

BAB III

LISTING DAN LIFE CYCLE METHOD

1. Install Library

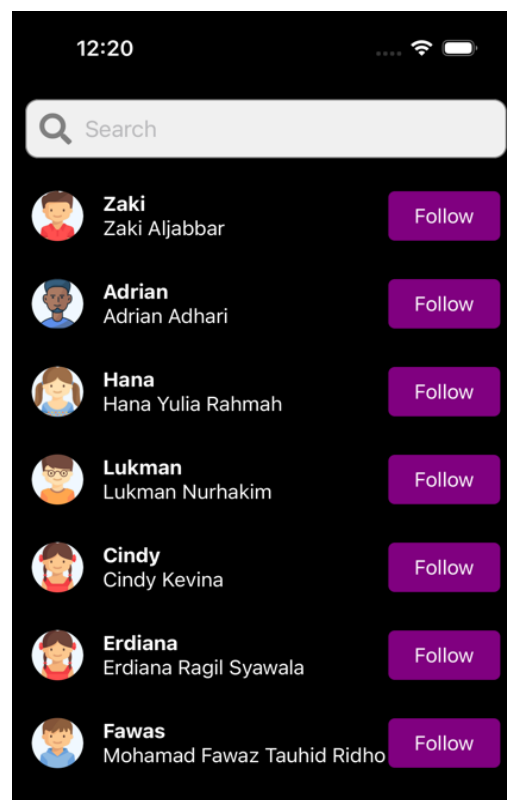
Pada materi kali ini memerlukan beberapa dependencies yang perlu digunakan, yaitu salah satunya ialah penggunaan icon pada aplikasi react native expo. Pada project kali ini akan memanfaatkan dependencies icon dari <https://oblador.github.io/react-native-vector-icons/> oleh karenanya bukalah terminal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

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

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

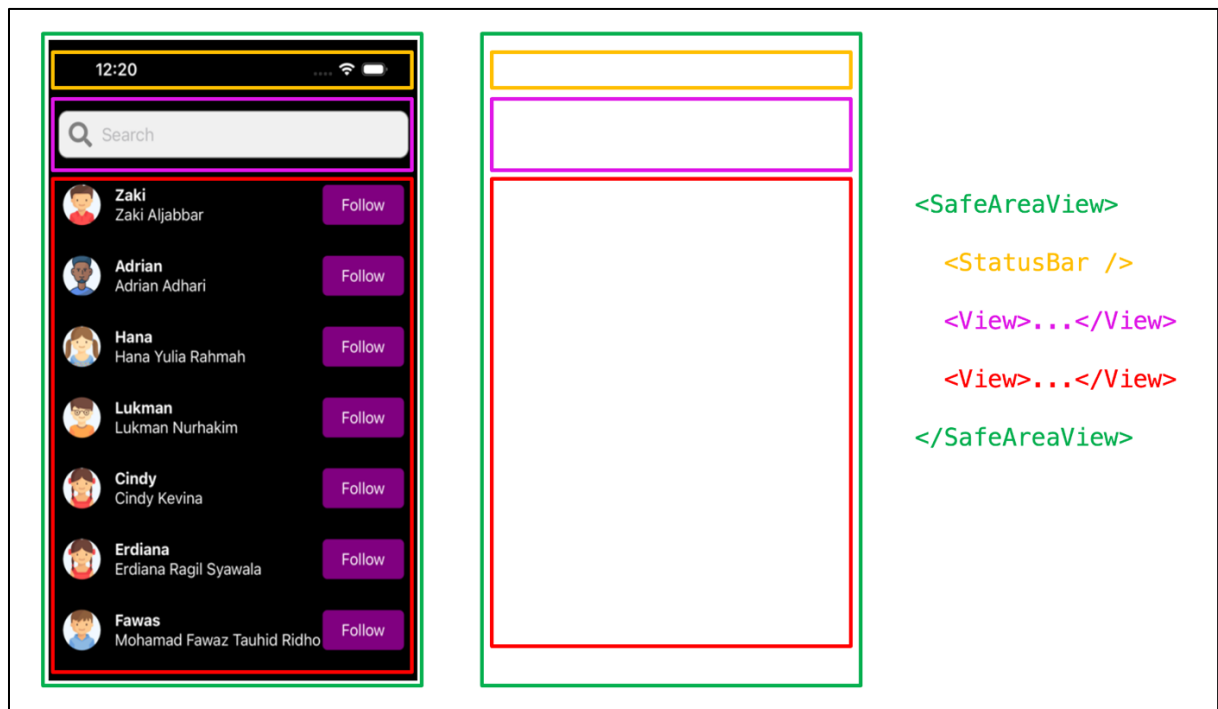
2. StyleSheet – Daftar Teman UI

Pada bagian ini akan membahas mengenai penggunaan Listing dan Life Cycle method pada react native expo cli. Namun sebelum kepembahasan materi, hal yang perlu dilakukan ialah membuat layouting pada aplikasi. Contoh kasus kali ini ialah membuat sebuah halaman daftar pertemanan. Jika merujuk pada desain dibawah ini:



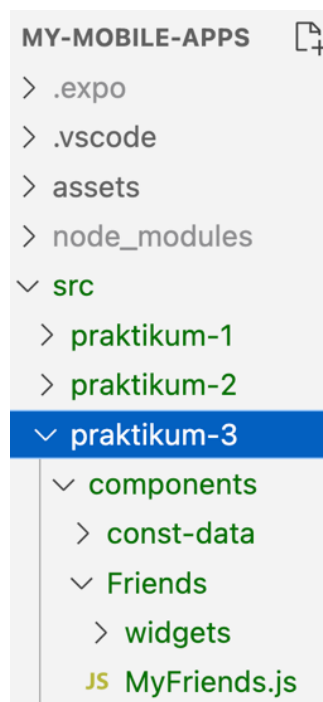
Gambar 1.a. Desain halaman daftar pertemanan

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



Gambar 1.b. 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 praktikum-3 didalam folder `./src/`.



Berdasarkan contoh gambar disamping mengenai structure folder pada project react native expo yang bernama MY-MOBILE-APPS.

Di dalam folder `src`, buatlah sebuah package folder baru bernama `praktikum-3/components` dan didalam folder `components` terdapat dua buah folder bernama `const-data` dan `Friends`.

Karna kita akan membuat titik awal aplikasi untuk halaman pertemanan olehkarenanya buatlah sebuah *React Class Component* didalam folder `Friends` bernama `MyFriends.js`

Masukan code JSX berdasarkan skema yang telah dibuat sesuai dengan *Gambar 1.b.* dan tambahkan stylesheet untuk membuat sketsa aplikasinya:

MyFriends.js

```
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"}
});
```

Pada sketsa scripting diatas, diamana titik awal aplikasi untuk halaman pertemanan ini dibuat dengan komponen class. Komponen ini menggunakan container SaveAreaView sebagai pembungkus pertama, didalamnya terdapat 3 buah elemen JSX yaitu ada StatusBar dan container View pertama 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 1.c. Output skema awal

Didalam folder Friends, buatlah sebuah folder bernama widgets. Dimana folder ini berisi file-file mentah komponen function react. Widget pertama yang akan dibuat ialah bagian header pada desain halaman pertemanan. Simpanlah *React Function Component* dengan nama `SearchBar.js` didalam 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 },
});
```

Pada script diatas terdapat syntax untuk menggunakan icon dari library yang telah diinstall yaitu `react-native-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.

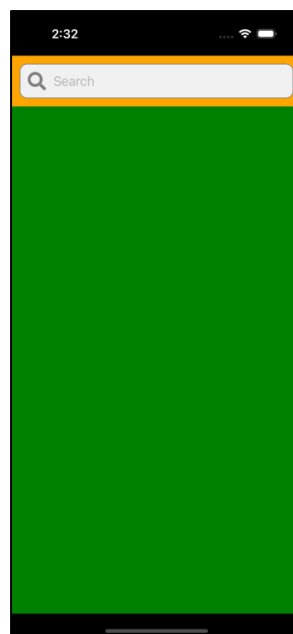
Setelah menambahkan script diatas silakan anda panggil RFC `SearchBar` didalam `RCC MyFriends` sesuai dengan posisi penempatan layoutnya.

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 ke-2

Maka output dari penampikan script diatas sebagai berikut:



Gambar 1.d. 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. 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 ialah 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.

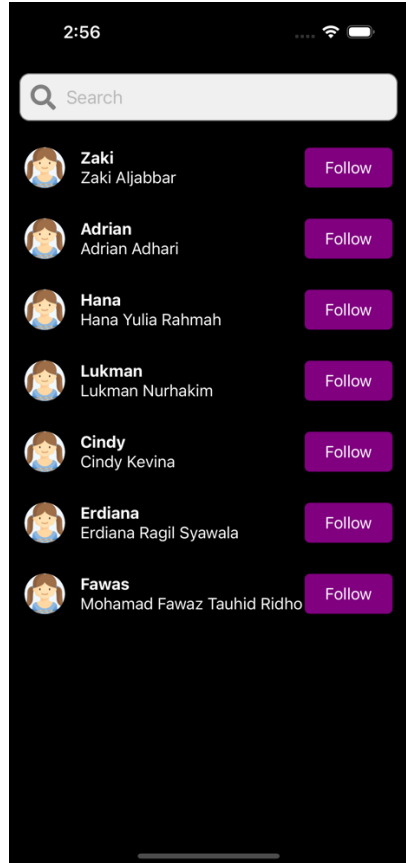
Sebelumnya, silakan anda siapkan folder `assets` untuk menyimpan file-file media seperti gambar. Folder `assets` ini sejajar dengan folder `src` pada project anda. Pada script dibawah ini pada emmet JSX Image memanggil sebuah file bernama `icon-girl-1.png` dimana file ini berada di folder `assets/icons/`.

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;
```

