LAPORAN HASIL PRAKTIKUM
PEMROGRAMAN WEB & MOBILE 1

NAMA : SATRIA SEPTA ARIANTO

NIM : 193020503026

KELAS : A

MODUL : VI (SEARCH BY FLATLIST)

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKA RAYA

TAHUN 2021

BAB I
TUJUAN DAN LANDASAN TEORI

1.1. Tujuan

1.1.1 Mahasiswa mampu memahami penggunaan Flatlist pada React Native.

1.1.2 Mahasiswa mampu memahami penggunaan Search pada Flatlist.

1.1.3 Mahasiswa mampu membuat program dengan Flatlist pada React Native.

1.1.4 Mahasiswa mampu membuat program Search pada Flatlist.

1.2. Landasan Teori

Komponen FlatList ini adalah cara yang efisien untuk membuat daftar data yang bergulir di aplikasi React Native. Komponen ini memiliki API sederhana untuk bekerja dengan dan menampilkan sejumlah besar informasi.

Secara default, Anda cukup mengirimkan array data dan komponen ini akan melakukan tugasnya. Anda biasanya tidak perlu mengurus format data (*How to Add a Search Bar in a FlatList in React Native Apps*, n.d.).

Penggunaan dasar komponen FlatList

Ada tiga alat peraga utama yang diperlukan komponen FlatList untuk menampilkan daftar data:

- **data** : larik data yang digunakan untuk membuat daftar. Umumnya, array ini dibangun dari banyak objek.
- **renderItem** : adalah fungsi yang mengambil elemen individu dari datalarik dan membuatnya di UI.
- **keyExtractor** : ini memberi tahu daftar data untuk menggunakan pengenal unik atau iduntuk elemen individu.

Contoh penggunaan komponen Flatlist (*FlatList · React Native*, n.d.):

```
const DATA = [
  {
    id: 'bd7acbea-c1b1-46c2-aed5-3ad53abb28ba',
```

```
    title: 'First Item',
  },
  {
    id: '3ac68afc-c605-48d3-a4f8-fbd91aa97f63',
    title: 'Second Item',
  },
  {
    id: '58694a0f-3da1-471f-bd96-145571e29d72',
    title: 'Third Item',
  },
];
```

```
const App = () => {
  const renderItem = ({ item }) => (
    <Item title={item.title} />
  );

  return (
    <SafeAreaView style={styles.container}>
      <FlatList
        data={DATA}
        renderItem={renderItem}
        keyExtractor={item => item.id}
      />
    </SafeAreaView>
  );
}
```

SearchBar Filter untuk ListView

Pada dasarnya, SearchBar pada ListView bekerja secara real-time untuk mencari data yang diinginkan. Karena jika kita memiliki daftar panjang di

aplikasi maka sangat merepotkan untuk mencari data yang diperlukan dengan menggulir seluruh daftar. Konsep pencarian pada daftar yang ada biasanya mengacu pada salah satu data yang ada pada daftar sehingga data lainnya juga dapat muncul, seperti konsep primary key pada database (*Searching Using Search Bar Filter in React Native List View - About React*, n.d.).

Untuk Pencarian Real-Time di ListView menggunakan SearchBar Filter

Logikanya sederhana **:**

- Memuat daftar dari panggilan jaringan dan kemudian menunjukkannya kepada pengguna.
- Pengguna dapat mencari data dengan memasukkan teks ke dalam TextInput.
- Setelah memasukkan teks SearchFilterFunction akan dipanggil
- Membandingkan data daftar dengan data yang dimasukkan dan akan membuat sumber Data baru.
- Memperbarui sumber data yang dilampirkan ke ListView.
- Ini akan merender ulang daftar dan pengguna akan dapat melihat data yang difilter.

```
const searchFilterFunction = (text) => {
  // Check if searched text is not blank
  if (text) {
    // Inserted text is not blank
    // Filter the masterDataSource and update FilteredDataSource
    const newData = masterDataSource.filter(
      function (item) {
        // Applying filter for the inserted text in search bar
        const itemData = item.title
            ? item.title.toUpperCase()
            : ''.toUpperCase();
        const textData = text.toUpperCase();
```

```
        return itemData.indexOf(textData) > -1;
      }
    );
    setFilteredDataSource(newData);
    setSearch(text);
  } else {
    // Inserted text is blank
    // Update FilteredDataSource with masterDataSource
    setFilteredDataSource(masterDataSource);
    setSearch(text);
  }
};
```

BAB II
PEMBAHASAN

Untuk membuat program react native menggunakan search pada flat list. Maka diperlukan sebuah kumpulan data yang akan dimasukkan pada program ini. Disini saya akan menggunakan data pada https://randomuser.me/ untuk mendapatkan profil orang dan datanya secara random dari seluruh dunia. Sedangkan, untuk program ini sendiri, ada beberapa installan tambahan yang saya lakukan seperti pada gambar 2.1 yaitu untuk menambahkan modul sumber selain react native biasa dan pada gambar 2.2 yaitu untuk peer dependency pada library sumber yang diinstall sebelumnya.

```
C:\Users\satri\ProjectSaya>yarn add @ui-kitten/components @eva-design/eva lodash.filter
yarn add v1.22.10
info No lockfile found.
```

