M Reza Faisal
Ilmu Komputer, FMIPA, ULM

Multiple Screen

Instruksi

Aplikasi mobile umumnya terdiri atas beberapa screen atau halaman dengan tampilan yang berbeda-beda. React Native memiliki modul yang berfungsi untuk mengelola cara untuk menampilkan dan transisi antara screen. Modul tersebut adalah React Navigation. Modul ini merupakan solusi untuk library navigasi yang memungkinkan developer untuk keperluan yang disebutkan di atas.

1. Installasi

Untuk menggunakan modul React Navigation maka harus dilakukan menambah modul ini ke project. Untuk menambahkan modul ini perlu dilakukan installasi modul dengan cara berikut ini.

```
npm install @react-navigation/native @react-navigation/native-stack
```

Kemudian dilanjutkan dengan installasi modul pendukung dengan perintah berikut ini.

```
npm install react-native-screens react-native-safe-area-context
```

Hasil installasi ini dapat dilihat pada direktori `node_modules`.

Bagi pengguna Mac dan mengembangkan aplikasi mobile React Native untuk iOS maka perlu diinstall pod via Cooapods dengan perintah berikut ini.

```
npx pod-install ios
```

2. Multi Screen dalam 1 File

Berikut ini adalah contoh sederhana membuat aplikasi multi screen dengan navigasi agar bisa saling pindah ke masing-masing screen yang dibuat. Langkah pertama adalah membuat

Pertama yang perlu ditambahkan pada kedua baris berikut ini yang berfungsi untuk menggunakan library React Navigation.

Kode 1. Import library React Navigation.

```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
```

Pada contoh ini akan dibuat dua screen. Screen pertama adalah `HomeScreen` yang akan menjadi screen utama. Screen yang kedua adalah `AboutScreen`. Untuk implementasi navigasi maka dibuat obyek `Stack` dengan menggunakan `createNativeStackNavigator()` seperti yang ditampilkan pada kode baris ke-1. Kemudian pada komponen `AppHome` didaftarkan komponen untuk menjadi screen yang akan dikelola oleh aplikasi. Pada kode di bawah ini dapat dilihat ada dua komponen yang didaftarkan yaitu `HomeScreen` seperti yang terlihat pada baris ke-7 dan `AboutScreen` seperti yang terlihat pada baris ke-8. Untuk menentukan komponen mana yang akan ditampilkan terlebih dahulu saat aplikasi dijalankan dapat dilihat pada baris ke-6. Penentuan komponen ini dituliskan pada atribut component. Sedangkan atribut name berfungsi sebagai alias untuk memanggil screen tersebut.

Kode 2. Komponen AppHome.

```
1  const Stack = createNativeStackNavigator();
2
3  const AppHome = () => {
4      return (
5          <NavigationContainer>
6            <Stack.Navigator initialRouteName="Home">
7              <Stack.Screen name="Home" component={HomeScreen} />
8              <Stack.Screen name="About" component={AboutScreen} />
9            </Stack.Navigator>
10         </NavigationContainer>
11     );
12 }
13
14 export default AppHome;
```

Selanjutnya dibuat komponen `HomeScreen` dengan kode seperti berikut ini.

Kode 3. Komponen HomeScreen.

```
1  const HomeScreen = ({ navigation }) => {
2      return (
3          <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
4          <Text>Home Screen</Text>
5          <Button
6              title="Go to About"
7              onPress={() => navigation.navigate('About')}
8          />
9          </View>
10     );
11 }
```

Pada komponen ini terdapat prop dengan nama navigation seperti yang terlihat pada baris ke-1 pada Kode 3. Prop `navigation` memiliki berbeberapa fungsi yang salah satunya adalah `navigate()` seperti yang terlihat pada baris ke-7. Fungsi-fungsi yang lain adalah sebagai berikut:

- `navigate`, fungsi ini berfungsi untuk pindah ke screen lain dengan menggunakan nama alias sesuai yang telah didaftarkan pada Kode 2.
- `reset`, berfungsi untuk menghapus state navigator dan mengganti dengan route baru.
- `goBack`, menutup screen yang sedang aktif dan menampilkan kembali screen dari stack sebelumnya.
- `setParams`, mengganti parameter route.
- `dispatch`, mengupdate state navigation dengan mengirim obyek action.
- `setOptions`, mengupdate opsi screen.
- `isFocused`, memeriksa apakah screen sedang dipilih.
- `addListener`, subscribe untuk mengupdate event dari navigator.

