1:- data not here in array

 {this.state.arrpeopleFilter.length > 0 ? (

            <FlatList

              data={

                this.state.arrpeopleFilter.length > 0

                  ? this.state.arrpeopleFilter

                  : this.state.arrpeople

              }

              renderItem={renderItem}

              // keyExtractor={item => item.title}

              extraData={this.state.selectedId}

              style={{

                width: '100%',

                marginTop: 27,

                paddingHorizontal: 10,

                bottom: 26,

                height: '30%',

              }}

            />

          )

          : (

            <View style={{flex: 1, justifyContent:'center'}}>

              <Text>No Data Found</Text>

            </View>

          )

        }

2:- Lower case in search

 searchfilter = peopleString => {

    var filterArray = this.state.arrpeople.filter(function (item) {

      return item.name.toLowerCase().includes(peopleString.toLowerCase());

    });

    this.setState({

      arrpeopleFilter: filterArray,

    });

  };

3:-comple word type display output:-

Check it low case value it is same output display in here

 this.setState({

      arrpeopleFilter: this.state.arrpeople.filter(function (item) {

        return (

          item.otherInfo.gender.toLowerCase() == peopleString.toLowerCase()

        );

      }),

    });

4: Integer Length Check in

this.setState({

      arrpeopleFilter: this.state.arrpeople.filter(function (item) {

        return item.id > parseInt(peopleString);

      }),

    });

\*\* gender check

 // if filterArray.length > 0 {

    //   // do nothing

    // }

    // else {

    //   arrGender.push)(iatem.gender);

    // }

5 :json array in unqiue value indenfiere

constructor() {

super();

this.state = {

courses: [

{ campusname: "A", coursename: "X", id: 1 },

{ campusname: "A", coursename: "Y", id: 2 },

{ campusname: "B", coursename: "Z", id: 3 }

]

};

this.\_getUniqueCampusName = this.\_getUniqueCampusName.bind(this);

}

\_getUniqueCampusName() {

const courses = this.state.courses;

const campusSelection = [...new Set(courses.map(item => item.campusname))];

//console.log(campusSelection);

return campusSelection.map((campus, index) => (

<option key={index}>{campus}</option>

));

}

https://stackoverflow.com/questions/53111298/distinct-value-in-json-data

dash circle in View

<View

style={{

flexDirection: 'column',

justifyContent: 'center',

flex: 1,

alignItems: 'center',

alignSelf: 'center',

alignContent: 'center',

bottom: '8%',

// height: '100%',

}}>

<View

style={{

height: 110,

width: 110,

marginTop: '10.0%',

right: '5.0%',

borderRadius: 110 / 2,

// borderWidth: 2,

// right: 5,

// borderColor: 'blue',

// borderStyle: 'dashed',

}}>

<Image

style={{

height: 70,

width: 70,

alignItems: 'center',

alignSelf: 'center',

justifyContent: 'center',

}}

source={item.image}></Image>

<Text

style={{

alignItems: 'center',

alignSelf: 'center',

color: 'black',

justifyContent: 'center',

}}>

{item.name}

</Text>

</View>

</View>

   options={{

            title: 'Welcome',

            headerTintColor: '#ffffff',

            headerTitleAlign: 'left',

            headerShown: true,

            headerTitleStyle: {colors: colors.Gray},

            headerStyle: {backgroundColor: colors.Cylindrical\_coordinate},

          }}

        />

   options={{

            headerTitleStyle: {

              fontSize: 16,

            },

            title: 'Expert Consultation',

            headerTintColor: colors.Black,

            headerTitleAlign: 'left',

            headerShown: true,

            headerStyle: {backgroundColor: colors.white},

          }}

        />

Echo > create a new file and rnc create a class

Current date in 7 days display in react native

<https://stackoverflow.com/questions/10032456/how-to-get-next-seven-days-from-x-and-format-in-js>

//date and value

import React, {Component} from 'react';

import {Text, View, FlatList, Image, TouchableOpacity} from 'react-native';

import {colors} from '../Untils/colors';

import Customschedual from './Customschedual/Customschedual';

import moment from 'moment';

let days = [];

let daysRequired = 2;

let i = 0;

for (i = i; i < daysRequired; i++) {

days.push(moment().add(i, 'days').format(' Do- MMMM- YYYY'));

}

let myScehduleAray = [

{

id: 1,

shot: '2 slots available',

},

{id: 2, shot: '4 slots available'},

{

id: 3,

shot: '6 slots available',

},

{

id: 4,

shot: '8 slots available',

},

{

id: 5,

shot: '10 slots available',

},

{

id: 6,

shot: '2 slots available',

},

{

id: 7,

shot: '4 slots available',

},

{

id: 8,

shot: '4 slots available',

},

{

id: 9,

shot: '2 slots available',

},

{id: 10, shot: '4 slots available'},

{

id: 11,

shot: '6 slots available',

},

{

id: 12,

shot: '8 slots available',

},

{

id: 13,

shot: '10 slots available',

},

{

id: 14,

shot: '2 slots available',

},

{

id: 15,

shot: '4 slots available',

},

{

id: 16,

shot: '4 slots available',

},

{

id: 17,

shot: '2 slots available',

},

{id: 18, shot: '4 slots available'},

{

id: 19,

shot: '6 slots available',

},

{

id: 20,

shot: '8 slots available',

},

{

id: 21,

shot: '10 slots available',

},

{

id: 22,

shot: '2 slots available',

},

{

id: 23,

shot: '4 slots available',

},

{

id: 24,

shot: '4 slots available',

},

{

id: 25,

shot: '2 slots available',

},

{id: 26, shot: '4 slots available'},

{

id: 27,

shot: '6 slots available',

},

{

id: 28,

shot: '8 slots available',

},

{

id: 29,

shot: '10 slots available',

},

{

id: 30,

shot: '2 slots available',

},

{

id: 31,

shot: '4 slots available',

},

{

id: 8,

shot: '4 slots available',

},

];

const rows = [

{

isExpanded: false,

url: require('../Image/sunrise.png'),

Slot: 'Morning',

timeinterval: '8:00Am to 12:00Pm',

url1: require('../Image/chevron.png'),

subSlot: [

{

id: 1,

Time: '8:00Am',

},

{

id: 2,

Time: '8:20Am',

},

{

id: 3,

Time: '8:40Am',

},

{

id: 4,

Time: '9:00Am',

},

{

id: 3,

Time: '8:40Am',

},

{

id: 4,

Time: '9:00Am',

},

{

id: 5,

Time: '9:20Am',

},

{

id: 6,

Time: '9:20Am',

},

],

},

{

isExpanded: false,

url: require('../Image/afternoon.png'),

Slot: 'Afternoon',

timeinterval: '12:00pm to 4:00pm',

url1: require('../Image/chevron.png'),

subSlot: [

{

id: 5,

Time: '12:00pm',

},

{

id: 6,

Time: '12:20pm',

},

{

id: 7,

Time: '12:40pm',

},

{

id: 8,

Time: '1:00pm',

},

],

},

{

isExpanded: false,

url: require('../Image/moon.png'),

Slot: 'Evening',

timeinterval: '4:00pm to 8:00pm',

url1: require('../Image/chevron.png'),

subSlot: [

{

id: 5,

Time: '4:00pm',

},

{

id: 6,

Time: '4:20pm',

},

{

id: 7,

Time: '4:40pm',

},

{

id: 8,

Time: '5:00pm',

},

{

id: 8,

Time: '5:20pm',

},

],

},

];

export default class Visitclinic extends Component {

constructor(props) {

super(props);

this.state = {

dropicon: false,

isexpanded: false,

isExpandIndex: 0,

data: rows,

};

}

titleview = () => {

return (

<View style={{padding: 15}}>

<Text

style={{

fontFamily: 'Prompt-SemiBoldItalic',

fontWeight: '700',

fontSize: 19,

color: colors.Black,

}}

onPress={() => {

alert(

'visitclinic' + JSON.stringify(this.state.schedualshot, null, 3),

);

}}>

Schedual

</Text>

</View>

);

};

render() {

// const renderItem = ({item}) => {

// return (

// <View

// style={{

// flex: 1,

// alignItems: 'center',

// marginVertical: 5,

// marginTop: 5,

// alignSelf: 'center',

// }}>

// <Customschedual item={item} days={days} />

// </View>

// );

// };

const morningview = ({item, index}) => {

let items = [];

const row = item.subSlot;

items = row.map((rowItem, Number) => {

return (

<View

style={{

flex: 1,

alignItems: 'center',

height: 30,

width: 50,

margin: 5,

backgroundColor: colors.white,

}}>

<Text

style={{

color: colors.Black,

alignSelf: 'center',

flex: 1,

textAlign: 'center',

justifyContent: 'center',

alignContent: 'center',

}}>

{rowItem.Time}

</Text>

</View>

);

});

return (

<TouchableOpacity

style={{flex: 1, justifyContent: 'center'}}

onPress={() => {

this.setState({

isExpandIndex: this.state.isExpandIndex == index ? -1 : index,

});

}}>

<View style={{marginTop: 20}}>

<View

style={{

alignItems: 'center',

flexDirection: 'row',

backgroundColor:

this.state.isExpandIndex == index

? colors.bluelight

: colors.Cultured,

padding: 5,

}}>

<Image

style={{

tintColor:

this.state.isExpandIndex == index

? colors.Cylindrical\_coordinate

: colors.Black,

}}

source={item.url}></Image>

<Text

style={{

marginLeft: 15,

color:

this.state.isExpandIndex == index

? colors.white

: colors.Black,

}}>

{item.Slot}

</Text>

<Text

style={{

marginLeft: 15,

color:

this.state.isExpandIndex == index

? colors.white

: colors.Black,

}}>

{item.timeinterval}

</Text>

<TouchableOpacity

style={{flex: 1, justifyContent: 'center'}}

onPress={() => {

this.setState({

isExpandIndex:

this.state.isExpandIndex == index ? -1 : index,

});

}}>

<View

style={{

flex: 1,

alignItems: 'center',

justifyContent: 'center',

}}>

<Image

style={{

alignSelf: 'flex-end',

tintColor:

this.state.isExpandIndex == index

? colors.white

: colors.Black,

}}

source={item.url1}></Image>

</View>

</TouchableOpacity>

</View>

{this.state.isExpandIndex == index ? (

<View

style={{

//he ight: this.state.isExpandIndex - index,

overflow: 'scroll',

flexDirection: 'column-reverse',

backgroundColor:

this.state.isExpandIndex == index

? colors.bluelight

: colors.Cultured,

}}>

{items}

</View>

) : null}

</View>

</TouchableOpacity>

);