Gambar 2.1 Menambah Library Sumber

```
C:\Users\satri\ProjectSaya>npm install react-native-svg
npm WARN deprecated uglify-es@3.3.9: support for ECMAScript is superseded by `uglify-js` as of v3.13.0

added 374 packages, removed 252 packages, changed 55 packages, and audited 1199 packages in 18s

73 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Gambar 2.2 Membuat Peer Dependency untuk Library

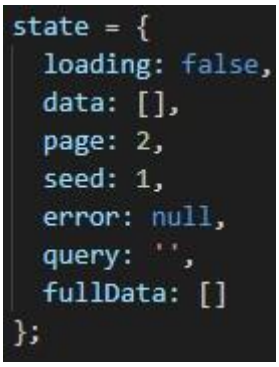
```
import React from 'react';
import {
  FlatList,
  View,
  ActivityIndicator,
  TouchableOpacity,
} from 'react-native';
import filter from 'lodash.filter';
import {
  ApplicationProvider,
  Text,
  Avatar,
  Input,
} from '@ui-kitten/components';
import { mapping, light as lightTheme } from '@eva-design/eva';
```

Gambar 2.3 Import Section

Pada bagian import adalah untuk mengambil beberapa kelas yang diperlukan dari library yang ada. Ada 4 library yang digunakan disini seperti react-native, lodash.filter, @ui-kitten/components dan @eva-design/eva.

Gambar 2.4 Pendeklarasian Class Utama

Pendeklarasian class Homescreen sebagai kelas utama yang mengextend React.Component. Pada class ini akan menampung banyak fungsi yang digunakan pada program ini.



Gambar 2.5 Pendefinisian Beberapa Nilai State

Pada kelas utama tadi, didefinisikan beberapa nilai awal state yang akan digunakan. Ada state yang berfungsi sebagai variabel, ada juga yang menjadi array untuk menampung data ataupun menampung perintah.

```
componentDidMount() {
  this.makeRemoteRequest();
}

makeRemoteRequest = () => {
  const { page, seed } = this.state;
  const url = `https://randomuser.me/api/?seed=${seed}&page=${page}&results=25`;
  this.setState({ loading: true });

  fetch(url)
    .then((res) => res.json())
    .then((res) => {
      this.setState({
        data: page === 1 ? res.results : [...this.state.data, ...res.results],
        error: res.error || null,
        loading: false,
        fullData: res.results,
      });
    })
    .catch((error) => {
      this.setState({ error, loading: false });
    });
};
```

Gambar 2.6 Fungsi untuk Mengambil Data Sesuai State

Pada fungsi componentDidMount diatas adalah memanggil atau menyimpan nilai dari fungsi makeRemoteRequest. Sedangkan, fungsi makeRemoteRequest disini mengambil state page dan seed dan memanggil link untuk data random yang akan digunakan dihubungkan dengan page dan seed.

Untuk fetch, adalah dari link yang ada tadi diambil datanya untuk disimpan pada state data sesuai dengan nilai dari state page.

```
contains = ({ name, email }, query) => {
  const { first, last } = name;
  if (first.includes(query) ||
      last.includes(query) ||
      email.includes(query)){
    return true;
  }
  return false;
};

handleSearch = (text) => {
  const formattedQuery = text.toLowerCase();
  const data = filter(this.state.fullData, (user) => {
    return this.contains(user, formattedQuery);
  });
  this.setState({ data, query: text });
};
```

Gambar 2.7 Fungsi Pencarian

Pada contains, adalah untuk kata kunci pencarian dan query yang digunakannya. Disini menggunakan name dan email sebagai acuannya. Sedangkan untuk handleSearch adalah untuk menampilkan hasil sementara dari pencarian dengan konstanta yang ada dengan formattedquery mengubah kata kunci menjadi lowercase jika yang diinputkan adalah uppercase dan data yang menyimpan hasil filter data pencarian sementara.

```
renderHeader = () => (
  <View
    style={{
      backgroundColor: '#fff',
      padding: 10,
      alignItems: 'center',
      justifyContent: 'center',
    }}>
    <Input
      autoCapitalize="none"
      autoCorrect={false}
      onChangeText={this.handleSearch}
      status="info"
      placeholder="Search"
      style={{
        borderRadius: 25,
        borderColor: '#333',
        backgroundColor: '#fff',
      }}
      textStyle={{ color: '#000' }}
      clearButtonMode="always"
    />
  </View>
);
```

Gambar 2.8 Fungsi untuk Mengatur Header Program

Pada renderHeader adalah untuk mengatur header program seperti mengatur tampilannya. Selain itu juga mengatur inputannya juga karena pada header terdapat search bar. Seperti pengaturan font, huruf bayang pada kotak pencarian, memanggil fungsi handleSearch untuk mengisi pencarian sementara, dll.

```
renderSeparator = () => {
  return (
    <View
      style={{
        height: 1,
        width: '100%',
        backgroundColor: '#CED0CE',
      }}
    />
  );
};

renderFooter = () => {
  if (!this.state.loading) return null;
  return (
    <View
      style={{
        paddingVertical: 20,
        borderTopWidth: 1,
        borderColor: '#CED0CE',
      }}>
      <ActivityIndicator animating size="large" />
    </View>
  );
};
```

Gambar 2.9 Fungsi untuk Mengatur Footer Program dan Sekat Data

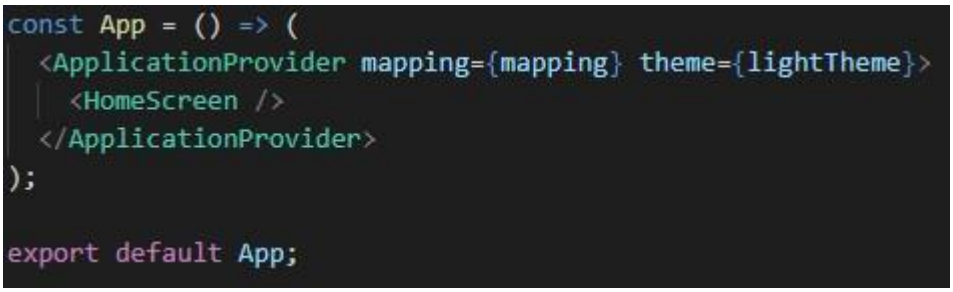
Pada renderSeparator untuk membuat tampilan sekat untuk setiap data yang ada. Sedangkan pada renderFooter adalah untuk membuat tampilan footer data dan mengatur animasinya dengan class ActivityIndicator

