M Reza Faisal
Ilmu Komputer, FMIPA, ULM

Full Stack Development – Bagian 2

Pada hands on labs ini akan diberikan contoh dan langkah-langkah untuk mengakses REST API pada backend dari aplikasi mobile yang dibangun dengan React Native. Penjelasan lengkap dari tentang hal ini dapat dilihat pada video berikut: https://youtu.be/LZFekwXY2pU.

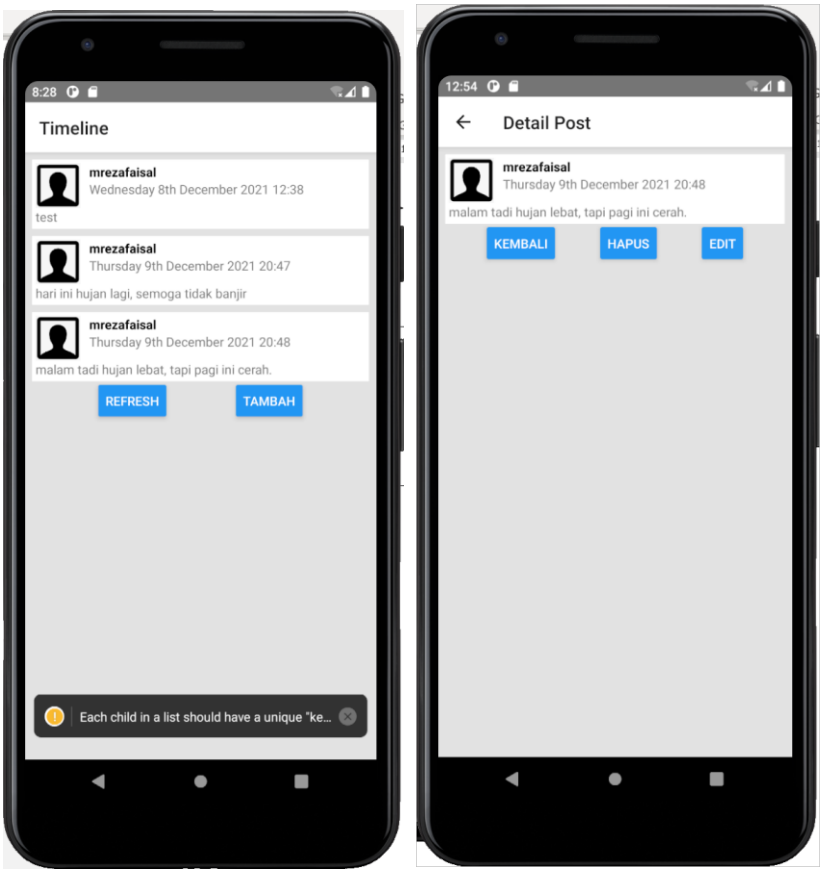
Instruksi

3. Frontend – Aplikasi Mobile dengan React Native

Aplikasi ini akan menggunakan endpoint pada backend yang telah dibuat pada hands on labs sebelumnya.

Retrieve Data

Saat aplikasi dijalankan maka ditampilkan data dari tabel posts berdasarkan username tertentu seperti yang terlihat pada antarmuka sebelah kiri pada Gambar 1.



Gambar 1. Retrieve Data.

Antarmuka daftar post tersebut merupakan component yang disimpan pada file PostList.js dengan isi yang dapat dilihat pada Kode 1.

