gambar 5. 6 Tampilan komponen Sign In

Pada gambar diatas akan dibuat sebuah validasi dimana pembuatan validasi pada form tersebut akan menggunakan operation statement, contoh kali ini akan menggunakan operation statement model **IF ELSE**.

### SignIn.js

```
import { View, Text, TextInput, StyleSheet, TouchableOpacity, Pressable } from
"react-native";
import React, { useState } from "react";
import FontAwesome5 from "react-native-vector-icons/FontAwesome5";

const SignInBasic = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  return (
    <View style={styles.container}>
      <View style={styles.frm_row}>
        <Text style={styles.text_label}>Email </Text>
        <TextInput
          placeholder="npm@student.ibik.ac.id"
          keyboardType="email-address"           autoCorrect={false}
          autoCapitalize="none"                 style={styles.text_input}
          defaultValue={email}                  onChangeText={(text) =>
            setEmail(text)}
          />
      </View>

      <View style={styles.frm_row}>
        <Text style={styles.text_label}>Password</Text>
        <View style={{
          ...styles.text_input,           flexDirection: "row",
          justifyContent: "space-between", }}>
          <TextInput
            secureTextEntry={true}
            placeholder="Enter your password"   autoCorrect={false}
            style={{ ...styles.text_input,       borderWidth:0,
                     padding:0,               width:"90%" }}
            defaultValue={password}
            onChangeText={(text)=>setPassword(text)}
          />
          <Pressable>
            <FontAwesome5 name={"eye"} size={25} color="purple" />
          </Pressable>
        </View>
      </View>
    </View>
  );
}

export default SignInBasic;
```

```

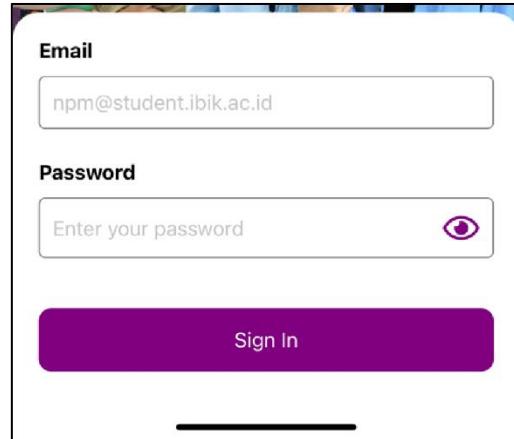
        <TouchableOpacity style={styles.btn_signin} onPress={()=> handlerSignIn()} >
            <Text style={styles.btn_signin_text}>
                Sign In
            </Text>
        </TouchableOpacity>
    </View>
);

};

export default SignInBasic;

const styles = StyleSheet.create({
    container: {
        padding: 20, backgroundColor: "white"
    },
    frm_row: { marginBottom: 15 },
    text_label: {
        fontWeight: "bold",
        marginBottom: 10,
        fontSize: 16,
    },
    text_input: {
        borderWidth: 1,
        borderColor: "grey",
        borderRadius: 5,
        padding: 10,
        fontSize: 16,
        color: "purple",
    },
    btn_signin: {
        backgroundColor: "purple",
        paddingVertical: 15,
        paddingHorizontal: 10,
        borderRadius: 10,
        marginTop: 15
    },
    btn_signin_text:{ color: "white", textAlign: "center", fontSize: 16 }
});

```



Gambar 5. 7 Skema dari RFC Sign In

Script diatas merupakan contoh bentuk skema dari RFC *Sign In* pada gambar 5.6, script tersebut akan dimodifikasi untuk membuat sebuah validasi, maka tahap pertama dari logika validasi ini ialah melakukan proses operation statement terhadap nilai masukan email dan

password. Berikut ini adalah contoh tahapan untuk memberikan sebuah validasi sekaligus contoh penggunaan **IF ELSE**.