};

return (

<View>

{this.titleview()}

<FlatList

horizontal={true}

data={days}

renderItem={({item, index}) => (

<View

style={{

flex: 1,

alignItems: 'center',

marginVertical: 5,

marginTop: 5,

alignSelf: 'center',

}}>

<Customschedual item={myScehduleAray[index]} days={days[index]} />

</View>

)}

keyExtractor={item => item.key}

contentContainerStyle={{margin: 5, textAlign: 'center'}}

ItemSeparatorComponent={this.renderSeparator}

/>

<View style={{padding: 15, marginTop: 10}}>

<Text

style={{

fontFamily: 'Prompt-SemiBoldItalic',

fontSize: 19,

fontWeight: '700',

color: colors.Cylindrical\_coordinate,

}}>

16 DEC

</Text>

</View>

<FlatList

data={this.state.data}

renderItem={morningview}

style={{marginHorizontal: 10}}></FlatList>

<View style={{}}>

<TouchableOpacity

onPress={() => {

// this.props.navigation.navigate('PatientDeaties');

}}>

<View

style={{

borderColor: colors.white,

flexDirection: 'row',

alignItems: 'center',

position: 'relative',

marginTop: 40,

bottom: 0,

paddingBottom: 80,

alignSelf: 'center',

display: 'flex',

}}>

<Text

style={{

letterSpacing: 1,

fontFamily: 'Prompt-SemiBoldItalic',

fontWeight: '800',

paddingRight: 2,

fontFamily: 'PromptMedium',

color: colors.Cylindrical\_coordinate,

}}>

Continue

</Text>

<Image

style={{

height: 40,

width: 19,

top: 1.0,

tintColor: colors.Cylindrical\_coordinate,

alignItems: 'flex-end',

alignSelf: 'flex-end',

resizeMode: 'center',

}}

source={

this.state.dropicon

? require('../Image/arrow-down-sign-to-navigate.png')

: require('../Image/rightarrow.png')

}></Image>

</View>

</TouchableOpacity>

</View>

</View>

);

}

}

Change icon in expandable in

<https://kps250.medium.com/creating-an-accordion-in-react-native-f313748b7b46>

import React, {Component} from 'react';

import {

  Text,

  View,

  FlatList,

  Image,

  ScrollView,

  TouchableOpacity,

  Dimensions,

} from 'react-native';

import {colors} from '../Untils/colors';

import Customschedual from './Customschedual/Customschedual';

import moment from 'moment';

let days = [];

let daysRequired = 5;

let i = 0;

for (i = i; i < daysRequired; i++) {

  days.push(moment().add(i, 'days').format('DD MMMM'));

}

const rows = [

  {

    isExpanded: false,

    url: require('../Image/sunrise.png'),

    Slot: 'Morning',

    timeinterval: '8:00Am to 12:00Pm',

    url1: require('../Image/chevron.png'),

    url2: require('../Image/arrowhead-up.png'),

    subSlot: [

      {

        id: 1,

        Time: '8:00Am',

      },

      {

        id: 2,

        Time: '8:20Am',

      },

      {

        id: 3,

        Time: '8:40Am',

      },

      {

        id: 4,

        Time: '9:00Am',

      },

      {

        id: 5,

        Time: '9:20Am',

      },

      {

        id: 6,

        Time: '9:40Am',

      },

    ],

  },

  {

    isExpanded: false,

    url: require('../Image/afternoon.png'),

    Slot: 'Afternoon',

    timeinterval: '12:00pm to 4:00pm',

    url1: require('../Image/chevron.png'),

    url2: require('../Image/arrowhead-up.png'),

    subSlot: [

      {

        id: 7,

        Time: '12:00pm',

      },

      {

        id: 8,

        Time: '12:20pm',

      },

      {

        id: 9,

        Time: '12:40pm',

      },

      {

        id: 10,

        Time: '1:00pm',

      },

      {

        id: 11,

        Time: '1:20pm',

      },

      {

        id: 12,

        Time: '1:40pm',

      },

    ],

  },

  {

    isExpanded: false,

    url: require('../Image/moon.png'),

    Slot: 'Evening',

    timeinterval: '4:00pm to 8:00pm',

    url1: require('../Image/chevron.png'),

    url2: require('../Image/arrowhead-up.png'),

    subSlot: [

      {

        id: 13,

        Time: '4:00pm',

      },

      {

        id: 14,

        Time: '4:20pm',

      },

      {

        id: 15,

        Time: '4:40pm',

      },

      {

        id: 16,

        Time: '5:00pm',

      },

      {

        id: 17,

        Time: '5:20pm',

      },

      {

        id: 18,

        Time: '5:40pm',

      },

      {

        id: 19,

        Time: '6:00pm',

      },

    ],

  },

];

export default class Visitclinic extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isSelectedVisit: false,

      schedualshot: [

        {

          id: 1,

          shot: '2 slots available',

        },

        {id: 2, shot: '4 slots available'},

        {

          id: 3,

          shot: '6 slots available',

        },

        {

          id: 4,

          shot: '8 slots available',

        },

        {

          id: 5,

          shot: '10 slots available',

        },

        {

          id: 6,

          shot: '2 slots available',

        },

        {

          id: 7,

          shot: '4 slots available',

        },

        {

          id: 8,

          shot: '4 slots available',

        },

        {

          id: 9,

          shot: '2 slots available',

        },

        {id: 10, shot: '4 slots available'},

        {

          id: 11,

          shot: '6 slots available',

        },

        {

          id: 12,

          shot: '8 slots available',

        },

        {

          id: 13,

          shot: '10 slots available',

        },

        {

          id: 14,

          shot: '2 slots available',

        },

        {

          id: 15,

          shot: '4 slots available',

        },

        {

          id: 16,

          shot: '4 slots available',

        },

        {

          id: 17,

          shot: '2 slots available',

        },

        {id: 18, shot: '4 slots available'},

        {

          id: 19,

          shot: '6 slots available',

        },

        {

          id: 20,

          shot: '8 slots available',

        },

        {

          id: 21,

          shot: '10 slots available',

        },

        {

          id: 22,

          shot: '2 slots available',

        },

        {

          id: 23,

          shot: '4 slots available',

        },

        {

          id: 24,

          shot: '4 slots available',

        },

        {

          id: 25,

          shot: '2 slots available',

        },

        {id: 26, shot: '4 slots available'},

        {

          id: 27,

          shot: '6 slots available',

        },

        {

          id: 28,

          shot: '8 slots available',

        },

        {

          id: 29,

          shot: '10 slots available',

        },

        {

          id: 30,

          shot: '2 slots available',

        },

        {

          id: 31,

          shot: '4 slots available',

        },

        {

          id: 8,

          shot: '4 slots available',

        },

      ],

      selectedId: null,

      schedualshotselectedId: null,

      isExpandIndex: 0,

      data: rows,

    };

  }

  titleview = () => {

    return (

      <View style={{padding: 10}}>

        <Text

          style={{

            fontFamily: 'Prompt-SemiBoldItalic',

            fontWeight: '700',

            fontSize: 19,

            color: colors.Black,

          }}>

          Schedule Appointment

        </Text>

      </View>

    );

  };

  render() {

    const renderItemShotdate = ({item, index}) => {

      const backgroundColor =

        this.state.schedualshot[index].id === this.state.schedualshotselectedId

          ? colors.Cylindrical\_coordinate

          : colors.Cultured;

      const color =

        this.state.schedualshot[index].id === this.state.schedualshotselectedId

          ? colors.white

          : colors.Cylindrical\_coordinate;

      return (

        <View

          style={{

            alignItems: 'center',

            bottom: 2,

            // backgroundColor: item.id ? 'red' : ' blue',

            alignSelf: 'center',

          }}>

          <Customschedual

            //old code  idnex

            item={this.state.schedualshot[index]}

            // item={item}

            days={days[index]}

            onPress={() => {

              this.setState({

                schedualshotselectedId: this.state.schedualshot[index].id,

              });

            }}

            backgroundColor={backgroundColor}

            color={color}

          />

        </View>

      );

    };

    // const renderItem = ({item}) => {

    //   return (

    //     <View

    //       style={{

    //         flex: 1,

    //         alignItems: 'center',

    //         marginVertical: 5,

    //         marginTop: 5,

    //         alignSelf: 'center',

    //       }}>

    //       <Customschedual item={item} days={days} />

    //     </View>

    //   );

    // };

    const renderItemDays = ({item, index, id}) => {

      let items = [];

      const row = item.subSlot;

      items = row.map(rowItem => {

        return (

          <TouchableOpacity

            onPress={() => {

              // alert('data' + rowItem.id);

              this.setState({selectedId: rowItem.id});

            }}>

            <View

              style={{

                alignSelf: 'center',

                width: 40,

                width: Dimensions.get('screen').width / 5,

                margin: 6,

                padding: 5,

                marginTop: 10,

                bottom: 5,

                backgroundColor:

                  rowItem.id == this.state.selectedId

                    ? colors.Cylindrical\_coordinate

                    : colors.white,

                justifyContent: 'center',

                alignSelf: 'center',

                alignItems: 'center',

                borderRadius: 3,

              }}>

              <Text

                style={{

                  fontSize: 12,

                  // flexShrink: 1,

                  color:

                    rowItem.id === this.state.selectedId

                      ? colors.white

                      : colors.Black,

                  // padding: 10,

                  // height: 30,

                  textAlign: 'center',

                }}>

                {rowItem.Time}

              </Text>

            </View>

          </TouchableOpacity>

        );

      });

      return (

        // <TouchableOpacity

        //   style={{flex: 1, justifyContent: 'center'}}

        //   onPress={() => {

        //     this.setState({

        //       isExpandIndex: this.state.isExpandIndex === index ? items : index,

        //     });

        //   }}>

        <View style={{bottom: 7, padding: 5}}>

          <TouchableOpacity

            onPress={() => {

              this.setState({

                isExpandIndex:

                  this.state.isExpandIndex === index ? items : index,

              });