```
render() {
  return (
    <View
      style={{
        flex: 1,
        paddingHorizontal: 20,
        paddingVertical: 20,
        marginTop: 40,
      }}>
      <FlatList
        data={this.state.data}
        renderItem={({ item }) => (
          <TouchableOpacity onPress={() => alert('Item pressed!')}>
            <View
              style={{
                flexDirection: 'row',
                padding: 16,
                alignItems: 'center',
              }}>
              <Avatar
                source={{ uri: item.picture.thumbnail }}
                size="giant"
                style={{ marginRight: 16 }}
              />
```

Gambar 2.10 Fungsi untuk Mengatur Data pada Tampilan

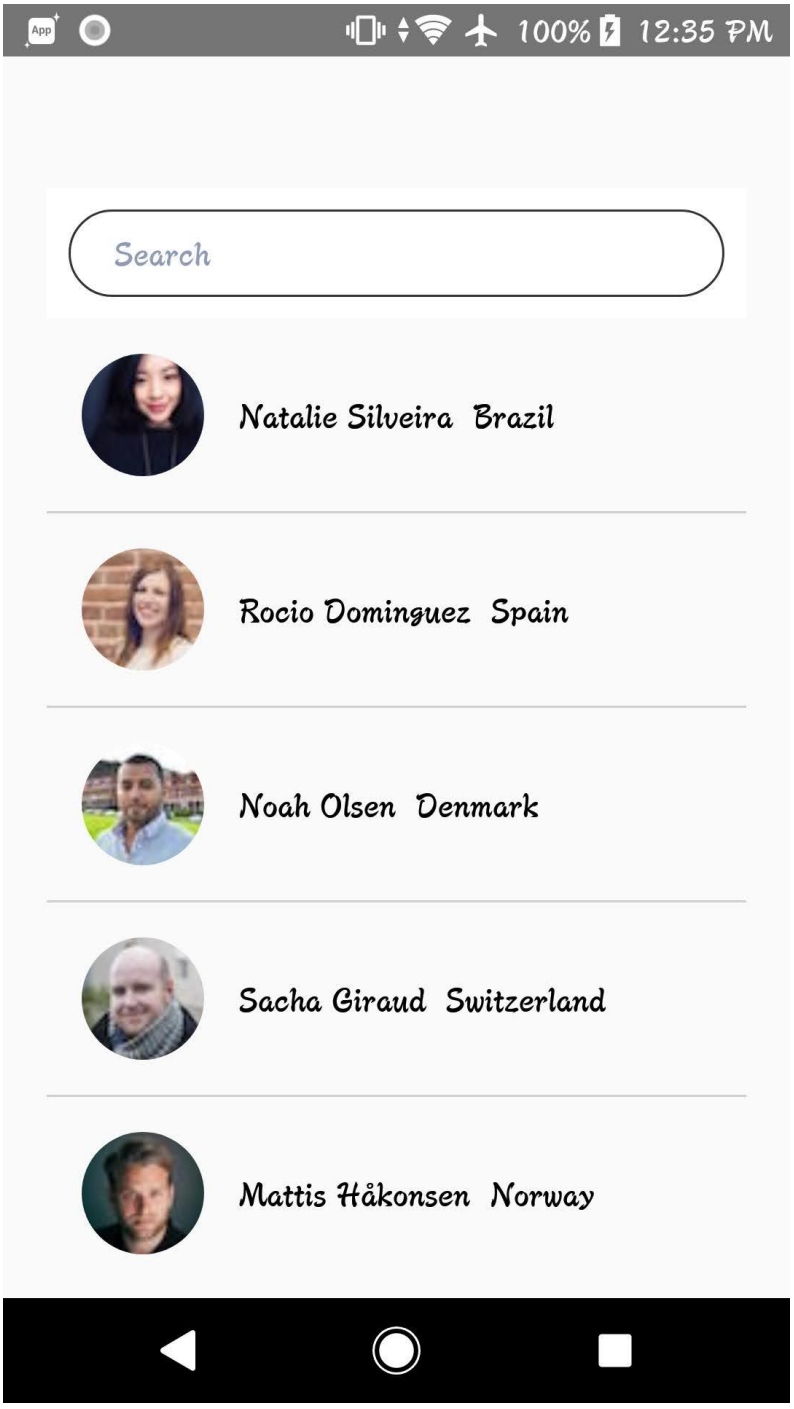
Gambar 2.11 Fungsi untuk Mengatur Data pada Tampilan (Lanjutan)

Pada render, ini adalah fungsi terakhir yang ada pada class Homescreen ini. Fungsi utama untuk menampilkan data, profile dari data yang ditampilkan, memberi animasi jika data ditekan, mengatur penempatan teks, serta memanggil keyExtractor untuk memanggil data dengan acuan yang digunakan dan juga menampung fungsi renderSeparator, renderHeader dan renderFooter.



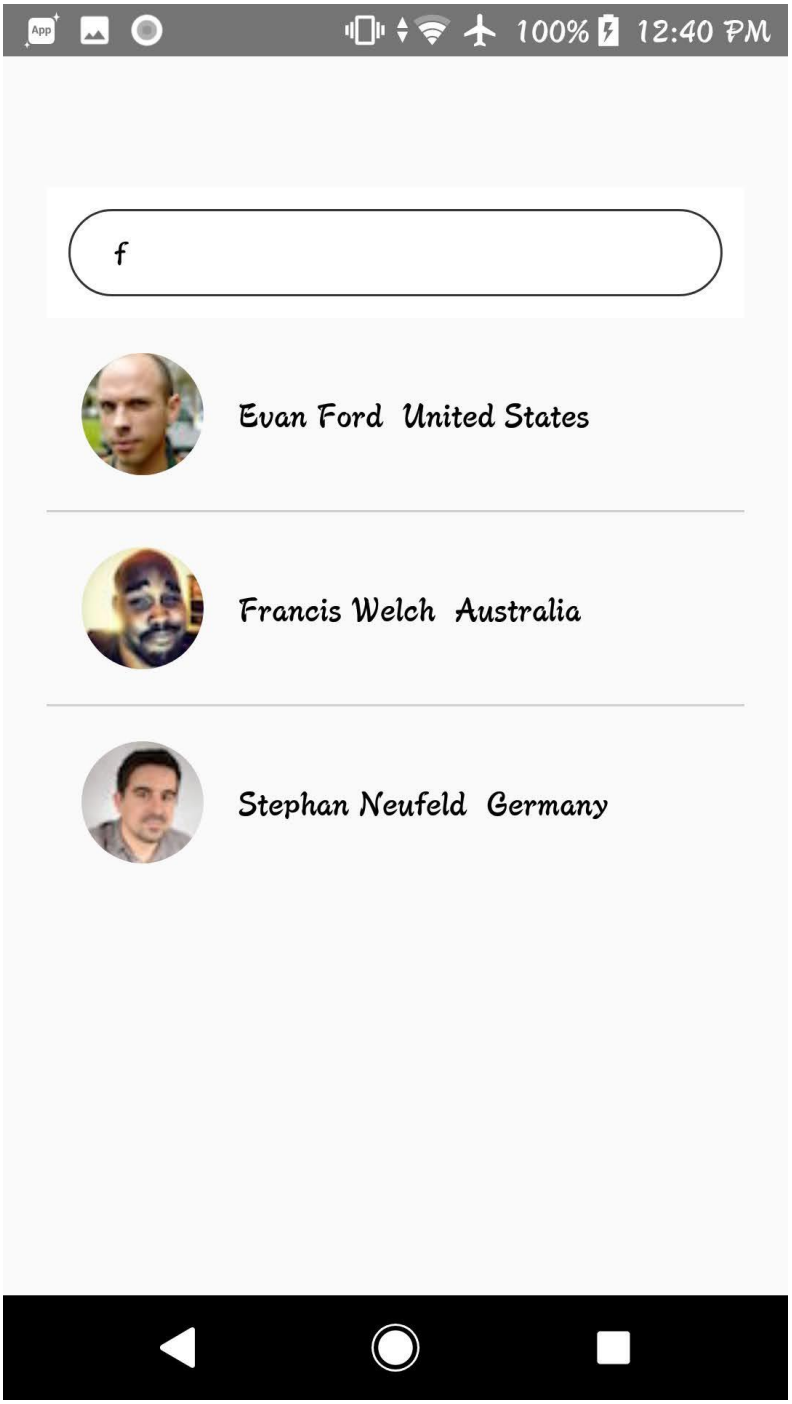
Gambar 2.12 Fungsi Utama pada Program

Pada fungsi utama ini, mengatur penempatan/mapping, tema dari aplikasi dengan class ApplicationProvider serta memanggil kelas utama tadi yaitu HomeScreen. Serta akhirnya mengexport App ini.