1. Membuat sebuah function untuk mengelola logika validasi email, function ES6 yang akan dibuat untuk mengelola logika tersebut adalah *handlerValidMail()*.

```
const SignInBasic = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const handlerValidMail = (value)=>{
    //statement logic validasi email
  }
```

2. Di Dalam function tersebut memiliki parameter bernama value, dimana parameter tersebut yang akan menyimpan nilai balik dari TextInput email. Nilai balik tersebut akan dilakukan selection dengan IF ELSE. Kondisi pertama ialah, jika nilai parameter kosong maka mencetak "*This field is required*" pada JSX.

```
const [validEmail, setValidEmail] = useState("");
const handlerValidMail = (value)=>{
  if(value){ //if value is not null
    setValidEmail("");
  }else{
    setValidEmail("This field is required");
  }
}
```

Lakukan perubahan script pada JSX *TextInput* email dengan mengubah nama tujuan function yang lama menjadi *handlerValidMail(...)*. Dan buatlah JSX untuk menampilkan pesan error dari validasi email tersebut.

```

<View style={styles.frm_row}>
  <Text style={styles.text_label}>Email </Text>
  <TextInput
    placeholder="npm@student.ibik.ac.id"
    keyboardType="email-address"
    autoCorrect={false}
    autoCapitalize="none"
    style={styles.text_input}
    defaultValue={email}
    onChangeText={(text) => handlerValidMail(text)}
  />
  <Text style={{ color:"red" }}>{validEmail}</Text>
</View>

```

Mengubah tujuan nama function

Menampilkan pesan error yang telah disetter pada function handlerValidMain( ...) dgn memanggil getter dari validEmail

Maka output sementara dari seleksi kondisi pertama akan sebagai berikut:

3. Kondisi statement kedua kali ini akan mengecek apakah inputan email memiliki format yang benar atau tidak. Untuk mengecek format expression pada JAVASCRIPT dapat menggunakan Teknik REGEX.

Berdasarkan format expression REGEX yang dilihat dari [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular\\_expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions)  
Maka dapat expression kan untuk email menjadi:

Email	Expression Regex
<a href="mailto:name@domain.com">name@domain.com</a>	/^(([^<>()[]\.,;:\s@"]+(\.[^<>()[]\.,;:\s@"]+ \s@"]+)* (.+"))@((\[[0-9]{1,3}\].[0-9]{1,3}\.\[0-9]{1,3}\] ([a-zA-Z]-[0-9]+\.)+[a-zA-Z]{2,}))\$/

4. Berdasarkan konsep data regex diatas maka dapat kita buat selection untuk mengecek apakah nilai inputan email sudah benar atau tidak:

```

const [validEmail, setValidEmail] = useState("");
const handlerValidMail = (value)=>{
  if(value){ //if value is not null
    let regEmail = /^[^<>()\\[]\\.,;:\\s@"]+(\. [^<>()\\[]\\.,;:\\s@"]+)*$/
    if (!regEmail.test(value)) {
      setValidEmail("Invalid Email Address");
    } else {
      setValidEmail("");
    }
  }else{
    setValidEmail("This field is required");
  }
}

```

Maka output sementara ialah sebagai berikut:

The screenshot shows a simple form with a single input field labeled "Email". The input field contains the text "212310016@student.ibik.". Below the input field, a red error message "Invalid Email Address" is displayed.

5. Contoh kasus kedua ialah penggunaan selection dalam bentuk SWITCH CASE. Penggunaan selection bentuk kedua ini akan diimplementasikan untuk membuka dan tutup value pada isian password. Logika kali ini akan disimpan dengan sebuah function bernama *handlerOpenPassword()*.

Logika untuk membuka dan menutup isian password ini akan mengambil trigger ketika mengklik icon EYE maka akan membuka isian password begitu sebaliknya.

Tambahkanlah function dibawah ini pada script RFC SignIn sebelumnya:

```

const [isOpenPass, setOpenPass] = useState(true);
const handlerOpenPassword = () =>{
  switch (!isOpenPass) {
    case true:
      setOpenPass(true);
      break;
    default:
      setOpenPass(false);
      break;
  }
}

```

Pada script diatas selain menambahkan function baru bernama *handlerOpenPassword*, ditambahkan juga sebuah variable hook untuk menyimpan nilai Boolean terhadap aksi