Pada baris ke-7, pada komponen `Button` di atas dapat dilihat ada event `onPress` yang berisi fungsi `navigate('About')`, artinya ketika komponen `Button` ditekan maka screen `AboutScreen` akan ditampilkan.

Berikutnya adalah membuat komponen `AboutScreen` dengan kode berikut ini. Penjelasan kode di bawah ini sama dengan kode komponen `HomeScreen` sebelumnya.

Kode 4. AboutScreen.

```
const AboutScreen = ({ navigation }) => {
    return (
        <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
        <Text>About Screen</Text>
        <Button
            title="Go to Home"
            onPress={() => navigation.navigate('Home')}
        />
        </View>
    );
}
```

Pada Kode 5 adalah kode lengkap dari potongan-potongan kode di atas.

Kode 5. AppHome.js

AppHome.js

```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { View, Text, Button } from 'react-native';

const HomeScreen = ({ navigation }) => {
    return (
      <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
        <Text>Home Screen</Text>
        <Button
          title="Go to About"
          onPress={() => navigation.navigate('About')}
        />
      </View>
    );
  }

const AboutScreen = ({ navigation }) => {
    return (
      <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
        <Text>About Screen</Text>
        <Button
          title="Go to Home"
          onPress={() => navigation.navigate('Home')}
        />
      </View>
    );
  }

const Stack = createNativeStackNavigator();

const AppHome = () => {
    return (
        <NavigationContainer>
          <Stack.Navigator initialRouteName="Home">
            <Stack.Screen name="Home" component={HomeScreen} />
            <Stack.Screen name="About" component={AboutScreen} />
          </Stack.Navigator>
        </NavigationContainer>
    );
}
export default AppHome;
```

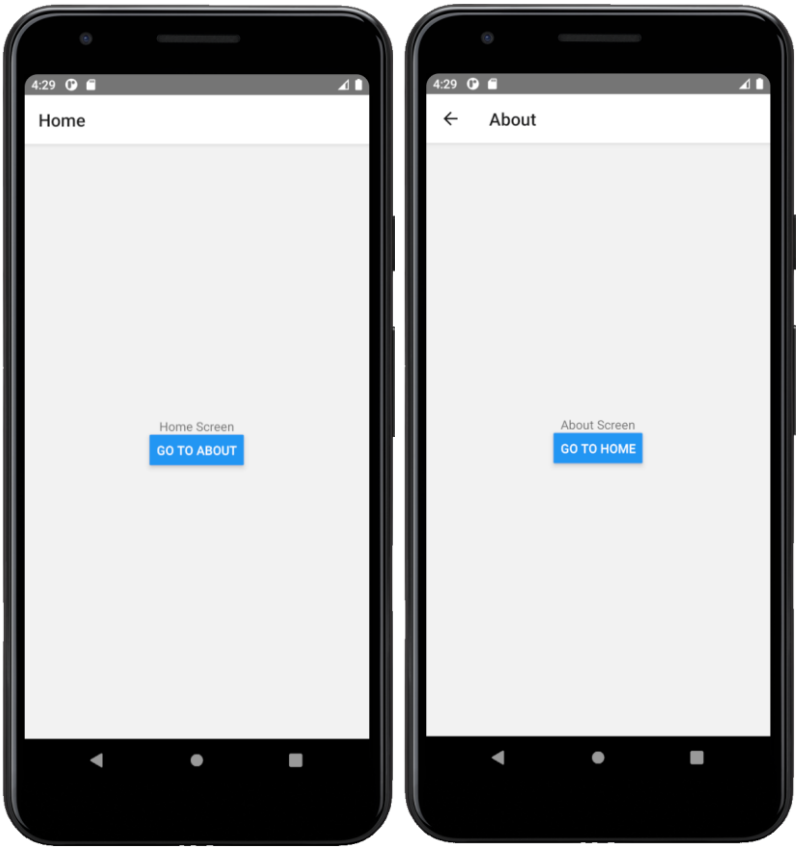
Kemudian jangan lupa untuk mengupdate file `index.js` menjadi seperti pada Kode 6.

Kode 6. index.js untuk AppHome.

index.js

```
1  /**
2   * @format
3   */
4
5  import {AppRegistry} from 'react-native';
6  import AppHome from './AppHome';
7  import {name as appName} from './app.json';
8
9  AppRegistry.registerComponent(appName, () => AppHome);
```

Hasilnya dapat dilihat pada Gambar 1. Saat aplikasi dijalankan maka akan ditampilkan screen dari komponen `HomeScreen` seperti yang terlihat pada gambar sebelah kiri. Jika tombol GO TO ABOUT diklik maka akan ditampilkan screen dari komponen `AboutScreen`. Sedangkan jika tombol GO TO HOME pada screen `AboutScreen` diklik maka screen `HomeScreen` akan ditampilkan kembali.