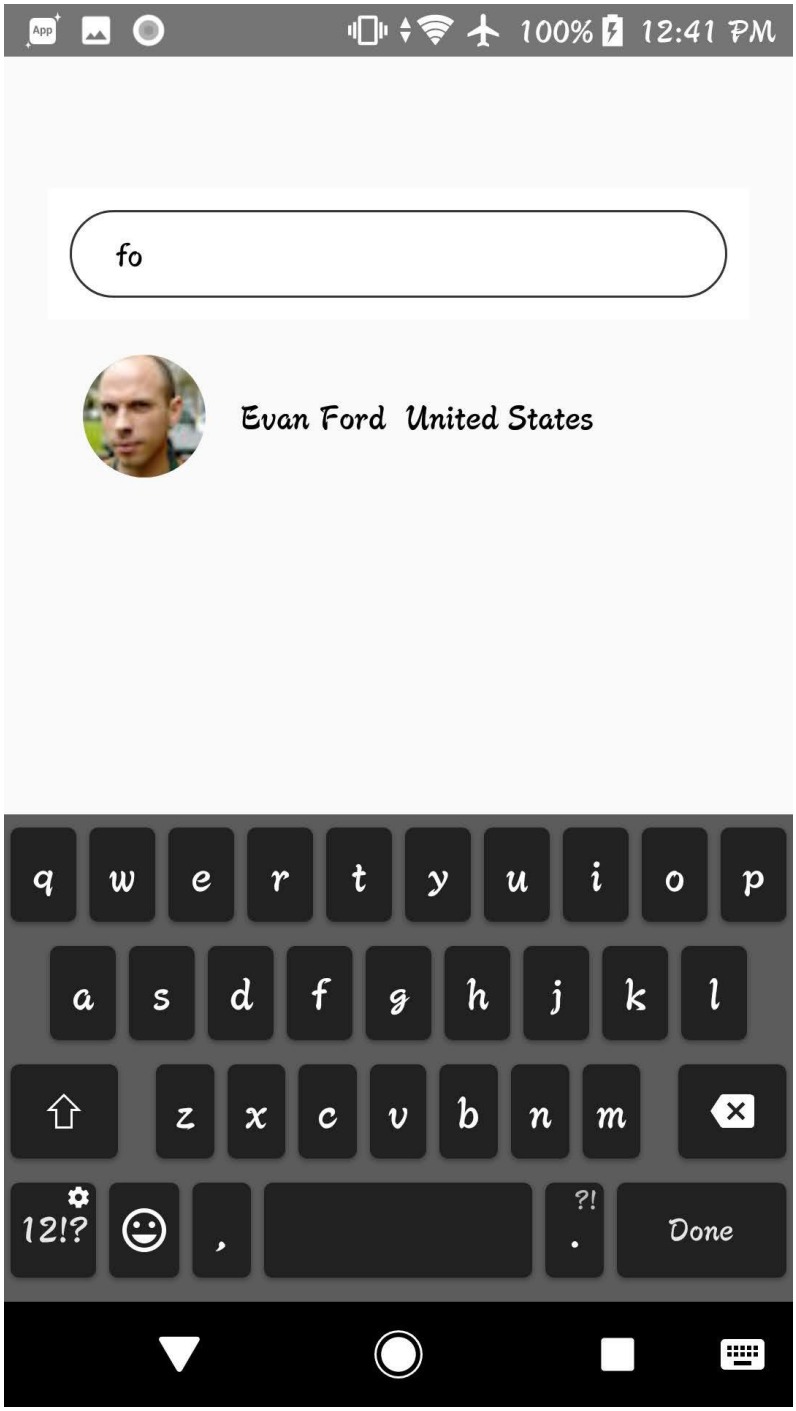
Gambar 2.13 Flatlist

Pada gambar diatas, ditampilkan program yang berisi flatlist dengan data beberapa orang serta negaranya. Tentu saja dengan menggunakan searchBar diatasnya.



Gambar 2.14 Pencarian pada Flatlist dengan 1 Huruf

Lalu, pada kotak pencarian ketika ada huruf ditekan dan disini dicontohkan dengan f. Maka Orang yang memiliki huruf f pada namanya akan dimunculkan sebagai hasil sementara.



Gambar 2.15 Pencarian pada Flatlist dengan Huruf Lanjutan

Sedangkan, jika kita melanjutkan kata kunci pencarian. Maka filtering akan semakin sempit sesuai dengan inputan yang kita berikan pada kotak pencarian.

BAB III
KESIMPULAN

Pencarian pada flatlist adalah sebuah fungsi yang sangat diperlukan mengingat flatlist dapat berisi banyak data. Juga, hal ini memudahkan kita untuk filtering ataupun sorting nama orang berdasarkan huruf. Selain itu, pembuatan flatlist juga perlu diperhatikan karena memerlukan beberapa komponen yang fatal jika dilupakan.

DAFTAR PUSTAKA

FlatList · React Native. (n.d.). Retrieved May 16, 2021, from https://reactnative.dev/docs/flatlist

How to Add a Search Bar in a FlatList in React Native Apps. (n.d.). Retrieved May 16, 2021, from https://blog.crowdbotics.com/add-search-bar-flatlist-react-native-apps/

Searching using Search Bar Filter in React Native List View - About React. (n.d.). Retrieved May 16, 2021, from https://aboutreact.com/react-native-search-bar-filter-on-listview/

LAMPIRAN