3.1. ScrollView

ScrollView merupakan elemen scroll secara umum yang dapat berisi banyak komponen dan tampilan. Item yang dapat digulir bisa berbeda-beda, dan dapat menggulir 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}) => { return (<ScrollView> {Users.map((v,index)=>{ <UserItem item={v} key={index} /> })} </ScrollView>) } export default ExpScrollView</pre> <pre>import { Users } from '../const-data/ObjDummies'; import ExpScrollView from './widgets/ExpScrollView'; 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}> <ExpScrollView Users={Users} /> </View> </SafeAreaView>) } }</pre>	

3.2. FlatList

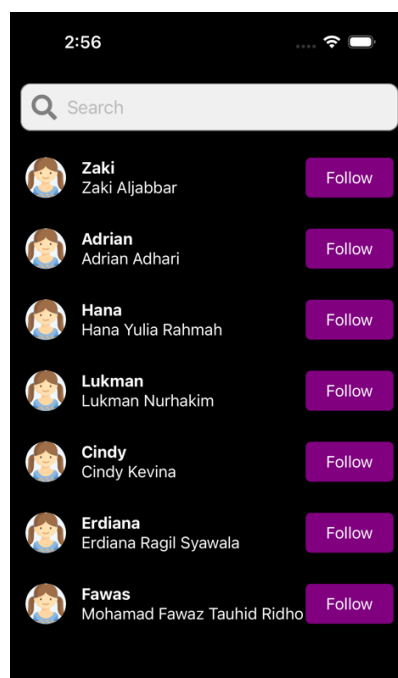
Penggunaan FlatList tidak jauh berbeda dengan ScrollView, pada emmed ini juga memiliki sebuah properti yang dapat digunakan untuk membentuk scroll dalam orientasi Horizontal maupun Vertikal. Cukup menambahkan property bernama *horizontal*={true/false}.

Namun yang berbeda ialah adanya property *renderItem*, property tersebut diperuntukan untuk mengirim sebuah object kedalam 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 3.2 Penggunaan FlatList

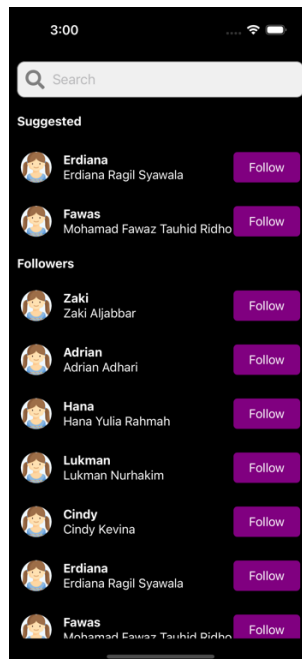
3.3. SectionList

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;
```

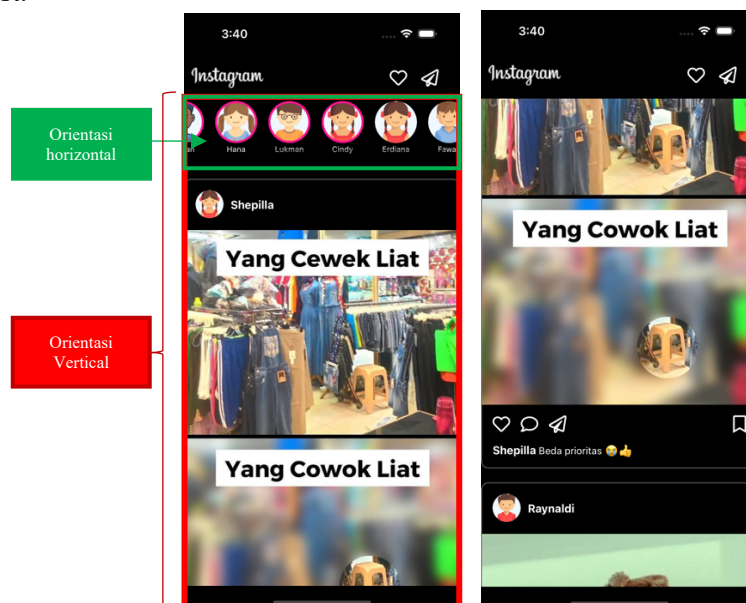



Gambar 1.3. Penerapan *SectionList*

Jika diperhatikan implementasi terhadap ketiga tipe listing, dari sisi script memiliki perbedaan dari cara merender atau menerima sebuah data hingga ditampilkannya. 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.

4. Latihan Praktikum

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





Matakuliah/Code
Dosen
Kelas

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3
: Irvan Rizky Ariansyah / Thesya Marcella
: TI-21-PA

Pengumpulan tugas Latihan praktikum dikumpulkan kedalam GITHUB masing-masing mahasiswa berdasarkan repository yang telah dibuat PPB-TI-21-[PA/KA]-NPM. File source code disimpan sesuai nama project-praktikum dan masukan kedalam repositori tersebut. Buatlah file dokumen dalam bentuk file pdf yang berisi Screen Capture dari hasil program yang telah dikerjakan. Simpan dalam file PDF tersebut kedalam project tersebut.

Tambahkan Collaborator management access pada repository anda kepada:

@FebryFairuz dan (@IrvanRizkyAriansyah atau @thesyamarcella)

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