Gambar 1. Hasil AppHome.js.

3. Multi Screen dalam Multi File

Pada contoh di atas komponen `HomeScreen`, `AboutScreen` dan `AppHome` disimpan di sebuah file yang sama. Seperti yang telah dipelajari pada hands on labs sebelumnya bahwa komponen dapat disimpan pada file berbeda kemudian diimport oleh file yang ingin menggunakannya.

Pada sub bab ini dicontohkan memecah ketiga komponen yang disebutkan di atas menjadi tiga file. sebagai contoh nama ketiga file tersebut adalah:

- `ScreenUtama.js` untuk menyimpan komponen `AppHome`.
- `Screen1.js` untuk menyimpan komponen `HomeScreen`.
- `Screen2.js` untuk menyimpan komponen `AboutScreen`.

Pada Kode 7 adalah isi dari file `Screen1.js`.

Kode 7. Screen1.js

Screen1.js

```
1  import * as React from 'react';
2  import { View, Text, Button } from 'react-native';
3
4  export const HomeScreen = ({ navigation }) => {
5      return (
6          <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
7          <Text>Home Screen</Text>
8          <Button
9              title="Go to About"
10             onPress={() => navigation.navigate('About')}
11         />
12         </View>
13     );
14 }
```

Pada Kode 8 adalah isi dari file `Screen2.js`

Kode 8. Screen2.js

Screen2.js

```
1  import * as React from 'react';
2  import { View, Text, Button } from 'react-native';
3
4  export const AboutScreen = ({ navigation }) => {
5      return (
6          <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
7          <Text>About Screen</Text>
8          <Button
9              title="Go to Home"
10             onPress={() => navigation.navigate('Home')}
11         />
12         </View>
13     );
14 }
```

Pada Kode 9 adalah membuat file utama aplikasi yang akan menggunakan kedua komponen di atas.

Kode 9. ScreenUtama.js

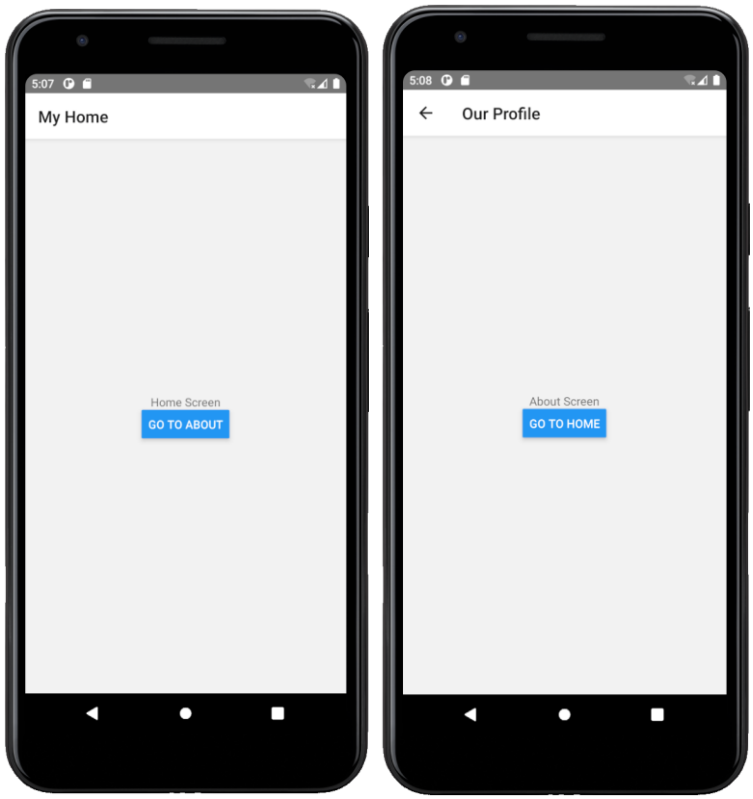
ScreenUtama.js

```
1  import * as React from 'react';
2  import { NavigationContainer } from '@react-navigation/native';
3  import { createNativeStackNavigator } from '@react-navigation/native-stack';
4  import {HomeScreen} from './Screen1'
5  import {AboutScreen} from './Screen2'
6
7  const Stack = createNativeStackNavigator();
8
```

```
9   const AppHome = () => {
10      return (
11          <NavigationContainer>
12            <Stack.Navigator initialRouteName="Home">
13              <Stack.Screen name="Home" component={HomeScreen} options={{ title: 'My Home'
14  }}/>
15                <Stack.Screen name="About" component={AboutScreen} options={{ title: 'Our
16  Profile' }}/>
17            </Stack.Navigator>
18          </NavigationContainer>
19      );
20  }
21
22  export default AppHome;
```

Pada baris ke-2 dan ke-3 adalah cara untuk mengimport module React Navigation. Kemudian pada baris ke-4 adalah cara untuk memanggil komponen `HomeScreen` yang ada pada file `Screen1.js`. Kemudian pada baris ke-5 adalah cara untuk memanggil komponen `AboutScreen` yang ada pada file `Screen2.js`.