```
C:\Users\satri\ProjectSaya>yarn add @ui-kitten/components @eva-design/eva lodash.filter
yarn add v1.22.10
info No lockfile found.
```

Gambar 2.1 Menambah Library Sumber

```
C:\Users\satri\ProjectSaya>npm install react-native-svg
npm WARN deprecated uglify-es@3.3.9: support for ECMAScript is superseded by `uglify-js` as of v3.13.0

added 374 packages, removed 252 packages, changed 55 packages, and audited 1199 packages in 18s

73 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

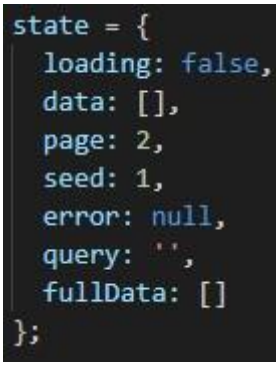
Gambar 2.2 Membuat Peer Dependency untuk Library

```
import React from 'react';
import {
  FlatList,
  View,
  ActivityIndicator,
  TouchableOpacity,
} from 'react-native';
import filter from 'lodash.filter';
import {
  ApplicationProvider,
  Text,
  Avatar,
  Input,
} from '@ui-kitten/components';
import { mapping, light as lightTheme } from '@eva-design/eva';
```