Kode 1. PostList.js

PostList.js

```
1  import * as React from 'react';
2  import { useEffect, useState } from 'react';
3  import { ActivityIndicator, TouchableOpacity, Text, View, Image, Button } from 'react-
4  native';
5
6  export const PostList = ({ navigation }) => {
7      const [data, setData] = useState([]);
8      const [username, setUsername] = useState("mrezafaisal");
9
10     const getData = async () => {
11         try {
12            const response = await fetch('http://10.0.2.2:3000/posts/username/'+username);
13             const json = await response.json();
14             setData(json.data);
15         } catch (error) {
16             console.error(error);
17         }
18     }
19
20     useEffect(() => {
21         getData();
22     }, []);
23
24     return (
25         <View style={{ flex: 1, padding: 4, backgroundColor: '#E2E2E2' }}>
26             {
27             data.map((item) => (
28                     <TouchableOpacity onPress={() => navigation.navigate('PostDetail',
29 {itemId:item.post_id})}>
30                     <View style={{backgroundColor: 'white', margin: 4, padding: 4}}>
31                         <View style={{flexDirection:'row'}}>
32                             <View>
33                             <Image source={{uri:'http://10.0.2.2:3000/images/user.png'}}
34 style={{width: 50, height: 50, resizeMode:'contain'}}/>
35                             </View>
36                             <View style={{marginLeft: 10}}>
37                                                 <Text  style={{fontWeight:'bold',
38 color:'black'}}>{item.username}</Text>
39                                 <Text>{item.post_date}</Text>
40                             </View>
41                         </View>
42                         <View>
43                             <Text>{item.post}</Text>
44                         </View>
45                     </View>
46                 </TouchableOpacity>
47             ))}
48             <View style={{flexDirection:'row', justifyContent:'space-evenly'}}>
49                 <Button title="refresh" onPress={() => getData()}/>
50                 <Button title="tambah" onPress={() => navigation.navigate('PostAdd')} />
51             </View>
52         </View>
53     );
54 }
55
56
57
58
```

Dari daftar post yang pada antarmuka sebelah kiri pada Gambar 1 dapat dipilih sebuah post dengan cara menekan salah satu post tersebut sehingga dapat dilihat detailnya dengan antarmuka sebelah kanan pada Gambar 1.

Untuk menampilkan sebuah post yang dipilih digunakan komponen PostDetail.js dengan isi kode seperti yang dilihat pada Kode 2.

Kode 2. PostDetail.js

PostDetail.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { View, Text, Button, Image } from 'react-native';

export const PostDetail = ({ route, navigation }) => {
    const { itemId } = route.params;
    const [isLoading, setLoading] = useState(true);
    const [data, setData] = useState([]);

    const [username, setUsername] = useState("");
    const [postDate, setPostDate] = useState("");
    const [post, setPost] = useState("");

    const getData = async () => {
        try {
            const response = await fetch('http://10.0.2.2:3000/posts/id/'+itemId);
            const json = await response.json();
            setData(json.data);
        } catch (error) {
            console.error(error);
        } finally {
            setLoading(false);
        }
    }

    const deleteData = async () => {
        try {
             const response = await fetch('http://10.0.2.2:3000/posts/id/'+itemId, {method:
'DELETE'});
        } catch (error) {
            console.error(error);
        } finally {
            navigation.navigate('PostList');
        }
    }

    useEffect(() => {
        getData();
    }, []);

    return (
        <View style={{ flex: 1, padding: 4, backgroundColor: '#E2E2E2' }}>
            {
            data.map((item) => (
            <View style={{backgroundColor: 'white', margin: 4, padding: 4}}>
                <View style={{flexDirection:'row'}}>
                    <View>
```