Selanjutnya pada komponen `AppHome` dapat dilihat komponen `NavigationContainer`. Pada baris ke-13 adalah mendaftarkan screen, pada baris ini ditambahkan atribut `options` yang berisi `title` dan nilainya. Ini adalah cara untuk menentukan title dari screen. Hasilnya dapat dilihat bagian atas pada Gambar 2.



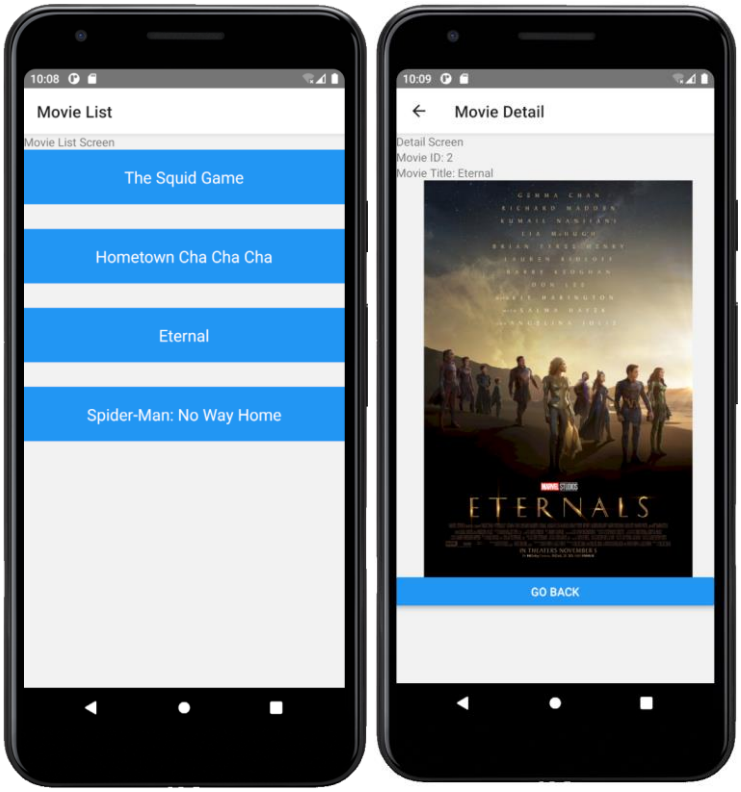
Gambar 2. Hasil ScreenUtama.js.

4. Mengirim Parameter

Saat pindah screen, umumnya juga dapat terjadi pengiriman parameter dari sebuah screen ke screen tujuan. Sebagai contoh sebuah screen menampilkan daftar film kemudian user mengklik salah satu film pada daftar tersebut. Selanjutnya user akan melihat detail dari film yang dia klik.

Contoh yang lain pada timeline Intagram ditampilkan daftar gambar dari banyak user Instagram. Kemudian user mengklik icon dari user pemilik gambar tersebut, maka akan ditampilkan profile dari user tersebut.

Berikut ini dibuat dua komponen yang akan menjadi dua screen. `Screen3.js` (gambar sebelah kiri pada Gambar 3) adalah komponen yang menampilkan daftar film. Sedangkan `Screen4.js` (gambar sebelah kanan pada Gambar 3) adalah komponen yang menampilkan detail dari film yang dipilih pada `Screen3.js`.



Gambar 3. Hasil Screen3.js dan Screen4.js.

Pada Kode 10 adalah isi dari file `Screen3.js`. Nama komponen dari file ini adalah `ListScreen` seperti yang dapat dilihat pada baris ke-4. Komponen-komponen yang digunakan pada file ini adalah `View`, `Text`, `StyleSheet` dan `TouchableOpacity` seperti yang dapat dilihat pada baris ke-2.