Gambar 2.3 Import Section

```
class HomeScreen extends React.Component {
```

Gambar 2.4 Pendeklarasian Class Utama



Gambar 2.5 Pendefinisian Beberapa Nilai State

```
componentDidMount() {
  this.makeRemoteRequest();
}

makeRemoteRequest = () => {
  const { page, seed } = this.state;
  const url = `https://randomuser.me/api/?seed=${seed}&page=${page}&results=25`;
  this.setState({ loading: true });

  fetch(url)
    .then((res) => res.json())
    .then((res) => {
      this.setState({
        data: page === 1 ? res.results : [...this.state.data, ...res.results],
        error: res.error || null,
        loading: false,
        fullData: res.results,
      });
    })
    .catch((error) => {
      this.setState({ error, loading: false });
    });
};
```

Gambar 2.6 Fungsi untuk Mengambil Data Sesuai State

```
contains = ({ name, email }, query) => {
  const { first, last } = name;
  if (first.includes(query) ||
      last.includes(query) ||
      email.includes(query)){
    return true;
  }
  return false;
};

handleSearch = (text) => {
  const formattedQuery = text.toLowerCase();
  const data = filter(this.state.fullData, (user) => {
    return this.contains(user, formattedQuery);
  });
  this.setState({ data, query: text });
};
```

Gambar 2.7 Fungsi Pencarian

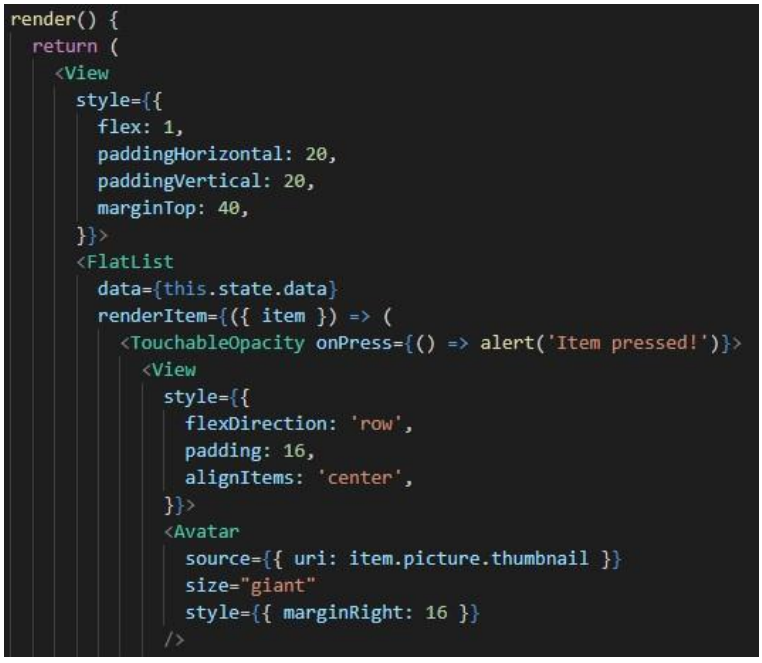
```
renderHeader = () => (
  <View
    style={{
      backgroundColor: '#fff',
      padding: 10,
      alignItems: 'center',
      justifyContent: 'center',
    }}>
    <Input
      autoCapitalize="none"
      autoCorrect={false}
      onChangeText={this.handleSearch}
      status="info"
      placeholder="Search"
      style={{
        borderRadius: 25,
        borderColor: '#333',
        backgroundColor: '#fff',
      }}
      textStyle={{ color: '#000' }}
      clearButtonMode="always"
    />
  </View>
);
```

Gambar 2.8 Fungsi untuk Mengatur Header Program

```
renderSeparator = () => {
  return (
    <View
      style={{
        height: 1,
        width: '100%',
        backgroundColor: '#CED0CE',
      }}
    />
  );
};

renderFooter = () => {
  if (!this.state.loading) return null;
  return (
    <View
      style={{
        paddingVertical: 20,
        borderTopWidth: 1,
        borderColor: '#CED0CE',
      }}>
      <ActivityIndicator animating size="large" />
    </View>
  );
};
```

Gambar 2.9 Fungsi untuk Mengatur Footer Program dan Sekat Data



```
render() {
  return (
    <View
      style={{
        flex: 1,
        paddingHorizontal: 20,
        paddingVertical: 20,
        marginTop: 40,
      }}>
      <FlatList
        data={this.state.data}
        renderItem={({ item }) => (
          <TouchableOpacity onPress={() => alert('Item pressed!')}>
            <View
              style={{
                flexDirection: 'row',
                padding: 16,
                alignItems: 'center',
              }}>
              <Avatar
                source={{ uri: item.picture.thumbnail }}
                size="giant"
                style={{ marginRight: 16 }}
              />
```