            }}>

            <View

              style={{

                alignItems: 'center',

                flexDirection: 'row',

                padding: 7,

                backgroundColor:

                  this.state.isExpandIndex == index

                    ? colors.bluelight

                    : colors.Cultured,

              }}>

              <Image

                style={{

                  resizeMode: 'contain',

                  aspectRatio: 1.4,

                  tintColor:

                    this.state.isExpandIndex == index

                      ? colors.Cylindrical\_coordinate

                      : colors.Black,

                }}

                source={item.url}></Image>

              <Text

                style={{

                  marginLeft: 15,

                  fontSize: 12,

                  fontWeight: 'bold',

                  alignItems: 'center',

                  alignSelf: 'center',

                  justifyContent: 'center',

                  color:

                    this.state.isExpandIndex == index

                      ? colors.white

                      : colors.Black,

                }}>

                {item.Slot}

              </Text>

              <Text

                style={{

                  marginLeft: 15,

                  fontSize: 10,

                  color:

                    this.state.isExpandIndex == index

                      ? colors.white

                      : colors.Black,

                }}>

                {item.timeinterval}

              </Text>

              <View

                style={{

                  flex: 1,

                  alignItems: 'center',

                  justifyContent: 'center',

                }}>

                <Image

                  style={{

                    alignSelf: 'flex-end',

                    tintColor:

                      this.state.isExpandIndex == index

                        ? colors.white

                        : colors.Black,

                  }}

                  source={

                    this.state.isExpandIndex === index ? item.url2 : item.url1

                  }></Image>

              </View>

            </View>

          </TouchableOpacity>

          {this.state.isExpandIndex == index ? (

            <View

              style={{

                //he ight: this.state.isExpandIndex - index,

                overflow: 'scroll',

                flexDirection: 'column',

                backgroundColor:

                  this.state.isExpandIndex == index

                    ? colors.bluelight

                    : colors.Cultured,

              }}>

              <View

                style={{

                  flexDirection: 'row',

                  // alignItems: 'center',

                  // alignSelf: 'center',

                  justifyContent: 'space-between',

                  flexWrap: 'wrap',

                  backgroundColor: colors.bluelight,

                }}>

                {items.isExpandIndex ? null : items}

              </View>

            </View>

          ) : null}

        </View>

      );

    };

    return (

      <View>

        <ScrollView>

          <View>

            {this.titleview()}

            <FlatList

              horizontal={true}

              data={days}

              renderItem={renderItemShotdate}

              keyExtractor={item => item.key}

              style={{marginHorizontal: 5}}

              // ItemSeparatorComponent={this.renderSeparator}

            />

            <Text

              style={{

                fontFamily: 'Prompt-SemiBoldItalic',

                fontSize: 19,

                padding: 15,

                fontWeight: '700',

                color: colors.Cylindrical\_coordinate,

              }}>

              {days[0]}

            </Text>

            <FlatList

              data={this.state.data}

              renderItem={renderItemDays}

              style={{marginHorizontal: 10}}></FlatList>

            <View style={{}}>

              <TouchableOpacity

                onPress={() => {

                  // this.props.navigation.navigate('PatientDeaties');

                }}>

                <View

                  style={{

                    borderColor: colors.white,

                    flexDirection: 'row',

                    alignItems: 'center',

                    position: 'relative',

                    marginTop: 40,

                    bottom: 0,

                    paddingBottom: 80,

                    alignSelf: 'center',

                    display: 'flex',

                  }}>

                  <Text

                    style={{

                      letterSpacing: 1,

                      fontFamily: 'Prompt-SemiBoldItalic',

                      fontWeight: '800',

                      paddingRight: 2,

                      fontFamily: 'PromptMedium',

                      color: colors.Cylindrical\_coordinate,

                    }}>

                    Continue

                  </Text>

                  <Image

                    style={{

                      height: 40,

                      width: 19,

                      top: 1.0,

                      tintColor: colors.Cylindrical\_coordinate,

                      alignItems: 'flex-end',

                      alignSelf: 'flex-end',

                      resizeMode: 'center',

                    }}

                    source={

                      this.state.dropicon

                        ? require('../Image/arrow-down-sign-to-navigate.png')

                        : require('../Image/rightarrow.png')

                    }></Image>

                </View>

              </TouchableOpacity>

            </View>

          </View>

        </ScrollView>

      </View>

    );

  }

}

Custom validation in React native link

:

<https://www.skptricks.com/2018/10/react-native-check-text-input-is-empty-or-not.html>

free api tools

send data

import {Text, View, FlatList} from 'react-native';

import React, {Component} from 'react';

import Fetchapicard from './Fetchapi/Fetchapicard';

import {roundToNearestPixel} from 'react-native/Libraries/Utilities/PixelRatio';

export default class fetchapicalling extends Component {

  constructor(pros) {

    super(pros);

    this.state = {

      datalist: [],

      country: '',

    };

  }

  datafetch = () => {

    // fetch('https://jsonplaceholder.typicode.com/photos', {

    // //   method: 'POST',

    // //   headers: {

    // //     Accept: 'application/json',

    // //     'Content-Type': 'application/json',

    // //   },

    // //   body: JSON.stringify(),

    // // })

    //   .then(response => response.json())

    //   .then(json => {

    //     this.setState({datalist: json});

    //     alert('JSON data :-' + JSON.stringify(json, null, 2));

    //   });

    fetch('https://api.instantwebtools.net/v1/airlines')

      .then(response => response.json())

      .then(

        responsejson => this.setState({datalist: responsejson}),

        // console.log('data' + JSON.stringify(responsejson, null, 2)),

      );

  };

  componentDidMount() {

    this.datafetch();

  }

  render() {

    const renderItem = ({item}) => {

      return (

        <View>

          <Fetchapicard

            item={item}

            onPress={() => {

              this.props.navigation.navigate('Displaydata', {

                text: item.country,

                // item: this.state.country,

              });

              console.log('list' + item);

              console.log('airline line  id :-  ' + item.id);

            }}

          />

        </View>

      );

    };

    return (

      <View>

        <FlatList data={this.state.datalist} renderItem={renderItem} />

        <Text>fetchapicalling</Text>

      </View>

    );

  }

}

//get data

import {Text, View} from 'react-native';

import React, {Component} from 'react';

export default class Displaydata extends Component {

  constructor(pros) {

    super(pros);

    // console.log(

    //   'ths.props = ' + JSON.stringify(this.props.state.country, null),

    // );

    console.log('\n\n\n\nthis.props = ' + JSON.stringify(this.props, null, 2));

  }

  render() {

    return (

      <View>

        <Text onPress={() => alert(this.props.route.name)}>

          {this.props.route.params.text}

        </Text>

      </View>

    );

  }

}

// custom navigation menu code

// import React from 'react';

// import {

//   StyleSheet,

//   View,

//   Text,

//   Image,

//   TouchableOpacity,

//   SafeAreaView,

// } from 'react-native';

// import 'react-native-gesture-handler';

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

// import {createStackNavigator} from '@react-navigation/stack';

// import {

//   createDrawerNavigator,

//   DrawerContentScrollView,

//   DrawerItem,

// } from '@react-navigation/drawer';

// const HamburgerIcon = props => {

//   const toggleDrawer = () => {

//     props.navigationProps.toggleDrawer();

//   };

//   return (

//     <View style={{flexDirection: 'row'}}>

//       <TouchableOpacity onPress={toggleDrawer}>

//         <Image

//           source={{

//             uri: 'https://reactnativecode.com/wp-content/uploads/2018/04/hamburger\_icon.png',

//           }}

//           style={{width: 25, height: 25, marginLeft: 5}}

//         />

//       </TouchableOpacity>

//     </View>

//   );

// };

// // const CustomSidebar = props => {

// //   const {state, descriptors, navigation} = props;

// //   let lastGroupName = '';

// //   let newGroup = true;

// //   return (

// //     <SafeAreaView style={{flex: 1}}>

// //       <DrawerContentScrollView {...props}>

// //         {state.routes.map(route => {

// //           const {drawerLabel, activeTintColor, groupName} =

// //             descriptors[route.key].options;

// //           if (lastGroupName !== groupName) {

// //             newGroup = true;

// //             lastGroupName = groupName;

// //           } else newGroup = false;

// //           return (

// //             <View>

// //               {newGroup ? (

// //                 <View style={styles.sectionView}>

// //                   <Text key={groupName} style={{marginLeft: 10}}>

// //                     {groupName}

// //                   </Text>

// //                   <View style={styles.separatorLine} />

// //                 </View>

// //               ) : null}

// //               <DrawerItem

// //                 key={route.key}

// //                 label={({color}) => <Text style={{color}}>{drawerLabel}</Text>}

// //                 focused={

// //                   state.routes.findIndex(e => e.name === route.name) ===

// //                   state.index

// //                 }

// //                 activeTintColor={activeTintColor}

// //                 onPress={() => navigation.navigate(route.name)}

// //               />

// //             </View>

// //           );

// //         })}

// //       </DrawerContentScrollView>

// //     </SafeAreaView>

// //   );

// // };

// const HomeScreen = () => {

//   return (

//     <SafeAreaView flex={1}>

//       <View style={styles.MainContainer}>

//         <Text style={{fontSize: 25, color: 'black'}}> Home Screen </Text>

//       </View>

//     </SafeAreaView>

//   );

// };

// const SecondScreen = () => {

//   return (

//     <SafeAreaView flex={1}>

//       <View style={styles.MainContainer}>

//         <Text style={{fontSize: 25, color: 'black'}}> Second Screen </Text>

//       </View>

//     </SafeAreaView>

//   );

// };

// const ThirdScreen = () => {

//   return (

//     <SafeAreaView flex={1}>

//       <View style={styles.MainContainer}>

//         <Text style={{fontSize: 25, color: 'black'}}> Third Screen </Text>

//       </View>

//     </SafeAreaView>

//   );

// };

// const styles = StyleSheet.create({

//   MainContainer: {

//     flex: 1,

//     alignItems: 'center',

//     justifyContent: 'center',

//     padding: 10,

//   },

//   sectionView: {

//     flex: 1,

//     flexDirection: 'row',

//     alignItems: 'center',

//     marginTop: 12,

//   },

//   separatorLine: {

//     flex: 1,

//     backgroundColor: 'black',

//     height: 1.2,

//     marginLeft: 12,

//     marginRight: 24,

//   },

// });

// const Stack = createStackNavigator();

// const Drawer = createDrawerNavigator();

// function HomeStack({navigation}) {

//   return (

//     <Stack.Navigator initialRouteName="HomeScreen">

//       <Stack.Screen

//         name="HomeScreen"

//         component={HomeScreen}

//         options={{

//           title: 'Home Screen',

//           headerLeft: () => <HamburgerIcon navigationProps={navigation} />,

//           headerStyle: {

//             backgroundColor: '#FF9800',

//           },

//           headerTintColor: '#fff',

//         }}

//       />

//     </Stack.Navigator>

//   );