Kode 10. Screen3.js

Screen3.js

```
1  import * as React from 'react';
2  import { View, Text, StyleSheet, TouchableOpacity } from 'react-native';
3
4  export const ListScreen = ({ navigation }) => {
```

```
5       const movies = [
6           {"id":0, "value":'The Squid Game'},
7           {"id":1, "value":'Hometown Cha Cha Cha'},
8           {"id":2, "value":'Eternal'},
9           {"id":3, "value":'Spider-Man: No Way Home'},
10        ];
11
12      return (
13          <View style={{ flex: 1}}>
14              <Text>Movie List Screen</Text>
15              { movies.map((item) => (
16                  <TouchableOpacity key={item.id} onPress={() =>
17  navigation.navigate('DetailScreen', {itemId:item.id})}>
18                      <View key={item.id} style={styles.button}>
19                          <Text style={styles.buttonText}
20  key={item.id}>{item.value}</Text>
21                      </View>
22                  </TouchableOpacity>
23              ))}
24          </View>
25      );
26  }
27
28  const styles = StyleSheet.create({
29      container: {
30        paddingTop: 60,
31        alignItems: 'center'
32      },
33      button: {
34        marginBottom: 30,
35        alignItems: 'center',
36        backgroundColor: '#2196F3'
37      },
38      buttonText: {
39        textAlign: 'center',
40        padding: 20,
41        color: 'white',
42        fontSize: 20
43      }
44    });
```

Data yang digunakan pada halaman ini adalah array yang berisi obyek seperti yang dapat dilihat pada baris ke-5 sampai dengan ke-10. Data disimpan ke dalam variable `movies`. Kemudian untuk menampilkan data dari obyek `movies` digunakan array method `.map()` seperti yang dapat dilihat pada ke-15. Pada baris ini dapat dilihat penggunaan komponen `TouchableOpacity` yang berfungsi sebagai komponen yang dapat ditekan. Kemudian pada komponen ini disisipkan event handler `onPress` yang menjalankan fungsi `navigation.navigate()`. Fungsi `navigate()` terdiri atas 2 nilai, yang pertama adalah `DetailScreen` yang merupakan nama komponen yang disimpan pada file `Screen4.js`. Kemudian nilai kedua adalah nilai parameter yang akan dikirimkan ke komponen `DetailScreen` (`Screen4.js`), nilai yang dikirimkan adalah nilai yang disimpan pada variable `item.id`.

Pada adalah kode dari file `Screen4.js`. Nama komponen dari file ini adalah `DetailScreen` seperti yang dapat dilihat pada baris ke-4. Pada komponen ini memiliki dua parameter yaitu `route` dan `navigation`. Parameter `route` digunakan untuk mendapatkan parameter yang dikirimkan dari komponen sebelumnya. Dapat dilihat pada baris ke-5 cara untuk mendapatkan nilai parameter dengan menggunakan `route.params`.

Kode 11. Screen4.js.

Screen4.js

```
1  import * as React from 'react';
2  import { View, Text, Button, Image } from 'react-native';
3
4  export const DetailScreen = ({ route, navigation }) => {
5      const { itemId } = route.params;
6
7      const movies = [
8          {"id":0, "value":'The Squid Game', "img":require('./images/img1.png')},
9          {"id":1, "value":'Hometown Cha Cha Cha', "img":require('./images/img2.png')},
10         {"id":2, "value":'Eternal', "img":require('./images/img3.png')},
11         {"id":3, "value":'Spider-Man: No Way Home', "img":require('./images/img4.png')},
12       ];
13
14       const selectedMovie = movies[itemId];
15
16       return (
17         <View style={{ flex: 1 }}>
18         <Text>Detail Screen</Text>
19         <Text>Movie ID: {selectedMovie.id}</Text>
20         <Text>Movie Title: {selectedMovie.value}</Text>
21         <Image source={selectedMovie.img} style={{width: 400, height: 490,
22 resizeMode:'contain'}} />
23         <Button
24             title="Go Back"
25             onPress={() => navigation.goBack()}
26         />
27         </View>
28     );
29 }
```

Pada kode di atas digunakan data yang lebih lengkap seperti pada baris ke-7 sampai dengan ke-12. Data ini mendapat tambahan atribut `img` yang menyimpan gambar dari film. Selanjutnya pada pada baris ke-14 dapat dilihat cara sederhana untuk mendapatkan obyek film yang dipilih. Karena variable `movies` adalah array maka untuk mendapatkan item yang dipilih adalah dengan cara menggunakan `itemId` sebagai index array yang dipilih. Obyek film yang dipilih kemudian disimpan pada obyek `selectedMovie`. Untuk menampilkan nilai dari obyek yang dipilih dapat dilihat pada baris ke-19 sampai dengan baris ke-21. Pastikan pada project telah ditambahkan folder `images` yang didalamnya disimpan file poster file dengan nama file adalah `img1.png`, `img2.png`, `img3.png` dan `img4.png`.