Gambar 2.10 Fungsi untuk Mengatur Data pada Tampilan

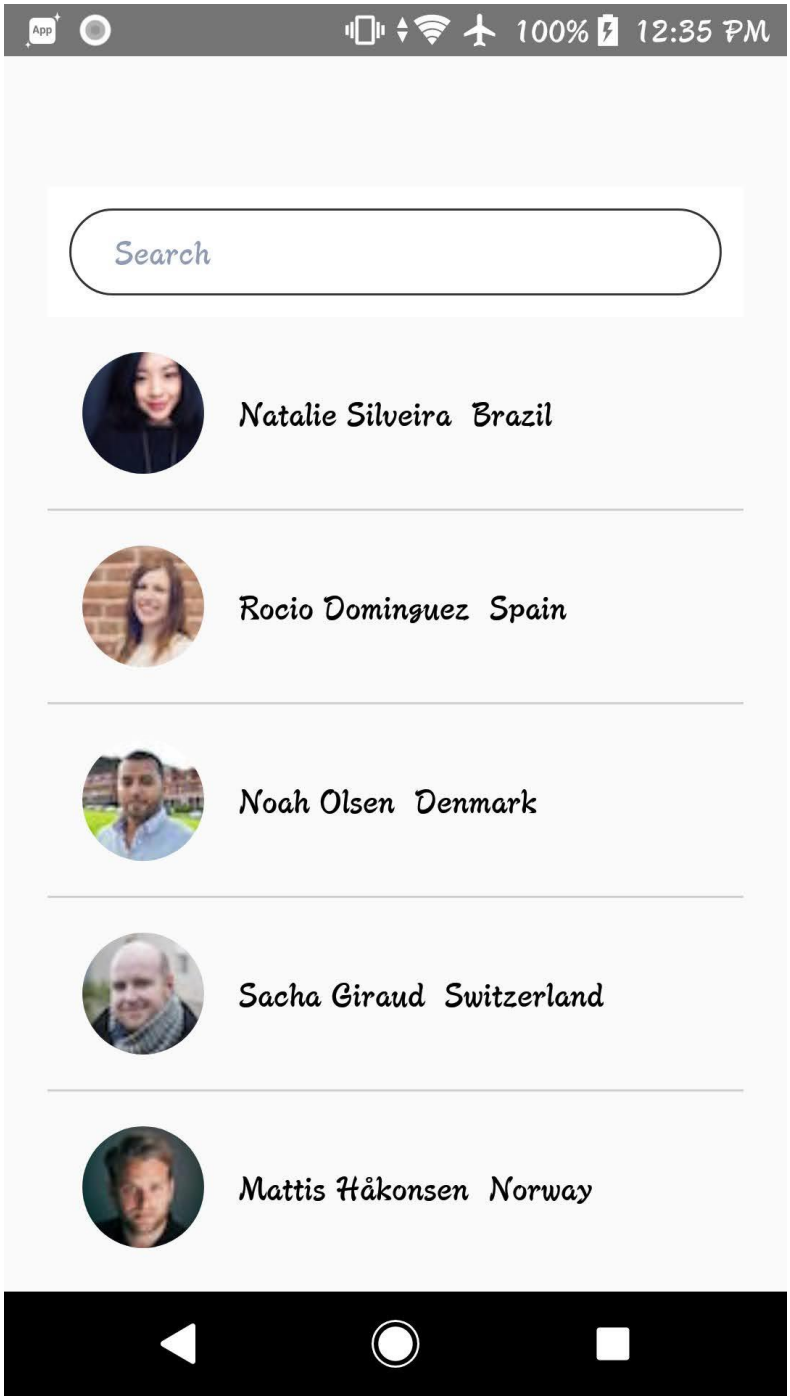
```
                <Text
                  category='s1'
                  style={{
                    color: '#000',
                  }}>{`${item.name.first} ${item.name.last}  `}</Text>
                <Text
                  category="p"
                  style={{
                    color: '#000',
                  }}>{`${item.location.country}`}</Text>
              </View>
            </TouchableOpacity>
          )}
          keyExtractor={(item) => item.email}
          ItemSeparatorComponent={this.renderSeparator}
          ListHeaderComponent={this.renderHeader}
          ListFooterComponent={this.renderFooter}
        />
      </View>
    );
  }
}
```

Gambar 2.11 Fungsi untuk Mengatur Data pada Tampilan (Lanjutan)

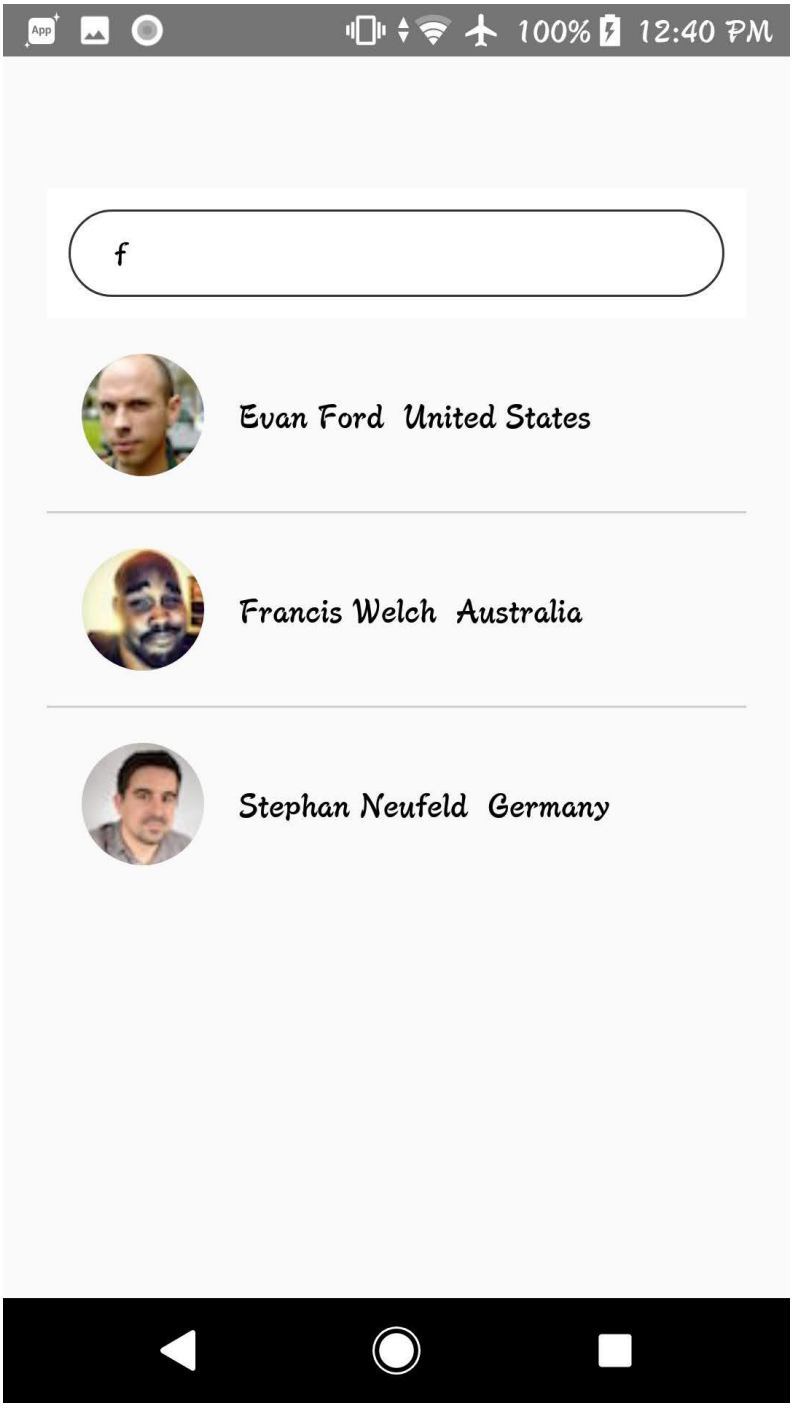
```
const App = () => (
  <ApplicationProvider mapping={mapping} theme={lightTheme}>
    <HomeScreen />
  </ApplicationProvider>
);

export default App;
```