```
                                <Image  source={{uri:'http://10.0.2.2:3000/images/user.png'}}
style={{width: 50, height: 50, resizeMode:'contain'}}/>
                    </View>
                    <View style={{marginLeft: 10}}>
                      <Text style={{fontWeight:'bold', color:'black'}}>{item.username}</Text>
                        <Text>{item.post_date}</Text>
                    </View>
                </View>
                <View>
                    <Text>{item.post}</Text>
                </View>
            </View>
            ))}
            <View style={{flexDirection:'row', justifyContent:'space-evenly'}}>
                <Button title="kembali" onPress={() => navigation.navigate('PostList')} />
                <Button title="hapus" onPress={() => deleteData()} />
               <Button title="edit" onPress={() => navigation.navigate('PostEdit', {itemId})}
/>
            </View>
        </View>
    );
}
```

Komponen yang menampilkan daftar post masih ada kekurangan saat pertama kali dijalankan, kekurangannya berupa munculnya pesan peringatan yang ditampilkan pada bagian bawah antarmuka seperti yang terlihat pada gambar di sebelah kanan pada Gambar 1.

Penjelasan tentang kode di atas dapat dilihat pada video berikut:

- Frontend dengan Aplikasi Mobile berbasis React Native Part 1: Menampilkan Data (https://youtu.be/QOG0H6Lqffo)
- Frontend dengan Aplikasi Mobile berbasis React Native Part 3: Menampilkan Detail & Menghapus Record (https://youtu.be/WWHaYxZbqIs).

Delete Data

Proses menghapus data dapat dilihat pada komponen yang menampilkan detail data. Proses menghapus data dilakukan ketika tombol Hapus pada komponen PostDetail.js diklik dengan kode yang dapat dilihat pada Kode 3. Kemudian user akan diantarkan ke antarmuka dari komponen PostList.js.

Kode 3. PostDetail.js – fungsi deleteData().

PostDetail.js

```
const deleteData = async () => {
    try {
          const  response  =  await  fetch('http://10.0.2.2:3000/posts/id/'+itemId,  {method:
'DELETE'});
    } catch (error) {
        console.error(error);
    } finally {
        navigation.navigate('PostList');
    }
}
```

Kekurangan dari proses ini adalah tidak adanya window konfirmasi yang menanyakan kepada user untuk memastikan apakah user yakin untuk menghapus data tersebut. Kekurangan yang lain dapat dilihat pada

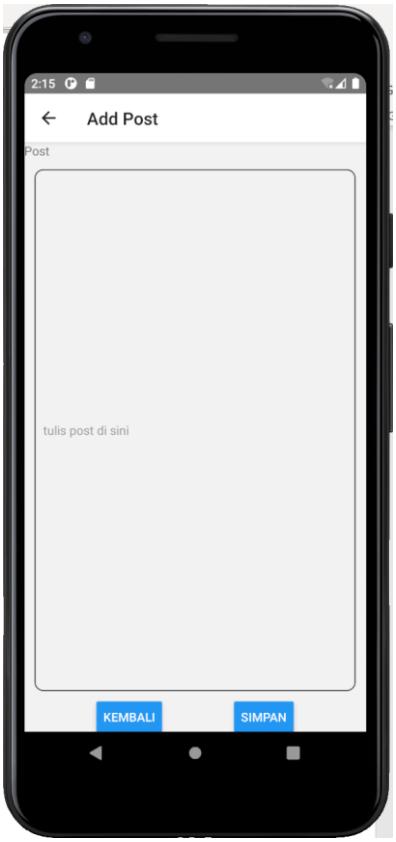
komponen PostList.js juga terlihat setelah data berhasil dihapus, daftar data baru terupdate setelah tombol tombol Refresh ditekan.

Penjelasan tentang kode di atas dapat dilihat pada video berikut:

- Frontend dengan Aplikasi Mobile berbasis React Native Part 3: Menampilkan Detail & Menghapus Record (https://youtu.be/WWHaYxZbqIs).

Create Data

Untuk menambah data digunakan komponen FormAdd.js dengan tampilan seperti pada Gambar 2.



Gambar 2. Tambah Data.

Pada Kode 4 adalah isi dari kode komponen FormAdd.js.

Kode 4. FormAdd.js

FormAdd.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { View, Text, Button, StyleSheet, TextInput, Alert } from 'react-native';


export const PostAdd = ({ route, navigation }) => {
    const [username, setUsername] = useState("mrezafaisal");
    const [textInputPost, setTextInputPost] = useState("");
```

```
    const saveData = async () => {
        try {
            const response = await fetch('http://10.0.2.2:3000/posts/', {
                method: 'POST',
                headers: {
                  Accept: 'application/json',
                  'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    username: 'mrezafaisal',
                    post: textInputPost
                })
            });
            //const json = await response.json();
            //setData(json.data);
        } catch (error) {
            console.error(error);
        } finally {
            navigation.navigate('PostList');
        }
    }

    return (
        <View style={{ flex: 1 }}>
            <Text>Post</Text>
           <TextInput placeholder='tulis post di sini' style={styles.input} onChangeText={text
=> setTextInputPost(text)} value={textInputPost} />
            <View style={{flexDirection:'row', justifyContent:'space-evenly'}}>
                <Button title="kembali" onPress={() => navigation.navigate('PostList')} />
                <Button title="simpan" onPress={() => saveData()}/>
            </View>
        </View>
    );
}

const styles = StyleSheet.create({
    input: {
        height: 40,
        margin: 12,
        borderWidth: 1,
        padding: 10,
        borderRadius:10,
        flex:1,
    },
    button: {
        height: 40,
        margin: 12,
        borderWidth: 1,
        padding: 10,
        borderRadius:10,
        backgroundColor: '#68a0cf',
        borderColor: '#fff',
        flex:1,
    },
    logo: {
        height:75,
        width:300,
        resizeMode:'contain',
        flex:1,
    },
```

```
});
```