Selanjutnya adalah mengedit `file ScreenUtama.js` menjadi seperti pada Kode 12.

Kode 12. ScreenUtama.js dengan tambahan Screen3.js dan Screen4.js.

ScreenUtama.js

```
1  import * as React from 'react';
2  import { NavigationContainer } from '@react-navigation/native';
3  import { createNativeStackNavigator } from '@react-navigation/native-stack';
4  import {HomeScreen} from './Screen1'
5  import {AboutScreen} from './Screen2'
6  import {ListScreen} from './Screen3'
7  import {DetailScreen} from './Screen4'
8
9  const Stack = createNativeStackNavigator();
10
11 const AppHome = () => {
```

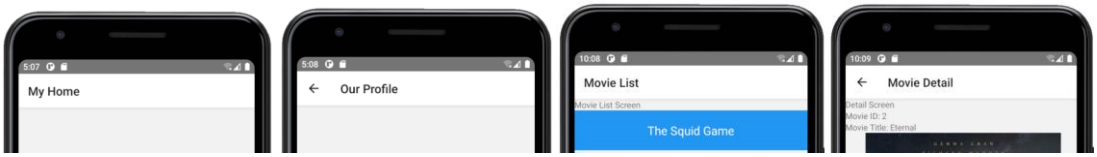
```
12      return (
13          <NavigationContainer>
14            <Stack.Navigator initialRouteName="MovieList" screenOptions={{headerShown:
15  true}}>
16              <Stack.Screen name="Home"
17                            component={HomeScreen}
18                            options={{
19                              title: 'My Home',
20                           }}/>
21              <Stack.Screen name="About" component={AboutScreen} options={{ title: 'Our
21  Profile' }}/>
22              <Stack.Screen name="MovieList" component={ListScreen} options={{ title:
23  'Movie List' }}/>
24              <Stack.Screen name="DetailScreen" component={DetailScreen} options={{ title:
25  'Movie Detail' }}/>
26            </Stack.Navigator>
27          </NavigationContainer>
28      );
29  }
30
31  export default AppHome;
```

Tambahan pada file ini dapat dilihat pada baris ke-6 dan ke-7 yang berfungsi untuk mengimport komponen `ListScreen` dan `DetailScreen`. Selanjutnya kedua komponen tersebut didaftarkan seperti yang dapat dilihat pada baris ke-22 sampai baris ke-25. Pada baris ke22, komponen `ListScreen` diberi nama alias adalah `MovieList` seperti yang terlihat pada nilai atribut name. Sedangkan nama alias komponen `DetailScreen` tetap sama seperti yang terlihat nilai atribut name di baris ke-24.

Selanjutnya untuk membuat komponen `ListScreen` sebagai komponen yang dipanggil pertama kali maka pada baris ke-14 dapat dilihat atribut `initialRouteName` diberi nilai dengan nama komponen `MovieList`, nama ini sesuai dengan nama alias yang diberikan kepada komponen ini.

5. Header Style

Pada beberapa contoh di atas telah digunakan atribut option dengan isi adalah title, sebagai contoh pada Kode 12 baris ke-19, baris ke- 21, baris ke-22 dan baris ke-24. Hasilnya dapat dilihat pada Gambar 4.



Gambar 4. Title pada header.

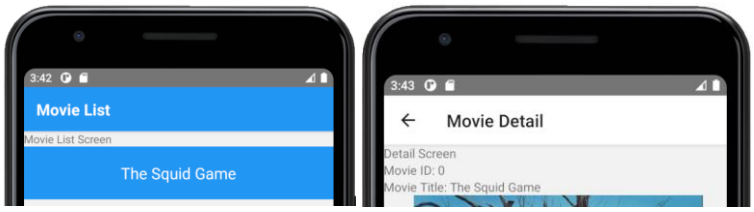
Selain itu kita juga dapat memberikan style pada header dengan menggunakan property berikut ini `headerStyle`, `headerTintColor`, dan `headerTitleStyle`. Penggunaan property ini juga diberikan pada atribut option seperti halnya property `title`.

Pada Kode 13 dapat dilihat bagaimana membuat style pada header sebuah screen yaitu screen komponen `ListScreen`.