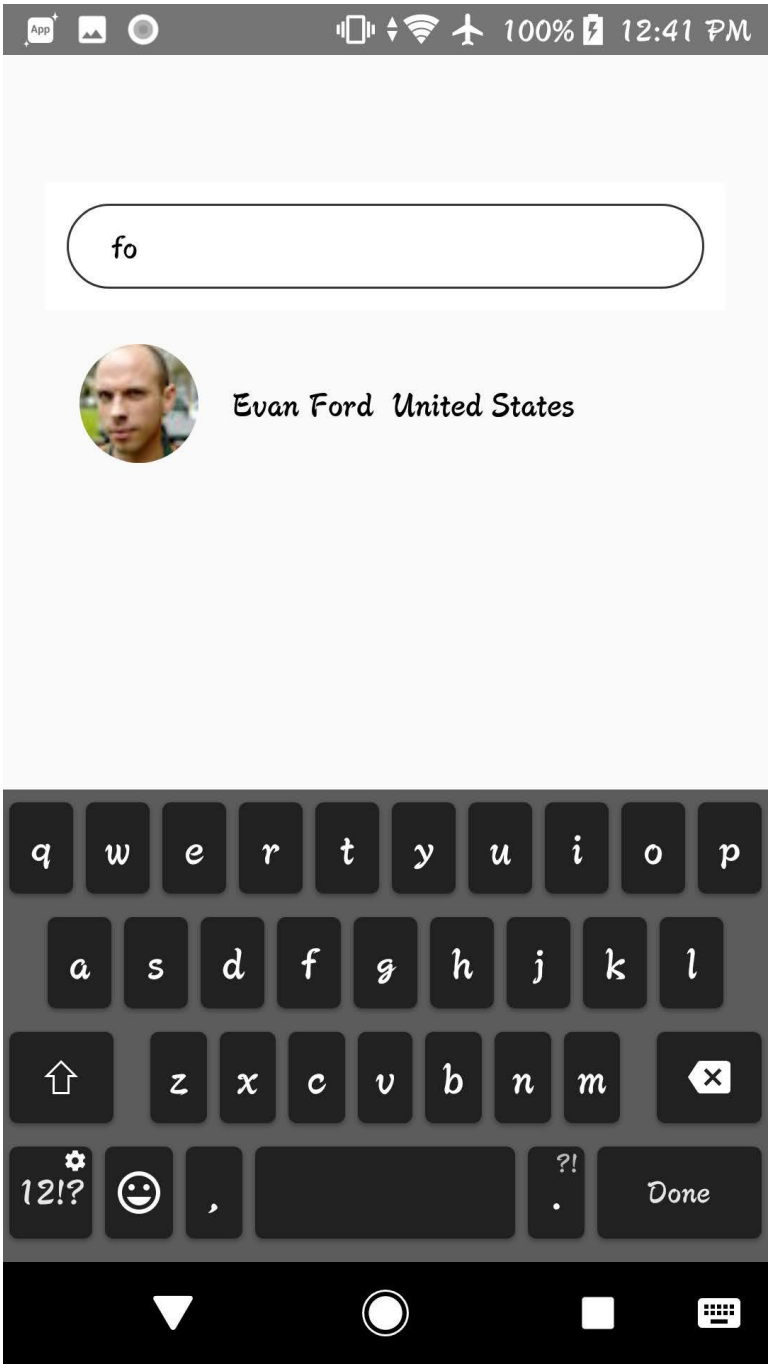
Gambar 2.12 Fungsi Utama pada Program



Gambar 2.13 Flatlist



Gambar 2.14 Pencarian pada Flatlist dengan 1 Huruf



Gambar 2.15 Pencarian pada Flatlist dengan Huruf Lanjutan