// }

// function SecondStack({navigation}) {

//   return (

//     <Stack.Navigator

//       screenOptions={{

//         headerLeft: () => <HamburgerIcon navigationProps={navigation} />,

//         headerStyle: {

//           backgroundColor: '#FF9800',

//         },

//         headerTintColor: '#fff',

//       }}>

//       <Stack.Screen

//         name="SecondScreen"

//         component={SecondScreen}

//         options={{

//           title: 'Second Screen',

//         }}

//       />

//     </Stack.Navigator>

//   );

// }

// function ThirdStack({navigation}) {

//   return (

//     <Stack.Navigator

//       screenOptions={{

//         headerLeft: () => <HamburgerIcon navigationProps={navigation} />,

//         headerStyle: {

//           backgroundColor: '#FF9800',

//         },

//         headerTintColor: '#fff',

//       }}>

//       <Stack.Screen

//         name="ThirdScreen"

//         component={ThirdScreen}

//         options={{

//           title: 'Third Screen',

//         }}

//       />

//     </Stack.Navigator>

//   );

// }

// const App = () => {

//   return (

//     <NavigationContainer>

//       <Drawer.Navigator

//       // drawerContent={props => <CustomSidebar {...props} />}

//       >

//         <Drawer.Screen

//           name="Home"

//           options={{

//             drawerLabel: 'Open Home Screen',

//             groupName: 'Category 1',

//             activeTintColor: '#FF6F00',

//           }}

//           component={HomeStack}

//         />

//         <Drawer.Screen

//           name="Second"

//           options={{

//             drawerLabel: 'Open Second Screen',

//             groupName: 'Category 1',

//             activeTintColor: '#FF6F00',

//           }}

//           component={SecondStack}

//         />

//         <Drawer.Screen

//           name="Third"

//           options={{

//             drawerLabel: 'Open Third Screen',

//             groupName: 'Category 2',

//             activeTintColor: '#FF6F00',

//           }}

//           component={ThirdStack}

//         />

//       </Drawer.Navigator>

//     </NavigationContainer>

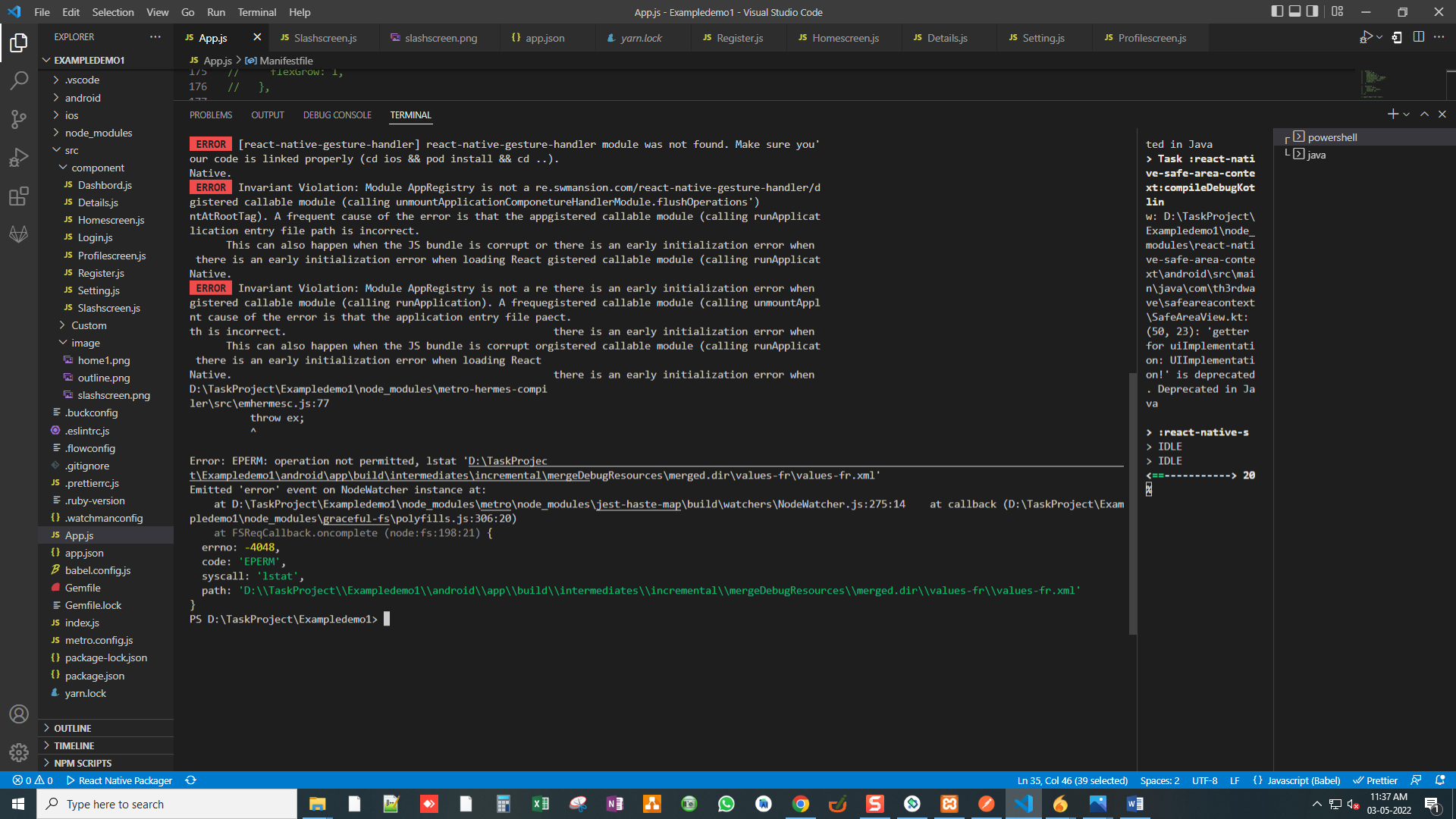
//   );

// };

// export default App;

Website : create custom menu

<https://reactnativecode.com/navigation-drawer-with-section-menu/>



This error solution in reset clear cache and node js delete and start now

Solution :-

<https://github.com/facebook/react-native/issues/26687>

asnc storage in react native of redux

<https://dev.to/skd1993/how-to-save-login-state-locally-using-asyncstorage-redux-in-react-native-31oo>

<https://jasonwatmore.com/post/2020/03/02/react-hooks-redux-user-registration-and-login-tutorial-example>

<https://www.youtube.com/watch?v=t6yN_NtTU6Q>

| Error Code | DescriptionThe following errors may occur during the Login API call. |
| --- | --- |
| 2002 | Version is disabled or provided wrong application info (application id or secret key) |
| 3000 | Login has been disabled for the user account. |
| 3001 | Missing login settings, possibly invalid application id or version. |
| 3002 | User cannot login because Multiple Logins disabled and there is a logged in user for the account. |
| 3003 | Invalid login or password. |
| 3006 | Either login or password is an empty string value. |
| 3034 | User logins are disabled for the version of the application. |
| 3036 | Account locked out due to too many failed logins. |
| 3038 | One of the required parameters (application id, version, login or password) is null |
| 3044 | Multiple login limit for the same user account has been reached. |
| 8000 | Property value exceeds the length limit |

Gitclone command

git clone <https://gitlab.com/ethics-mobile-team/davaindia-reactnative-app.git>

Gitlab command

<https://www.nobledesktop.com/learn/git/git-branches>

web view url load in three or more than

webview screen

:-

import React from 'react';

import { View } from 'react-native';

import { WebView } from 'react-native-webview';

import AppBase from '../AppBase';

import Consts from '../helper/Const';

export default class WebViewScreen extends AppBase {

  constructor(props) {

    super(props);

    this.state = {

      webViewUrl: 'https://www.google.com',

    };

  }

  componentDidMount() {

    let titleName = "";

    let urlType = Consts.Url\_Type;

    urlType = this.props.route.params.selectedUrlType;

    if (urlType == Consts.Url\_Type.Url\_Type\_Privacy\_Policy) {

      titleName = "Privacy Policy";

      this.setState({

        webViewUrl: 'https://policies.google.com/privacy?hl=en-US',

      });

    } else if (urlType == Consts.Url\_Type.Url\_Type\_Refund\_Cancellation) {

      titleName = "Refund & Cancellation Policy";

      this.setState({

        webViewUrl: 'https://www.interweave.in/cancellation-refund-policy/',

      });

    } else if (urlType == Consts.Url\_Type.Url\_Type\_Shipping\_Policy) {

      titleName = "Shipping Policy";

      this.setState({

        webViewUrl:

          'https://wearzuka.com/pages/shipping-policy?gclid=EAIaIQobChMIxO\_\_rrHZ9wIVE5BoCR2tVAdSEAAYASAAEgJDn\_D\_BwE',

      });

    }

    else {

      this.setState({

        webViewUrl:

          'https://google.com',

      })

    }

    this.props.navigation.setOptions({

      title: titleName,

    });

  }

  render() {

    return (

      <View style={{flex:1}}>

        <WebView source={{ uri: this.state.webViewUrl }} />

      </View>

    );

  }

}

User Profile screen

:-

import AsyncStorage from '@react-native-async-storage/async-storage';

import { encode as btoa } from 'base-64';

import moment from 'moment';

import React from 'react';

import {

  Alert, Dimensions, Image, KeyboardAvoidingView, Modal, Platform, SafeAreaView, ScrollView,