Kode 13. Potongan isi file ScreenUtama.js untuk membuat style pada header komponen ListScreen.

```
<Stack.Screen name="MovieList"
              component={ListScreen}
              options={{ title: 'Movie List',
                         headerStyle: {backgroundColor: '#2196F3'},
                         headerTintColor: '#fff',
                         headerTitleStyle: {fontWeight: 'bold'} }}/>
```

Pada property `headerStyle` dapat berisi banyak atribut style yang dapat diisikan sesuai dengan keperluan, begitu juga property `headerTitleStyle`, sehingga nilai dari property ini menggunakan tanda { }. Sedangkan property `headerTintColor` cukup diisi dengan sebuah nilai saja. Hasilnya dapat dilihat pada gambar sebelah kiri di Gambar 5. Header style ini hanya untuk komponen `ListScreen` saja namun tidak untuk header style pada komponen `DetailScreen`.



Gambar 5. Header style pada komponen ListScreen.

Jika ingin seluruh screen memiliki header style yang seragam maka pada menggunakan atribut `screenOption` yang disisipkan pada `<Stack.Navigator>` seperti yang terlihat pada potongan kode di Kode 14.

Kode 14. Potongan isi file ScreenUtama.js untuk membuat style pada seluruh header screen.

```
<Stack.Navigator initialRouteName="MovieList"
                 screenOptions={{headerShown: true,
                                 headerStyle: {backgroundColor: '#2196F3'},
                                 headerTintColor: '#fff',
                                 headerTitleStyle: {fontWeight: 'bold'}}}>
```

6. Menyisipkan Komponen pada Header

Selain title, pada header juga dapat disisipkan komponen. Seperti yang telah dijelaskan di atas, komponen dapat disisipkan per screen atau disisipkan untuk seluruh screen.

Menyisipkan Komponen pada Sebuah Screen

Hal pertama yang dilakukan adalah membuat komponen yang akan disisipkan. Untuk contoh pertama adalah menyisipkan komponen pada sebuah screen saja yaitu pada `DetailScreen`. Berikut adalah komponen yang akan disisipkan.

Kode 15. Komponen AppHeaderWoLogo.

```
const AppHeaderWoLogo = () => {
  return(
    <View style={{ flex: 1,
                   width:'auto',
                   flexDirection:'row',
                   marginLeft:-15,
                   paddingLeft:0 }}>
```

```
      <TextInput placeholder='ketik kata kunci di sini'
                 style={{width:'66%',
                         height:40,
                         backgroundColor:'white',
                         marginTop:5,
                         marginLeft:10,
                         marginRight:10}} />
      <Image style={{ width: 50, height: 50 }}
             source={require('./images/search.png')} />
    </View>
  );
}
```

Selanjutnya berikut adalah cara untuk menyisipan komponen tersebut.

Kode 16. Cara menyisipkan komponen pada sebuah screen.

```
1 <Stack.Screen name="DetailScreen"
2               component={DetailScreen}
3               options={{ title: 'Movie Detail',
4                          headerTitle: (props) => <AppHeaderWoLogo {...props}/> }}/>
```

Pada baris ke-2 dapat dilihat nama komponen/screen yang akan disisipkan komponen. Kemudian pada baris ke-4 dilihat cara untuk menyisipkan komponen `AppHeaderWoLogo` yang menjadi nilai property `headerTitle` dengan menggunakan arrow function.

Menyisipkan Komponen pada Seluruh Screen

Berikut ini adalah cara untuk menyisipkan komponen pada seluruh screen. Langkah pertama adalah membuat komponen berikut ini.

Kode 17. Komponen AppHeader.

```
const AppHeader = () => {
  return(
    <View style={{ flex: 1,
                   width:'auto',
                   flexDirection:'row',
                   marginLeft:-15,
                   paddingLeft:0 }}>
      <Image style={{ width: 50,
                      height: 50 }}
                      source={require('./images/telu.png')} />
      <TextInput placeholder='ketik kata kunci di sini'
                 style={{width:'66%',
                         height:40,
                         backgroundColor:'white',
                         marginTop:5,
                         marginLeft:10,
                         marginRight:10}} />
      <Image style={{ width: 50, height: 50 }}
             source={require('./images/search.png')} />
    </View>
  );
}
```

Untuk menyisipkan kompoen AppHeader pada seluruh screen dilakukan dengan cara berikut.

```
1  <NavigationContainer>
2    <Stack.Navigator initialRouteName="MovieList"
3                     screenOptions={{headerShown: true,
4                                     headerStyle: {backgroundColor: '#ec3237'},
5                                     headerTintColor: '#fff',
6                                     headerTitleStyle: {fontWeight: 'bold'},
7                                     headerTitle: (props) => <AppHeader {...props} />}}>
```

Untuk menyisipkan komponen pada seluruh screen maka penulisan komponen pada bagian `<Stack.Navigator>`. Seperti cara penyisipan per screen pada sub bab sebelumnya, di sini juga digunakan property `headerTitle` yang menjadi nilai pada atribut `screenOptions` seperti yang dapat dilihat pada baris ke-7.