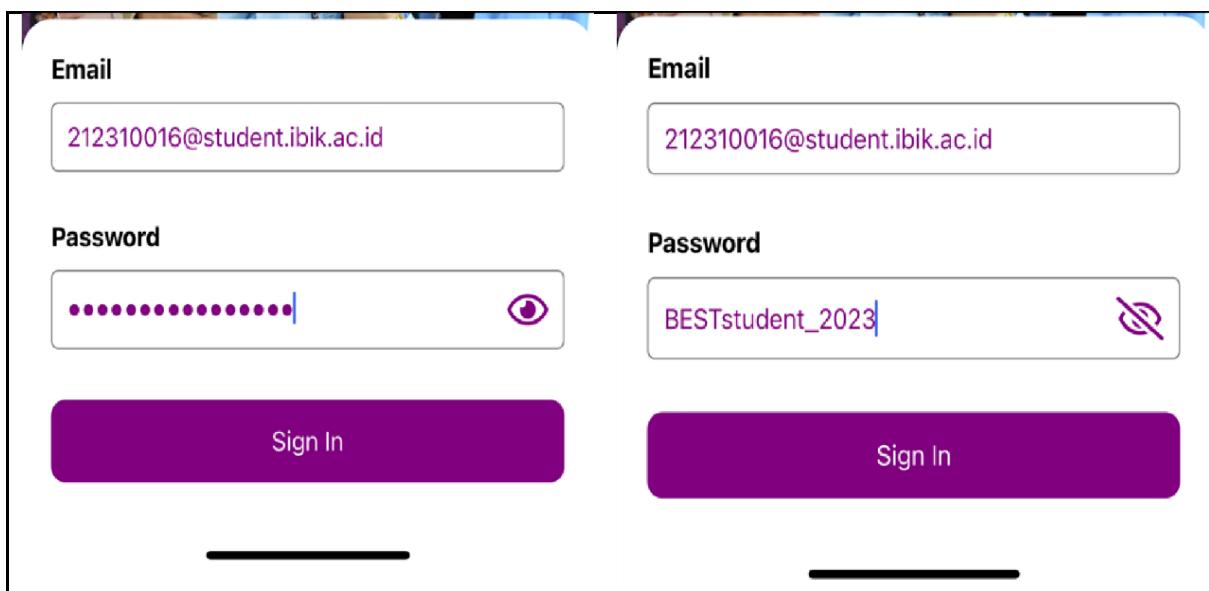
buka dan tutup password dengan nama isOpenPass sebagai current variable (getternya) dan setOpenPass sebagai bentuk pengisian nilai value baru (setternya).

Selanjutnya diperlukan untuk memperbarui elemen JSX bagian password agar dapat memperlihatkan isian dari TextInput dan memberikan trigger terhadap icon EYE agar dapat mengakses function *handlerOpenPassword()*.

```
<TextInput  
  secureTextEntry={isOpenPass}  
  placeholder="Enter your password"  
  autoCorrect={false}  
  style={{ ...styles.text_input, borderWidth:0, padding:0, width:"90%" }}  
  defaultValue={password}  
  onChangeText={(text)=>setPassword(text)}  
/>  
<Pressable onPress={()=>handlerOpenPassword()}>  
  <FontAwesome5 name={(isOpenPass) ? "eye":"eye-slash"} size={25} color="purple" />  
</Pressable>
```

Bentuk SHORT IF pada JSX

Dari aksi script diatas maka output sementara seperti berikut ini:



## B. Repetition

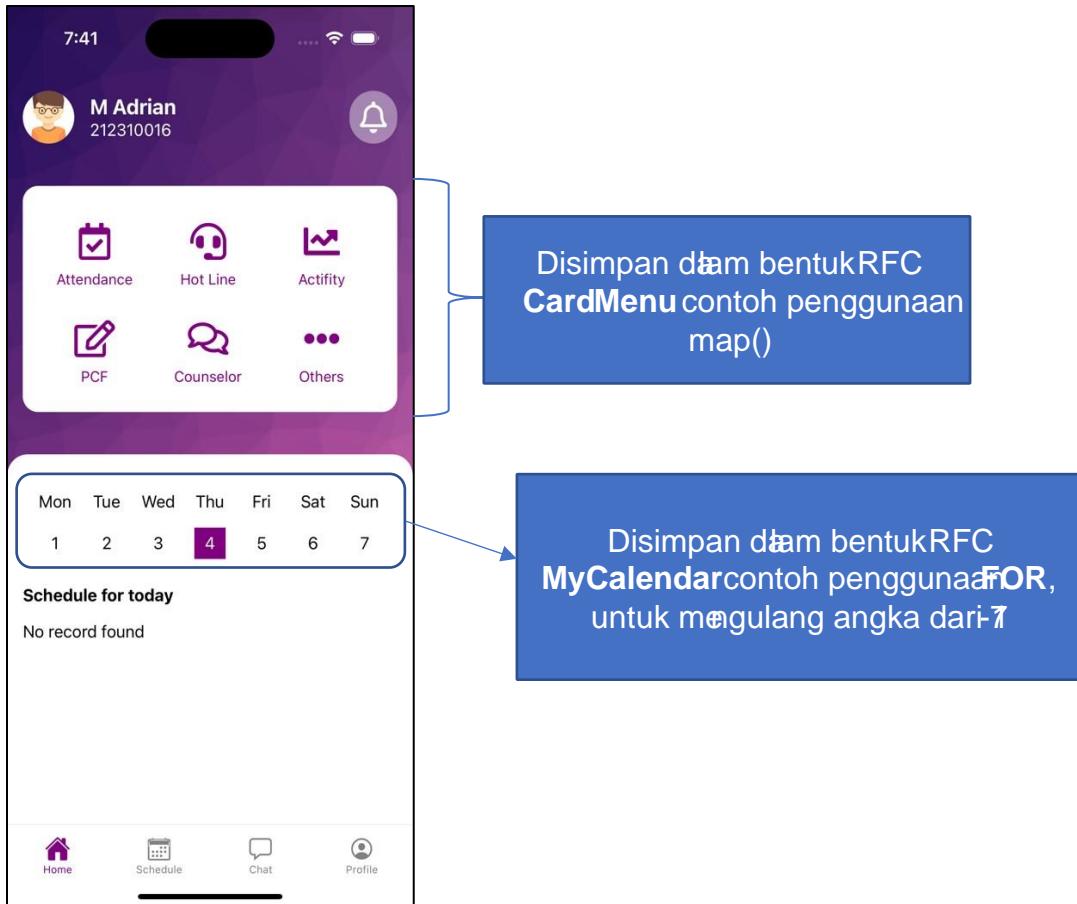
Untuk tipe-tipe dari repetition sama halnya dengan beberapa Bahasa pemrograman lainnya, yaitu memiliki bentuk :

- Do{ statement }while( condition )
- While( condition ){ statement }
- For(condition){ statement }, dan

Dari ketiga bentuk repetition diatas, sama halnya dengan selection. Bentuk repetition tersebut tidak bisa ditulis kedalam render JSX.

Untuk mengeksekusi perintah logika repetition disarankan membentuk function tersendiri dan mereturn nilai balik dalam bentuk JSX.

Namun pada JAVASCRIPT ada 1 bentuk repetition yang dapat digunakan didalam JSX ataupun diluar JSX, yaitu menggunakan map(). Berikut ini adalah contoh penggunaan repetition dalam bentuk FOR dan map:



*Tampilan penggunaan Repetition*

### CardMenu.js

```

import { View, Text, StyleSheet } from "react-native"; import
React from "react";
import FontAwesome5 from "react-native-vector-icons/FontAwesome5";

const CardMenu = () => {
  const menuList = [
    { id: 1, name: "Attendance", icon: "calendar-check" },
    { id: 2, name: "Hot Line", icon: "headset" },
    { id: 3, name: "Actifity", icon: "chart-line" },
    { id: 4, name: "PCF", icon: "edit" },
  ]
}

```

```

        { id: 5, name: "Counselor", icon: "comments" },
        { id: 6, name: "Others", icon: "ellipsis-h" },
    ];
}

n (
<View w>
    <View style={{ flexDirection:"row", flexWrap:"wrap" }}>
{menuList.map((v, index) => (
    <MenuItem item={v} key={index} />
))}

</View>
</View>
);

}
;

export default CardMenu;

const MenuItem = ({ item }) => {
return (
<View style={styles.card_item}>
    <View style={styles.card_item}>
        <FontAwesome5 name={item.icon} size={35} color={"purple"} />
    </View>
        <Text style={styles.card_text}>{item.name}</Text>
    </View>
);
}
;
```

```

const styles = StyleSheet.create({
  card_item: {
    width: 100, marginHorizontal: 5, marginVertical: 10, flexDirection: "column",
    justifyContent: "center", alignItems: "center"
  },
  card_icon: { marginBottom: 10 },
  card_text: { color: "purple", fontSize: 14 }
});