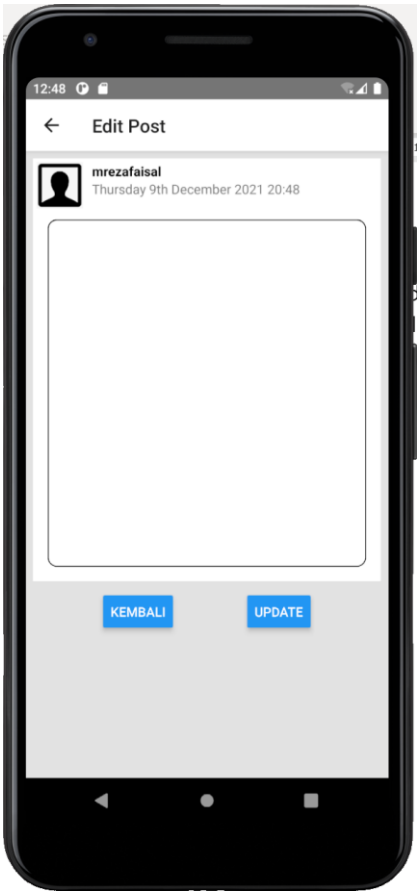
Setelah tombol Simpan diklik maka data disimpan dan antarmuka dikembalikan ke PostList.js. Jika tombol Kembali diklik maka antarmuka juga akan dikembalikan ke PostList.js.

Bagian ini masih memiliki kekurangan yaitu setelah data disimpan daftar post tidak langsung terupdate. Record yang baru ditambahkan baru terlihat jika tombol Refresh pada PostList.js diklik.

Penjelasan tentang kode di atas dapat dilihat pada video berikut:

- Frontend dengan Aplikasi Mobile berbasis React Native Part 2: Menambah Data (https://youtu.be/u4s4NkWPjAw).

Edit Data



Gambar 3. Edit Data.

Edit data dilakukan dengan memilih sebuah post pada daftar post kemudian ditampilkan detail dari data seperti yang dapat di Gambar 1. Kemudian dengan mengklik tombol Edit maka akan ditampilkan komponent PostEdit.js yang terlihat pada Gambar 3. Kode dari komponen ini dapat dilihat pada Kode 5.