  Share, StyleSheet, Switch, TouchableOpacity

} from 'react-native';

import { View } from 'react-native-animatable';

import DeviceInfo from 'react-native-device-info';

import DropDownPicker from 'react-native-dropdown-picker';

import \* as ImagePicker from "react-native-image-picker";

import TextAvatar from 'react-native-text-thumbnail';

import IconMaterialIcons from 'react-native-vector-icons/MaterialIcons'; //https://oblador.github.io/react-native-vector-icons/

import IconEdit from '../../Assets/Images/edit-icon.png';

import NoImage from '../../Assets/Images/noimage.png';

import IconPrivacyPolicy from '../../Assets/Images/Privacy-Policy.png';

import IconRefundCancellation from '../../Assets/Images/Refund-Cancellation.png';

import IconShoppingPolicy from '../../Assets/Images/Shipping-Policy.png';

import AppBase from '../AppBase';

import API from '../connection/http-utils';

import AppTheme from '../helper/AppTheme';

import Consts from '../helper/Const';

import ActionSheet from '../helper/customView/ActionSheet';

import AnimatedButton from '../helper/customView/AnimatedButton';

import BaseView from '../helper/customView/BaseView';

import CustomLoadingView from '../helper/customView/CustomLoadingView';

import CustomText from '../helper/customView/CustomText';

import CustomTextInput from '../helper/customView/CustomTextInput';

import Singleton from '../helper/Singleton';

import strings from '../LanguageFiles/LocalizedStrings';

export default class UserProfile extends AppBase {

  constructor(props) {

    super(props);

    this.state = {

      isConnected: true,

      todayString: moment(new Date()).format('DD-MMM-YYYY'),

      userInfo: '',

      modal: false,

      loadingCounter: false,

      userName: '',

      userMobile: '',

      userEmail: '',

      userAddress: '',

      userCity: '',

      userState: '',

      userZip: '',

      userCountry: '',

      statesList: [],

      cityList: [],

      isSoundOff: Singleton.getInstance().isSoundOff,

      resourcePath: {},

      actionSheet: false,

      actionItems: [

        {

          id: 1,

          label: strings.CAM,

          onPress: () => this.cameraLaunch()

        },

        {

          id: 2,

          label: strings.GAL,

          onPress: () => this.imageGalleryLaunch()

        }

      ],

      imagedata: '',

      base64Image: ''

    };

  }

  getUserData = async () => {

    console.log('isSoundOff = ' + Singleton.getInstance().isSoundOff);

    this.setState(

      {

        loadingCounter: true,

        isSoundOff: Singleton.getInstance().isSoundOff,

      },

      () => {

        API.getRequest('api/Account/getuserprofile')

          .then((dataenc) => {

            const data = API.decrpt(dataenc.jsonResponse);

            console.log(data.data);

            setTimeout(() => {

              this.setState({

                loadingCounter: false,

                userInfo: data.data,

                userName:

                  data.data && data.data.firstName ? data.data.firstName : '',

                userMobile:

                  data.data && data.data.mobileNo ? data.data.mobileNo : '',

                userEmail:

                  data.data && data.data.email ? data.data.email : '',

                userAddress:

                  data.data && data.data.address1 ? data.data.address1 : '',

                userCity:

                  data.data && data.data.city ? parseInt(data.data.city) : 0,

                userState:

                  data.data && data.data.userState

                    ? parseInt(data.data.userState)

                    : 0,

                userZip:

                  data.data && data.data.zipcode ? data.data.zipcode : '',

                imagedata: data.data && data.data.docPath ? data.data.docPath : '',

              }, () => {

                this.getStates();

              });

            }, 1000);

          }).catch((err) => {

            this.setState({

              loadingCounter: false,

            });

            console.log(err);

          });

      }).catch((err) => {

        this.setState({

          loadingCounter: false,

        });

        console.log(err);

        if (err.jsonResponse.status == false) {

          this.showAlertMessage(err.jsonResponse.message);

        }

      });

  };

  getStates = () => {

    this.setState({

      loadingCounter: true,

    }, () => {

      API.getRequest('api/user/statecity')

        .then((data) => {

          this.setState({

            loadingCounter: false,

          });

          var \_statesList = [];

          var \_tempData = data.jsonResponse.data;

          // console.log(\_tempData);

          if (\_tempData.length > 0) {

            \_tempData.map((stateItem) => {

              \_statesList.push({ label: stateItem.name, value: stateItem.id });

            });

            this.setState(

              {

                statesList: \_statesList,

              },

              () => {

                if (this.state.userState) {

                  this.getCity(this.state.userState);

                }

              },

            );

          }

        })

        .catch((err) => {

          this.setState({

            loadingCounter: false,

          });

          console.log(err);

          if (err.jsonResponse.status == false) {

            this.showAlertMessage(err.jsonResponse.message);

          }

        });

    });

  };

  getCity = (userState) => {

    this.setState({

      loadingCounter: true,

    }, () => {

      API.getRequest('api/user/statecity?sid=' + userState)

        .then((data) => {

          var \_cityList = [];

          var \_tempData = data.jsonResponse.data;

          // console.log(\_tempData);

          this.setState({

            loadingCounter: false,

          });

          if (\_tempData.length > 0) {

            \_tempData.map((stateItem) => {

              \_cityList.push({ label: stateItem.name, value: stateItem.id });

            });

            this.setState({

              cityList: \_cityList,

            });

          }

        })

        .catch((err) => {

          this.setState({

            loadingCounter: false,

          });

          console.log(err);

          if (err.jsonResponse.status == false) {

            this.showAlertMessage(err.jsonResponse.message);

          }

        });

    });

  };

  logoutCall = () => {

    this.setState({

      loadingCounter: true,

    }, () => {

      API.getRequest('api/Account/resetfcmtocken')

        .then((data) => {

          console.log('Logout data response');

          console.log(data);

          // API.decrpt(data.jsonResponse);

          this.setState({

            loadingCounter: false,

          });

          if (data.jsonResponse.success) {

            AsyncStorage.removeItem('userInfo');

            AsyncStorage.removeItem('access\_token');

            AsyncStorage.removeItem('loginData');

            this.props.navigation.replace('login');

          } else {

            this.showAlertMessage(data.jsonResponse.message);

            AsyncStorage.removeItem('userInfo');

            AsyncStorage.removeItem('access\_token');

            AsyncStorage.removeItem('loginData');

            this.props.navigation.replace('login');

          }

        })

        .catch((err) => {

          this.setState({

            loadingCounter: false,

          });

          if (err.jsonResponse.status == false) {

            this.showAlertMessage(err.jsonResponse.message);

          }

        });

    });

  }

  componentDidMount = () => {

    this.\_unsubscribe = this.props.navigation.addListener('focus', () => {

      this.getUserData();

    });

  };

  processToLogOut = () => {

    this.logoutCall();

  };

  goAddressList = () => {

    this.props.navigation.push('addressList');

  };

  goFavorite = () => {

    this.props.navigation.push('favorite');

  };

  goLanguage = () => {

    this.props.navigation.push('selectLanguage');

  };

  goAboutUs = () => {

    this.props.navigation.push('about');

  };

  goFeedback = () => {

    this.props.navigation.push('feedback');

  };

  goContactUs = () => {

    this.props.navigation.push('contact');

  };

  editProfile = () => {

    if (this.state.statesList.length > 0) {

      this.setState({

        modal: true,

      });

    }

  };

  updateProfile = () => {

    if (this.state.userName.length <= 0) {

      this.showAlertMessage(strings.Valid\_User\_Name);

    } else if (!this.validateEmail(this.state.userEmail)) {

      this.showAlertMessage(strings.Valid\_Email);

    } else if (!this.validateZip(this.state.userZip)) {

      this.showAlertMessage(strings.Valid\_Zip);

    } else {

      var params = {

        userName: this.state.userMobile,

        firstName: this.state.userName,

        email: this.state.userEmail,

        address1: this.state.userAddress,

        city: this.state.userCity,

        userState: this.state.userState,

        zipcode: this.state.userZip,

      };

      var strReq = btoa(API.encrpt(JSON.stringify(params)));

      API.postRequest(

        'api/Account/updateprofilenew',

        API.encryptedString(strReq),

      )

        .then((dataenc) => {

          var data = API.decrpt(dataenc.jsonResponse);

          if (

            (data && data.message == 'Invalid Request') ||

            data.message == 'User not found'

          ) {

            this.showAlertMessage(data.message);

            this.setState({

              loadingCounter: false,

            });

          } else {

            // this.showAlertMessage(data.message);

            this.showToastMessage(data.message);

            this.setState(

              {

                modal: false,

              },

              () => {

                if (this.state.base64Image) {

                  this.updatePhoto(this.state.base64Image);

                }

                else {

                  this.getUserData();

                }

              },

            );

          }

        })

        .catch((err) => {

          this.setState({

            loadingCounter: false,

          });

          console.log(err);

          if (err.jsonResponse.status == false) {

            this.showAlertMessage(err.jsonResponse.message);

          }

        });

    }

  };

  shareApp = async () => {

    try {

      await API.getRequest('api/AppMaster/getshareappdetail')

        .then((data) => {

          // console.log(data.jsonResponse.status)

          if (data.jsonResponse.success) {

            const result = Share.share({

              message:

                data.jsonResponse.data.shareAppText +

                '\n' +

                data.jsonResponse.data.shareLink,

              title: data.jsonResponse.data.shareAppSubject,

            });

            if (result.action === Share.sharedAction) {

              if (result.activityType) {

                // shared with activity type of result.activityType

                console.log(result.activityType);

              } else {

                // shared

                console.log('Shared Success');

              }

            } else if (result.action === Share.dismissedAction) {

              // dismissed

              console.log('Shared Dismiss');

            }

          }

        })

        .catch((err) => {

          console.log(err);

          if (err.jsonResponse.status == false) {

            this.showAlertMessage(err.jsonResponse.message);

          }

        });

    } catch (error) {

      alert(error.message);

    }

  };

  logoutAlertMessage = (message) => {

    Alert.alert(

      'Alert',

      message,

      [

        {

          text: strings.CANCEL,

          onPress: () => console.log('Cancel Pressed'),

        },

        {

          text: strings.LOGOUT,

          style: 'destructive',

          onPress: () => {

            console.log('OK Pressed');

            this.processToLogOut();

          },

        },

      ],

      { cancelable: true },

    );

  };

  updatePhoto = (base64) => {

    var params = {

      Base64Document: base64,

      DocName: this.state.userMobile + '.jpg',

      RefType: "UserImage"

    };

    this.setState({

      loadingCounter: true

    });

    // var strReq = btoa(API.encrpt(JSON.stringify(params)));

    API.postRequest('api/common/uploaduserimage', params).then((response) => {

      this.setState({

        // loadingCounter: false,

        base64Image: ''

      }, () => {

        this.getUserData();

      });

      if (!response.jsonResponse.status) {

        this.showAlertMessage(response.jsonResponse.message);

      } else {

        this.showToastMessage(response.jsonResponse.message);

      }

      // console.log(response.jsonResponse);

    }).catch((err) => {

      this.setState({

        loadingCounter: false,

      });

      console.log(err);

      if (err.jsonResponse.status == false) {

        this.showAlertMessage(err.jsonResponse.message);

      }

    });

  }

  cameraLaunch = () => {

    this.closeActionSheet();

    let options = {

      storageOptions: {

        skipBackup: true,

        path: 'images',

      },

      quality: 0.3,

      includeBase64: true

    };

    setTimeout(() => {

      ImagePicker.launchCamera(options, (res) => {

        // console.log('Response = ', res.assets[0].base64);

        if (res.didCancel) {

          console.log('User cancelled image picker');