```

## MyCalendar.js

```

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

const CalendarFirstWeek = () => {
  const
  FirstWeek = () => {
    const days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]; var
    day = []; for (let index = 0; index < 7; index++) { day.push(
      <View key={index} style={styles.call_box}>
        <Text style={{ ...styles.call_text, marginBottom: 10 }}>
          {days[index]}</Text>
        <Text>{index + 1}</Text>
      </View>
    );
  }
}

```

```
        return day;
    };
    return (
        <View style={styles.callendar_container}>
            <FirstWeek />
        </View>
    );
}
;

export default CalendarFirstWeek;

const styles = StyleSheet.create({
    callendar_container: {
        flexDirection: "row",
        alignItems: "center",
        justifyContent: "center",
        marginVertical: 15,
    },
    call_box: {
        width: 50,
        flexDirection: "column",
        alignItems: "center",
    },
    call_text: {
        fontSize: 16,
        paddingVertical: 5,
    },
    call_curr: {
        padding: 10,
        backgroundColor: "purple",
        color: "white"
    }
});
```

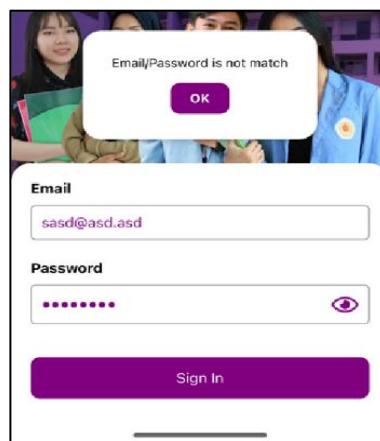
### C. Operasi Aritmatika

Bentuk dari operasi aritmatika pada JAVASCRIPT sebenarnya tidaklah terlalu berbeda dengan Bahasa pemrograman umum lainnya, berikut ini adalah contoh operasi aritmatika yang dapat digunakan pada JAVASCRIPT:

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
>	Lebih Besar
<	Lebih Kecil
>=	Lebih besar sama dengan
<=	Lebih kecil sama dengan
==	Sama dengan / equals
&	AND
	OR
++	Increments
--	Decrements
%	Modulus
Length	Mendapatkan nilai total data dari variable
Object.key()	Mendapatkan nama-nama key pada object
Object.value()	Mendapatkan nilai value pada object

### 5.3 Latihan Pembelajaran

1. Buatlah package baru bernama latihan-5.
2. Melanjutkan codingan, buatlah sebuah validasi untuk password dengan kondisi sebagai berikut:
3. Jika jumlah password kurang dari 3 maka menampilkan pesan "*Type minimum 3 character*".
4. Jika kondisi value password memiliki bentuk isian berupa: huruf, angka, dan simbol maka kondisi password dinyatakan lulus, namun jika tidak memenuhi kondisi ini buatlah pesan kesalahan "*Value must contain alphabet, number and symbol*"
5. Jika soal nomor 2 sudah anda kerjakan buatlah sebuah event handler untuk mengeksekusi kedua buah isian tersebut. Jika mengklik tombol SIGN IN maka akan mengeksekusi logika untuk:
  - a. Jika Email yang dimasukan berisi [212310016@student.ibik.ac.id](mailto:212310016@student.ibik.ac.id)
  - b. Dan password yang dimasukan berisi BESTstudent\_2023
  - c. Maka dinyatakan memenuhi kondisi transaksi, dan dapat di redirect ke halaman berikutnya. Namun jika tidak memenuhi kondisi diatas maka menampilkan pesan kesalahan dalam bentuk Pop Up "*Email/Password is not match*".



## **DAFTAR PUSTAKA**

Masiello, Eric. Dan Friedmann, Jacob. 2017. Mastering React Native. Packt Publishing

Ward, Dan. 2019. React Native Cookbook: Recipes for Solving Common. Packt Publishing Ltd.

Phang, Chong Lip. 2021. React, React Router, & React Native A Comprehensive & Comprehensible Guide. Chong Lip Phang.

Boduch, Adam. Derks, Roy. Dan Sakhniuk, Mickhail, 2022. React and React Native Build Cross-platform JavaScript Applications with Native Power for the Web, Desktop, and Mobile. Packt Publishing.

# **PEMROGRAMAN PERANGKAT BERGERAK**

---