Kode 5. PostEdit.js

PostEdit.js

```
import * as React from 'react';
import { useEffect, useState } from 'react';
import { View, Text, Button, StyleSheet, TextInput, Image } from 'react-native';

export const PostEdit = ({ route, navigation }) => {
    const { itemId } = route.params;
    const [textInputPost, setTextInputPost] = useState("");
    const [isLoading, setLoading] = useState(true);
    const [data, setData] = useState([]);
    const [username, setUsername] = useState("");
    const [postDate, setPostDate] = useState("");
    const [postMessage, setPostMessage] = useState("");

    const getData = async () => {
        try {
            const response = await fetch('http://10.0.2.2:3000/posts/id/'+itemId);
            const json = await response.json();
            setData(json.data);
        } catch (error) {
            console.error(error);
        } finally {
            setLoading(false);
        }
    }

    const updateData = async () => {
        try {
            const response = await fetch('http://10.0.2.2:3000/posts/id/'+itemId, {
                method: 'PUT',
                headers: {
                  Accept: 'application/json',
                  'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    username: 'mrezafaisal',
                    post: textInputPost
                })
            });
        } catch (error) {
            console.error(error);
        } finally {
            navigation.navigate('PostList');
        }
    }

    useEffect(() => {
        getData();
    }, []);

    return (
        <View style={{ flex: 1, padding: 4, backgroundColor: '#E2E2E2' }}>
            {
            data.map((item) => (
            <View style={{backgroundColor: 'white', margin: 4, padding: 4}}>
                <View style={{flexDirection:'row'}}>
                    <View>
                                <Image  source={{uri:'http://10.0.2.2:3000/images/user.png'}}
style={{width: 50, height: 50, resizeMode:'contain'}}/>
```

```
                    </View>
                    <View style={{marginLeft: 10}}>
                      <Text style={{fontWeight:'bold', color:'black'}}>{item.username}</Text>
                        <Text>{item.post_date}</Text>
                    </View>
                </View>
                <View style={{height: 400}}>
                                    <TextInput   style={styles.input}   onChangeText={text   =>
setTextInputPost(text)} value={textInputPost} />
                </View>
            </View>
            ))}
            <View style={{flexDirection:'row', justifyContent:'space-evenly', marginTop:11}}>
                <Button title="kembali" onPress={() => navigation.navigate('PostList')} />
                <Button title="update" onPress={() => updateData()} />
            </View>
        </View>
    );
}

const styles = StyleSheet.create({
    input: {
        height: 40,
        margin: 12,
        borderWidth: 1,
        padding: 10,
        borderRadius:10,
        flex:1,
    },
});
```

Penjelasan tentang kode di atas dapat dilihat pada video berikut:

- Frontend dengan Aplikasi Mobile berbasis React Native Part 4: Mengedit Data (https://youtu.be/7ALg3gcXowU)

Halaman Utama Multi Screen

Berikutnya tambahkan Kode 6 sebagai komponen utama.

Kode 6. SocialMediaApp.js

SocialMediaApp.js

```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import {PostList} from './PostList';
import {PostDetail} from './PostDetail';
import {PostAdd} from './PostAdd';
import {PostEdit} from './PostEdit';

const Stack = createNativeStackNavigator();

const AppHome = () => {
    return (
        <NavigationContainer>
            <Stack.Navigator initialRouteName="PostList">
```

```
                <Stack.Screen name="PostList"
                              component={PostList}
                              options={{ title: 'Timeline' }}/>
                <Stack.Screen name="PostDetail"
                              component={PostDetail}
                              options={{ title: 'Detail Post' }}/>
                <Stack.Screen name="PostAdd"
                              component={PostAdd}
                              options={{ title: 'Add Post' }}/>
                <Stack.Screen name="PostEdit"
                              component={PostEdit}
                              options={{ title: 'Edit Post' }}/>
            </Stack.Navigator>
        </NavigationContainer>
    );
}

export default AppHome;
```

Entry Point

Selanjutnya adalah mengedit file index.js pada project agar komponen dari file SocialMediaApp.js dipanggil pertama kali dan dijalankan. pada

Kode 7. index.js

index.js

```
import {AppRegistry} from 'react-native';
import AppHome from './SocialMediaApp';
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, () => AppHome);
```

Tugas

Lakukan tugas-tugas berikut:

1. Ikuti seluruh instruksi di atas.
2. Perbaiki kekurangan dari setiap komponen yang telah disebutkan di atas!
3. Perbaiki antarmuka dari setiap komponen agar terlihat lebih bagus, indah dan seperti aplikasi mobile pada umumnya!
4. Tambahkan fitur-fitur yang umum ada pada media sosial seperti registrasi, login, logout atau fitur-fitur umum lainnya!

Referensi

1. http://expressjs.com/
2. https://reactnative.dev/docs/network