Pada Kode 18 dapat dilihat kode lengkap dari kedua sub bab di atas.

Kode 18. ScreenUtama.js yang diedit untuk menambahkan komponen pada header.

ScreenUtama.js

```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import {HomeScreen} from './Screen1'
import {AboutScreen} from './Screen2'
import {ListScreen} from './Screen3'
import {DetailScreen} from './Screen4'
import { TextInput, View, Image } from 'react-native';

const Stack = createNativeStackNavigator();

const AppHome = () => {
    return (
      <NavigationContainer>
        <Stack.Navigator initialRouteName="MovieList"
                         screenOptions={{headerShown: true,
                                         headerStyle: {backgroundColor: '#ec3237'},
                                         headerTintColor: '#fff',
                                         headerTitleStyle: {fontWeight: 'bold'},
                                        headerTitle: (props) => <AppHeader {...props} />}}>
            <Stack.Screen name="Home"
                          component={HomeScreen}
                          options={{ title: 'My Home' }}/>
            <Stack.Screen name="About"
                          component={AboutScreen}
                          options={{ title: 'Our Profile' }}/>
            <Stack.Screen name="MovieList"
                          component={ListScreen}
                          options={{ title: 'Movie List',
                                     headerStyle: {backgroundColor: '#ec3237'},
                                     headerTintColor: '#fff',
                                     headerTitleStyle: {fontWeight: 'bold'} }}/>
            <Stack.Screen name="DetailScreen"
                          component={DetailScreen}
                          options={{ title: 'Movie Detail',
                                  headerTitle: (props) => <AppHeaderWoLogo {...props}/> }}/>
          </Stack.Navigator>
        </NavigationContainer>
    );
}
export default AppHome;
```

```
const AppHeader = () => {
  return(
    <View style={{ flex: 1,
                   width:'auto',
                   flexDirection:'row',
                   marginLeft:-15,
                   paddingLeft:0 }}>
      <Image style={{ width: 50,
                      height: 50 }}
                      source={require('./images/telu.png')} />
      <TextInput placeholder='ketik kata kunci di sini'
                 style={{width:'66%',
                         height:40,
                         backgroundColor:'white',
                         marginTop:5,
                         marginLeft:10,
                         marginRight:10}} />
      <Image style={{ width: 50, height: 50 }}
             source={require('./images/search.png')} />
    </View>
  );
}

const AppHeaderWoLogo = () => {
  return(
    <View style={{ flex: 1,
                   width:'auto',
                   flexDirection:'row',
                   marginLeft:-15,
                   paddingLeft:0 }}>
      <TextInput placeholder='ketik kata kunci di sini'
                 style={{width:'66%',
                         height:40,
                         backgroundColor:'white',
                         marginTop:5,
                         marginLeft:10,
                         marginRight:10}} />
      <Image style={{ width: 50, height: 50 }}
             source={require('./images/search.png')} />
    </View>
  );
}
```

Pada kode di atas dapat dilihat terdapat dua komponen yaitu `AppHeader` dan `AppHeaderWoLogo`. Komponen `AppHeader` disisipkan untuk ditampilkan pada seluruh screen. Sedangkan `AppHeaderWoLogo` disisipkan hanya untuk screen `DetailScreen` saja. Walau seluruh screen telah disisipkan dengan komponen `AppHeader`, namun karena pada screen `DetailScreen` juga disisipkan komponen pada header maka screen ini akan memiliki header yang berbeda dibandingkan screen lainnya.

Tugas

Lakukan tugas-tugas berikut:

1. Ikuti seluruh instruksi di atas.
2. Pilih salah satu aplikasi mobile yang sudah ada dan pernah anda gunakan untuk dibuat ulang tampilkannya dengan React Native (misal Instragram, Facebook, Twitter dan lain-lain). Minimal terdiri atas 3 screen.
3. Buat aplikasi multi screen yang terdiri atas beberapa screen, screen pertama menampilkan nama kelompok kalian beserta daftar anggotanya. Kemudian setiap item pada daftar anggota dapat diklik yang akan mengantarkan ke screen yang memberikan informasi detail dari anggota yang dipilih tersebut. Buat antarmuka yang bagus dan semenarik mungkin dengan memanfaatkan komponen-komponen yang telah disediakan pada React Native.

Referensi

1. https://reactnative.dev/docs/navigation
2. https://reactnavigation.org/