        } else if (res.error) {

          console.log('ImagePicker Error: ', res.error);

        } else if (res.customButton) {

          console.log('User tapped custom button: ', res.customButton);

          alert(res.customButton);

        } else {

          // console.log(res.assets[0].uri);

          this.setState({

            imagedata: res.assets[0].uri,

            base64Image: res.assets[0].base64

          }, () => {

            // this.updatePhoto(res.assets[0].base64);

          });

        }

      });

    }, 500);

  }

  imageGalleryLaunch = () => {

    this.closeActionSheet();

    let options = {

      storageOptions: {

        skipBackup: true,

        path: 'images',

      },

      quality: 0.3,

      includeBase64: true

    };

    ImagePicker.launchImageLibrary(options, (res) => {

      // console.log('Response = ', res);

      if (res.didCancel) {

        console.log('User cancelled image picker');

      } else if (res.error) {

        console.log('ImagePicker Error: ', res.error);

      } else if (res.customButton) {

        console.log('User tapped custom button: ', res.customButton);

        alert(res.customButton);

      } else {

        console.log(res.assets[0].uri);

        this.setState({

          imagedata: res.assets[0].uri,

          base64Image: res.assets[0].base64

        }, () => {

          // this.updatePhoto(res.assets[0].base64);

        });

      }

    });

  }

  setActionSheet = () => {

    console.log('SHOW ACTIONSHEET');

    this.setState({

      // modal: false,

      actionSheet: true

    })

  }

  closeActionSheet = () => {

    this.setState({

      actionSheet: false

    })

  }

  render() {

    return (

      <BaseView style={{ flex: 1 }}>

        <CustomLoadingView

          isShowModal={this.state.loadingCounter ? true : false}

        />

        <Modal

          style={[styles.modalcss, { zIndex: 99 }]}

          animationType={'slide'}

          transparent={true}

          backdropOpacity={0.1}

          visible={this.state.modal}

          onRequestClose={() => {

            this.setState({ modal: false });

          }}>

          <KeyboardAvoidingView

            behavior={"padding"}

            keyboardVerticalOffset={Platform.OS == "ios" ? 0 : 50}

            enabled={Platform.OS === "ios" ? true : false} style={styles.modalmainview}>

            <View style={styles.boxview}>

              <ScrollView>

                <View style={styles.centerContent}>

                  <TouchableOpacity onPress={() => this.setActionSheet()} >

                    <View style={styles.whitecircle}>

                      <View style={styles.circleImage}>

                        {this.state.imagedata ? (

                          <Image

                            source={{

                              uri: this.state.imagedata + "?date=" + moment().valueOf(),

                              cache: "reload",

                              headers: {

                                Pragma: "no-cache"

                              }

                            }}

                            key={new Date()}

                            style={styles.productIMG}

                            resizeMode={"stretch"}

                          />

                        ) : (

                          <Image source={NoImage} style={styles.productIMG} />

                        )}

                      </View>

                    </View>

                  </TouchableOpacity>

                </View>

                <CustomTextInput

                  ref={(ref) => (this.nameInputTextRef = ref)}

                  name={'Name'}

                  // maxLength={35}

                  value={this.state.userName}

                  style={styles.textinput}

                  maxLength={20}

                  returnKeyType={'next'}

                  placeholder={strings.NAME}

                  placeholderTextColor={AppTheme.APPCOLOR.TEXT}

                  onChangeText={(value) => this.setState({ userName: value })}

                />

                <CustomTextInput

                  ref={(ref) => (this.mobileInputTextRef = ref)}

                  name={'Mobile'}

                  maxLength={10}

                  value={this.state.userMobile}

                  editable={false}

                  style={styles.textinput}

                  returnKeyType={'next'}

                  placeholder={strings.MOBILE}

                  placeholderTextColor={AppTheme.APPCOLOR.TEXT}

                  onChangeText={(value) => this.setState({ userMobile: value })}

                />

                <CustomTextInput

                  ref={(ref) => (this.emailInputTextRef = ref)}

                  name={'Email'}

                  keyboardType={'email-address'}

                  value={this.state.userEmail}

                  style={styles.textinput}

                  returnKeyType={'next'}

                  placeholder={strings.EMAIL}

                  placeholderTextColor={AppTheme.APPCOLOR.TEXT}

                  onChangeText={(value) => this.setState({ userEmail: value })}

                />

                <CustomTextInput

                  ref={(ref) => (this.addressInputTextRef = ref)}

                  name={'Address'}

                  value={this.state.userAddress}

                  style={styles.textinput}

                  returnKeyType={'done'}

                  maxLength={50}

                  multiline={true}

                  placeholder={strings.ADDRESS}

                  placeholderTextColor={AppTheme.APPCOLOR.TEXT}

                  onChangeText={(value) => this.setState({ userAddress: value })}

                />

                {this.state.statesList.length > 0 ? (

                  <DropDownPicker

                    zIndex={2000}

                    items={this.state.statesList}

                    placeholder={strings.STATE}

                    //style={styles.textinput}

                    activeLabelStyle={{ color: AppTheme.APPCOLOR.PRIMARY }}

                    defaultValue={

                      this.state.userState ? this.state.userState : null

                    }

                    containerStyle={{ height: 40, marginTop: 15 }}

                    style={{ backgroundColor: AppTheme.APPCOLOR.WHITE }}

                    itemStyle={{

                      justifyContent: 'flex-start',

                    }}

                    dropDownStyle={{ backgroundColor: AppTheme.APPCOLOR.WHITE }}

                    onChangeItem={(item) => {

                      this.setState({

                        userState: item.value,

                        userCity: '',

                      });

                      this.getCity(item.value);

                    }}

                  />

                ) : null}

                {this.state.userState && this.state.cityList.length > 0 ? (

                  <DropDownPicker

                    zIndex={1000}

                    items={this.state.cityList}

                    placeholder={strings.CITY}

                    //style={styles.textinput}

                    activeLabelStyle={{ color: AppTheme.APPCOLOR.PRIMARY }}

                    defaultValue={

                      this.state.userCity ? this.state.userCity : null

                    }

                    containerStyle={{ height: 40, marginTop: 15 }}

                    style={{ backgroundColor: AppTheme.APPCOLOR.WHITE }}

                    itemStyle={{

                      justifyContent: 'flex-start',

                    }}

                    dropDownStyle={{ backgroundColor: AppTheme.APPCOLOR.WHITE }}

                    onChangeItem={(item) => {

                      this.setState({

                        userCity: item.value,

                      });

                      console.log(item.label, item.value);

                    }}

                  />

                ) : null}

                <CustomTextInput

                  ref={(ref) => (this.zipInputTextRef = ref)}

                  name={'Zip'}

                  value={this.state.userZip.toString()}

                  style={styles.textinput}

                  keyboardType={'phone-pad'}

                  maxLength={6}

                  returnKeyType={'done'}

                  placeholder={strings.ZIP}

                  placeholderTextColor={AppTheme.APPCOLOR.TEXT}

                  onChangeText={(value) => this.setState({ userZip: value })}

                />

                <View

                  style={[

                    styles.centerContent,

                    { borderRadius: 0, borderWidth: 0 },

                  ]}>

                  <View style={{ flexDirection: 'row' }}>

                    <AnimatedButton

                      onPress={() => {

                        this.updateProfile();

                      }}

                      style={styles.savebtn}>

                      <CustomText

                        text={strings.UPDATE}

                        customStyle={styles.savetxt}

                      />

                    </AnimatedButton>

                    <AnimatedButton

                      onPress={() => {

                        this.setState({ modal: false }, () => this.getUserData());

                      }}

                      style={styles.savebtn}>

                      <CustomText

                        text={strings.CANCEL}

                        customStyle={styles.savetxt}

                      />

                    </AnimatedButton>

                  </View>

                </View>

              </ScrollView>

            </View>

          </KeyboardAvoidingView>

          <Modal

            style={{ justifyContent: 'flex-end', margin: 0, zIndex: 999 }}

            animationType={'slide'}

            transparent={true}

            backdropOpacity={0.1}

            visible={this.state.actionSheet}

          >

            <View style={styles.actionsheet\_containerStyle}>

              <View style={styles.actionsheet\_content}>

                <ActionSheet

                  actionItems={this.state.actionItems}

                  onCancel={this.closeActionSheet}

                />

              </View>

            </View>

          </Modal>

        </Modal>

        <SafeAreaView

          style={{ backgroundColor: AppTheme.APPCOLOR.PRIMARY, flex: 1 }}>

          <ScrollView

            style={{ backgroundColor: AppTheme.APPCOLOR.WHITE, flex: 1 }}>

            <View style={styles.headerbox}>

              <View style={styles.row}>

                <View style={{ flex: 0.3, alignItems: 'center' }}>

                  <View style={styles.whitecircle}>

                    <View style={styles.circleImage}>

                      {

                        this.state.imagedata ?

                          <Image

                            source={{

                              uri: this.state.imagedata + "?date=" + moment().valueOf(),

                              cache: "reload",

                              headers: {

                                Pragma: "no-cache"

                              }

                            }}

                            key={new Date()}

                            style={styles.profilePic}

                            resizeMode={"stretch"}

                          />

                          :

                          <TextAvatar

                            backgroundColor={AppTheme.APPCOLOR.WHITE}

                            textColor={AppTheme.APPCOLOR.PRIMARY}

                            size={70}

                            type={'circle'}>

                            {this.state.userInfo.firstName

                              ? this.state.userInfo.firstName

                              : 'NA'}

                          </TextAvatar>

                      }

                    </View>

                  </View>

                </View>

                <View style={{ flex: 0.7 }}>

                  <View style={{ flexDirection: 'row' }}>

                    <CustomText

                      customStyle={[styles.userName]}

                      text={

                        this.state.userInfo.firstName

                          ? this.state.userInfo.firstName

                          : 'NA'

                      }

                    />

                    <AnimatedButton

                      style={styles.editAnimatedButton}

                      onPress={() => {

                        this.editProfile();

                      }}>

                      <View style={{ alignItems: 'center', flexDirection: 'row' }}>

                        <Image

                          source={IconEdit}

                          resizeMode="cover"

                          style={styles.editicon}

                        />

                        <CustomText

                          customStyle={[styles.editText]}

                          text={

                            strings.Edit

                          }

                        />

                      </View>

                    </AnimatedButton>

                  </View>

                  <CustomText

                    customStyle={[styles.userMobile]}

                    text={

                      this.state.userInfo.mobileNo

                        ? this.state.userInfo.mobileNo

                        : 'NA'

                    }

                  />

                </View>

              </View>

            </View>

            <View style={styles.centercontent}>

              <View style={styles.menuItemContainer}>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.goAboutUs();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="info"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.ABOUTUS}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.goFavorite();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="favorite"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.FAVORITE}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.goAddressList();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="place"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.ADDRESS\_TITLE}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                {/\* <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.goLanguage();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="language"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.LANGUAGE}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton> \*/}

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.goFeedback();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="feedback"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.FEEDBACK}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.shareApp();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="share"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.SHAREAPP}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.props.navigation.push('WebViewScreen', { "selectedUrlType": Consts.Url\_Type.Url\_Type\_Refund\_Cancellation });

                    // this.props.navigation.push('WebViewScreen', {

                    //   selectedUrlType: Consts.UrlType.Url\_Type\_Refund\_Cancellation,

                    // });

                  }}>

                  <View style={styles.menuItemBox}>

                    <Image

                      style={{ alignSelf: 'center', height: 25, width: 25 }}

                      source={IconRefundCancellation}

                    />

                    <CustomText

                      text={strings.REFUND\_CANCELLATION\_POLICY}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.goContactUs();

                  }}>

                  <View style={styles.menuItemBox}>

                    <IconMaterialIcons

                      name="email"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.CONTACTUS}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <View style={styles.menuItem}>

                  <View style={[styles.menuItemBox, { flex: 2 }]}>

                    <IconMaterialIcons

                      name={this.state.isSoundOff === false

                        ? 'volume-up'

                        : 'volume-off'}

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.SOUND}

                      customStyle={[styles.textstyle, { flex: 1.6 }]}

                    />

                    <Switch

                      trackColor={{ false: AppTheme.APPCOLOR.GREEN, true: AppTheme.APPCOLOR.PRIMARY }}

                      thumbColor={AppTheme.APPCOLOR.GREY}

                      // ios\_backgroundColor={AppTheme.APPCOLOR.GREEN}

                      onValueChange={() => {

                        this.setState({ isSoundOff: !this.state.isSoundOff }, () => {

                          Singleton.getInstance().turnSoundOnOff(

                            this.state.isSoundOff,

                          );

                        });

                      }}

                      style={{ flex: 0.4 }}

                      value={!this.state.isSoundOff}

                    />

                  </View>

                </View>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.props.navigation.push('WebViewScreen', {

                      "selectedUrlType": Consts.Url\_Type.Url\_Type\_Privacy\_Policy,

                    });

                  }

                  }>

                  <View

                    style={{

                      flexDirection: 'row',

                      padding: 15,

                      borderBottomColor: AppTheme.APPCOLOR.GRAY,

                      borderBottomWidth: 1,

                    }}>

                    <Image

                      style={{ alignSelf: 'center', height: 25, width: 25 }}

                      source={IconPrivacyPolicy}

                    />

                    <CustomText

                      text={strings.PRIVACY\_POLICY}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    console.warn("shipping pressed");

                    this.props.navigation.push('WebViewScreen', {

                      "selectedUrlType": Consts.Url\_Type.Url\_Type\_Shipping\_Policy,

                    });

                  }

                  }>

                  <View

                    style={{

                      flexDirection: 'row',

                      padding: 15,

                      borderBottomColor: AppTheme.APPCOLOR.GRAY,

                      borderBottomWidth: 1,

                    }}>

                    <Image

                      style={{ alignSelf: 'center', height: 25, width: 25 }}

                      source={IconShoppingPolicy}

                    />

                    <CustomText

                      text={strings.SHIPPING\_POLICY}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

                <AnimatedButton

                  style={styles.menuItem}

                  onPress={() => {

                    this.logoutAlertMessage(strings.LOGOUT\_CONFIRMATION);

                  }}>

                  <View style={{ flexDirection: 'row', padding: 15 }}>

                    <IconMaterialIcons

                      name="logout"

                      size={25}

                      color={AppTheme.APPCOLOR.GREEN}

                      style={styles.itemicon}

                    />

                    <CustomText

                      text={strings.LOGOUT}

                      customStyle={styles.textstyle}

                    />

                  </View>

                </AnimatedButton>

              </View>

              <View style={{ paddingTop: 20 }}>

                <CustomText

                  text={

                    strings.VERSION +

                    ' ' +

                    DeviceInfo.getVersion() +

                    ' (' +

                    DeviceInfo.getBuildNumber() +

                    ')'

                  }

                  customStyle={{

                    fontFamily: 'verdana',

                    fontWeight: '500',

                    color: 'gray',

                  }}

                />

              </View>

            </View>

          </ScrollView>

        </SafeAreaView>

      </BaseView>

    );

  }

}

const styles = StyleSheet.create({

  actionsheet\_containerStyle: {

    flex: 1,

    flexDirection: 'row',

    justifyContent: 'space-around',

    alignItems: 'flex-end',

    backgroundColor: '#00000080',

    // backgroundColor: AppTheme.APPCOLOR.PRIMARY

  },

  actionsheet\_content: {

    width: '99%',

    overflow: 'hidden'

  },

  row: {

    flexDirection: 'row',

    padding: 5,

    top: 10,

    flex: 1,

  },

  col: {

    flexDirection: 'column',

    alignItems: 'flex-start',

  },

  userName: {

    textAlign: 'left',

    color: AppTheme.APPCOLOR.WHITE,

    fontSize: 22,

    fontWeight: '700',

    // left: 10,

    // flex:0.5,

    width: '70%',

  },

  editText: {

    textAlign: 'center',

    color: AppTheme.APPCOLOR.WHITE,

    fontWeight: '500',

  },

  userMobile: {

    textAlign: 'left',

    color: AppTheme.APPCOLOR.WHITE,

    fontSize: 15,

    fontWeight: '500',

    // left: 10,

  },

  menuItemContainer: {

    width: '95%',

    backgroundColor: AppTheme.APPCOLOR.WHITE,

    marginTop: -30,

    justifyContent: 'center',

    borderWidth: 1,

    borderRadius: 5,

    elevation: 5,

  },

  menuItemBox: {

    flexDirection: 'row',

    borderBottomColor: AppTheme.APPCOLOR.GRAY,

    borderBottomWidth: 1,

    padding: 15,

    // flex:1

  },

  editAnimatedButton: {

    borderColor: AppTheme.APPCOLOR.WHITE,

    borderWidth: 1,

    padding: 5,

    borderRadius: 10,

    // paddingHorizontal: 10,

  },

  menuItem: {

    color: AppTheme.APPCOLOR.PRIMARY,

    fontWeight: 'bold',

    fontSize: 4,

    // flex:0.5

    // marginLeft: 20

  },

  textstyle: {

    fontSize: 20,

    fontWeight: '700',

    marginLeft: 20,

    // width: '100%',

  },

  headerbox: {

    backgroundColor: AppTheme.APPCOLOR.PRIMARY,

    height: 150,

    // paddingLeft: 20

  },

  editicon: {

    alignSelf: 'center',

    height: 20,

    width: 20,

    margin: 5,

    // marginLeft: 20,

  },

  itemicon: {

    alignSelf: 'center',

    height: 25,

    width: 25,

  },

  centercontent: {

    alignItems: 'center',

    justifyContent: 'center',

  },

  modalcss: {

    justifyContent: 'center',

    alignItems: 'center',

    // height: '100%',

    flex: 1,

    margin: 10,

  },

  modalmainview: {

    flex: 1,

    width: '100%',

    backgroundColor: '#00000080',

    flexDirection: 'column',

    justifyContent: 'center',

    alignItems: 'center',

  },

  boxview: {

    width: '95%',

    // margin: '5%',

    borderRadius: 10,

    overflow: 'hidden',

    justifyContent: 'center',

    alignItems: 'center',

    backgroundColor: AppTheme.APPCOLOR.WHITE,

    // padding: 20,

  },

  centerContent: {

    marginTop: 15,

    justifyContent: 'center',

    alignItems: 'center',

    // backgroundColor: 'red',

    alignSelf: 'center',

  },

  modalimage: {

    width: 80,

    height: 80,

    alignItems: 'center',

    justifyContent: 'center',

    textAlign: 'center',

  },

  textinput: {

    borderBottomColor: AppTheme.APPCOLOR.GRAY,

    borderBottomWidth: 1,

    width: '100%'

  },

  savebtn: {

    justifyContent: 'center',

    backgroundColor: AppTheme.APPCOLOR.PRIMARY,

    alignItems: 'center',

    height: 44,

    margin: 20,

    borderRadius: 10,

    width: 130,

  },

  savetxt: {

    color: 'white',

    textAlign: 'center',

  },

  sheet: {

    position: "absolute",

    top: Dimensions.get("window").height,

    left: 0,

    right: 0,

    height: "100%",

    justifyContent: "flex-end",

  },

  popup: {

    backgroundColor: "#FFF",

    marginHorizontal: 10,

    borderTopLeftRadius: 5,

    borderTopRightRadius: 5,

    minHeight: 80,

    alignItems: "center",

    justifyContent: "center",

  },

  circleImage: {

    elevation: 5,

    borderRadius: 50,

    justifyContent: 'center',

    overflow: 'hidden',

    alignItems: 'center',

    borderColor: AppTheme.APPCOLOR.PRIMARY,

    borderWidth: 2,

  },

  whitecircle: {

    elevation: 5,

    borderRadius: 50,

    overflow: 'hidden',

    borderColor: AppTheme.APPCOLOR.WHITE,

    borderWidth: 3,

  },

  productIMG: {

    width: 70,

    height: 70,

  },

  profilePic: {

    width: 70,

    height: 70,

    borderRadius: 50

  }

});

Const :-

export default class Consts {

  static myInstance = null;

  static GridColumns = 2;

  /\*\*

   \* @returns {Consts}

   \*/

  static getInstance() {

    if (Consts.myInstance == null) {

      console.warn('Create new instance');

      Consts.myInstance = new Consts();

      //   getGridColumns()

    }

    return this.myInstance;

  }

  // getGridColumns() {

  //   if (Platform.isPad) {

  //     // iPad

  //     // return 500

  //     this.GridColumns = 4;

  //   } else {

  //     if (Platform.OS == 'android' && DeviceInfo.isTablet) {

  //       // tablet

  //       // return 500

  //       this.GridColumns = 4;

  //     } else {

  //       //iPhone

  //       // return 250

  //       this.GridColumns = 2;

  //     }

  //   }

  //   return this.GridColumns;

  // }

  // static APP\_SCREENS = {

  //   // SPLASH1: splash1,

  //   USERDRAWER: UserDrawer,

  //   DASHBOARD: Dashboard,

  //   POPULAR\_MOULDINGS: PopularMouldings,

  //   CAROUSAL\_SAMPLE: CarousalSample,

  //   TYPES\_LIST: TypesList,

  //   FILTER: Filter,

  //   SEARCH\_RESULT\_MOULDINGS: SearchResultMouldings,

  //   MATERIAL\_LIST: MaterialList,

  //   COLLECTION\_LIST: CollectionList,

  //   SEARCH\_RESULT\_COLLECTION: SearchResultCollection,

  //   PROFILE\_DETAILS: ProfileDetails,

  //   SEARCH\_PROFILE: SearchProfile,

  //   SEARCH\_RESULT\_WITHOUT\_HEADER: SearchresultMouldingsWithoutHeader,

  // };

  // static DRAWER\_NAME = {

  //   // HOME\_DRAWER: 'Home',

  //   POPULAR\_MOULDINGS\_DRAWER: 'PopularMouldingsDrawer',

  //   TYPES\_DRAWER: 'TypeListDrawer',

  //   MATERIAL\_DRAWER: 'MaterialDrawer',

  //   COLLECTION\_DRAWER: 'CollectionDrawer',

  //   PROFILE\_NUMBER\_DRAWER: 'ProfileNumberDrawer',

  //   FIND\_RETAILER\_DRAWER: 'FindRetailerDrawer',

  //   INSPIRATION\_GALLERY\_DRAWER: 'InspirationGalleryDrawer',

  // };

  static Url\_Type = {

    Url\_Type\_Privacy\_Policy:1,

    Url\_Type\_Refund\_Cancellation:2,

    Url\_Type\_Shipping\_Policy:3,

  };

  static SCREEN\_TITLES = {

    // SPLASH1: 'splash1',

    USERDRAWER: 'UserDrawer',

    DASHBOARD: 'Dashboard',

    POPULAR\_MOULDINGS: 'PopularMouldings',

    TYPES\_LIST: 'TypesList',

    PROFILE\_LIST: 'ProfileList',

    FILTER: 'Filter',

    FIND\_RETAILER: 'FindRetailer',

    SEARCH\_RESULT\_MOULDINGS: 'SearchResultMouldings',

    MATERIAL\_LIST: 'MaterialList',

    COLLECTION\_LIST: 'CollectionList',

    SEARCH\_RESULT\_COLLECTION: 'SearchResultCollection',

    PROFILE\_DETAILS: 'ProfileDetails',

    GALLERY\_LIST: 'GalleryList',

    GALLERY\_ITEM: 'GalleryItem',

    SEARCH\_PROFILE: 'SearchProfile',

    SEARCH\_RESULT\_WITHOUT\_HEADER: 'SearchresultMouldingsWithoutHeader',

    CAROUSAL\_SAMPLE: 'CarouselSample',

  };

}

// import React, { Component } from 'react';

// import { View, Text, Image, ScrollView, Dimensions, StyleSheet } from 'react-native';

// const image = [

//     "https://res.cloudinary.com/twenty20/private\_images/t\_watermark-criss-cross-10/v1435993328000/photosp/cbee31ab-1619-4b83-ad45-07ecd1a7f85e/stock-photo-portrait-coffee-confidence-mug-smile-man-corporate-cbee31ab-1619-4b83-ad45-07ecd1a7f85e.jpg"

//     , "https://res.cloudinary.com/twenty20/private\_images/t\_watermark-criss-cross-10/v1435993328000/photosp/cbee31ab-1619-4b83-ad45-07ecd1a7f85e/stock-photo-portrait-coffee-confidence-mug-smile-man-corporate-cbee31ab-1619-4b83-ad45-07ecd1a7f85e.jpg",

//     "https://res.cloudinary.com/twenty20/private\_images/t\_watermark-criss-cross-10/v1435993328000/photosp/cbee31ab-1619-4b83-ad45-07ecd1a7f85e/stock-photo-portrait-coffee-confidence-mug-smile-man-corporate-cbee31ab-1619-4b83-ad45-07ecd1a7f85e.jpg"

// ]

// const style = StyleSheet.create({

//     dot: {

//         margin: 3,

//         color: '#888'

//     },

//     wrapdot: {

//         position: 'absolute',

//         bottom: 0,

//         flexDirection: 'row',

//         alignSelf: 'center'

//     },

//     dotActive: {

//         margin: 3,

//         color: 'black'

//     }

// })

// export default class Dashboard extends Component {

//     constructor(props) {

//         super(props);

//         this.state = {

//             active: 0

//         };

//     }

//     change(nativeEvent) {

//         console.log("native Event :-> ", nativeEvent)

//         const silder = Math.ceil(nativeEvent.contentOffset.x / nativeEvent.layoutMeasurement.width)

//         if (silder !== this.state.active) {

//             this.setState({ active: silder })

//         }

//     }

//     ImageSilder = () => {

//         return (

//             <View>

//                 <View>

//                     <ScrollView onScroll={({ nativeEvent }) => this.change(nativeEvent)} horizontal showsHorizontalScrollIndicator={false} style={{ height: Dimensions.get('screen').height \* 0.25 }}>

//                         {

//                             image.map((e, index) =>

//                                 <Image

//                                     key={e}

//                                     source={{ uri: e }}

//                                     style={{ width: Dimensions.get('screen').width, height: Dimensions.get('screen').height \* 0.25 }}

//                                 />

//                             )

//                         }

//                     </ScrollView>

//                     <View style={style.wrapdot}>

//                         {

//                             image.map((e, index) =>

//                                 <Text key={e} style={this.state.active == index ? style.dotActive : style.dot}>

//                                     ●

//                                 </Text>

//                             )

//                         }

//                     </View>

//                 </View>

//             </View>

//         )

//     }

//     render() {

//         const { active } = this.state

//         return (

//             <View>

//                 <Text>sdfiho</Text>

//                 {this.ImageSilder()}

//                 <Text> Dashboard </Text>

//             </View>

//         );

//     }

// }

// import React, { Component } from "react";

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

// import BackgroundCarousel from "./BackgroundCarousel";

// const images = [

//     "https://images.unsplash.com/photo-1508138221679-760a23a2285b?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=500&q=60",

//     "https://images.unsplash.com/photo-1485550409059-9afb054cada4?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=701&q=80",

//     "https://images.unsplash.com/photo-1519125323398-675f0ddb6308?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1050&q=80",

//     "https://images.unsplash.com/photo-1429087969512-1e85aab2683d?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=634&q=80",

//     "https://images.unsplash.com/photo-1505678261036-a3fcc5e884ee?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1050&q=80"

// ];

// export default class Dashboard extends Component {

//     render() {

//         return (

//             <View style={styles.container}>

//                 <BackgroundCarousel images={images} />

//             </View>

//         );

//     }

// }

// const styles = StyleSheet.create({

//     container: {

//         flex: 1,

//         justifyContent: "center",

//         alignItems: "center",

//         backgroundColor: "#F5FCFF"

//     }

// });

import React from "react";

import { StyleSheet, View, ScrollView, Dimensions, Image } from "react-native";

const DEVICE\_WIDTH = Dimensions.get("window").width;

class BackgroundCarousel extends React.Component {

    scrollRef = React.createRef();

    constructor(props) {

        super(props);

        this.state = {

            selectedIndex: 0

        };

        this.scrollRef = React.createRef();

    }

    componentDidMount = () => {

        setInterval(() => {

            this.setState(

                prev => ({

                    selectedIndex:

                        prev.selectedIndex === this.props.images.length - 1

                            ? 0

                            : prev.selectedIndex + 1

                }),

                () => {

                    this.scrollRef.current.scrollTo({

                        animated: true,

                        x: DEVICE\_WIDTH \* this.state.selectedIndex,

                        y: 0

                    });

                }

            );

        }, 3000);

    };

    setSelectedIndex = event => {

        const contentOffset = event.nativeEvent.contentOffset;

        const viewSize = event.nativeEvent.layoutMeasurement;

        // Divide the horizontal offset by the width of the view to see which page is visible

        const selectedIndex = Math.floor(contentOffset.x / viewSize.width);

        this.setState({ selectedIndex });

    };

    render() {

        const { images } = this.props;

        const { selectedIndex } = this.state;

        return (

            <View style={{ height: "100%", width: "100%" }}>

                <ScrollView

                    horizontal

                    pagingEnabled

                    onMomentumScrollEnd={this.setSelectedIndex}

                    ref={this.scrollRef}

                >

                    {images.map(image => (

                        <Image

                            style={styles.backgroundImage}

                            source={{ uri: image }}

                            key={image}

                        />

                    ))}

                </ScrollView>

                <View style={styles.circleDiv}>

                    {images.map((image, i) => (

                        <View

                            style={[

                                styles.whiteCircle,

                                { opacity: i === selectedIndex ? 0.5 : 1 }

                            ]}

                            key={image}

                            active={i === selectedIndex}

                        />

                    ))}

                </View>

            </View>

        );

    }

}

const styles = StyleSheet.create({

    backgroundImage: {

        height: Dimensions.get('screen').height \* 78,

        width: Dimensions.get("window").width / 1

    },

    circleDiv: {

        position: "absolute",

        bottom: 15,

        display: "flex",

        flexDirection: "row",

        alignItems: "center",

        justifyContent: "center",

        width: "100%",

        height: 10

    },

    whiteCircle: {

        width: 6,

        height: 6,

        borderRadius: 3,

        margin: 5,

        backgroundColor: "#fff"

    }

});

export default BackgroundCarousel;

Image silder list :-

<https://reactnativecode.com/create-app-intro-slider/>

<https://reactnativecode.com/auto-image-slider-gallery-with-indicator/>

hight rating Ecommerce app

1:- <https://play.google.com/store/apps/details?id=com.incroyable.mantra>

2:- <https://play.google.com/store/apps/details?id=in.dmart>

3:- <https://play.google.com/store/apps/details?id=com.fsn.nykaa>

4:- <https://play.google.com/store/apps/details?id=com.bigbasket.mobileapp>

Design template

1:-https://www.pinterest.com/pin/884112970569311363/?mt=login

2:- <https://www.pinterest.com/pin/674484481704626269/?mt=login>

Sh1 key genretae in report

./gradlew signingReport