**Q1: List the Different Location Permissions in Context of Accuracy and Explain the Scenarios Where You Can Use Each**

**Android Location Permissions**

1. **ACCESS\_FINE\_LOCATION**
   * **Description**: Provides high accuracy location access using GPS and network location sources.
   * **Use Case**: Ideal for apps that require precise location data, such as navigation apps, location-based games, and applications that provide turn-by-turn directions.
2. **ACCESS\_COARSE\_LOCATION**
   * **Description**: Provides access to approximate location using Wi-Fi and cell towers. Less accurate than ACCESS\_FINE\_LOCATION.
   * **Use Case**: Suitable for apps that need general location information, such as weather apps or location-based reminders, where pinpoint accuracy is not crucial.

**iOS Location Permissions**

1. **LOCATION\_WHEN\_IN\_USE**
   * **Description**: Allows the app to access location services only when the app is in the foreground.
   * **Use Case**: Useful for apps that need location information while actively being used, such as fitness trackers, social networking apps with location sharing, or apps that display local points of interest.
2. **LOCATION\_ALWAYS**
   * **Description**: Allows the app to access location services even when the app is running in the background.
   * **Use Case**: Necessary for apps that need to track location continuously, such as fleet management apps, location-based notifications (geofencing), or apps that log user movements for later analysis.
3. **LOCATION\_BACKGROUND**
   * **Description**: Allows the app to access location services when the app is running in the background.
   * **Use Case**: Similar to LOCATION\_ALWAYS, but explicitly requested when an app needs background location access for specific tasks like tracking delivery routes or providing location updates for security purposes.

**Q2: Code to Get the Current Location of User and Draw a Circle of Radius 100 m on That Point**

javascript

import React, { useEffect, useState } from 'react';

import { View, Text, StyleSheet, Platform, PermissionsAndroid, Alert } from 'react-native';

import Geolocation from 'react-native-geolocation-service';

import MapView, { Circle, Marker } from 'react-native-maps';

import { check, request, PERMISSIONS, RESULTS } from 'react-native-permissions';

const LocationCircleApp = () => {

const [location, setLocation] = useState(null);

const [locationPermission, setLocationPermission] = useState(false);

const [error, setError] = useState(null);

useEffect(() => {

requestLocationPermission();

}, []);

useEffect(() => {

if (locationPermission) {

getLocation();

}

}, [locationPermission]);

const requestLocationPermission = async () => {

if (Platform.OS === 'android') {

try {

const granted = await PermissionsAndroid.request(

PermissionsAndroid.PERMISSIONS.ACCESS\_FINE\_LOCATION,

{

title: 'Location Access Required',

message: 'This App needs to access your location',

buttonNeutral: 'Ask Me Later',

buttonNegative: 'Cancel',

buttonPositive: 'OK',

}

);

if (granted === PermissionsAndroid.RESULTS.GRANTED) {

setLocationPermission(true);

} else {

setError('Location permission denied');

}

} catch (err) {

console.warn(err);

}

} else {

checkAndRequestIOSPermission();

}

};

const checkAndRequestIOSPermission = async () => {

const permission = PERMISSIONS.IOS.LOCATION\_WHEN\_IN\_USE;

const result = await check(permission);

switch (result) {

case RESULTS.UNAVAILABLE:

setError('Location services are not available on this device');

break;

case RESULTS.DENIED:

const requestResult = await request(permission);

if (requestResult === RESULTS.GRANTED) {

setLocationPermission(true);

} else {

setError('Location permission denied');

}

break;

case RESULTS.GRANTED:

setLocationPermission(true);

break;

case RESULTS.BLOCKED:

Alert.alert(

'Location Permission Blocked',

'Please enable location services in settings',

[{ text: 'OK' }],

{ cancelable: false }

);

break;

}

};

const getLocation = () => {

Geolocation.getCurrentPosition(

position => {

const { latitude, longitude } = position.coords;

setLocation({ latitude, longitude });

},

error => {

console.error('Error getting current location:', error);

setError('Error getting current location');

},

{ enableHighAccuracy: true, timeout: 15000, maximumAge: 10000 }

);

};

return (

<View style={styles.container}>

{location ? (

<MapView

style={styles.map}

initialRegion={{

latitude: location.latitude,

longitude: location.longitude,

latitudeDelta: 0.01,

longitudeDelta: 0.01,

}}

>

<Marker coordinate={location} />

<Circle

center={location}

radius={100}

strokeWidth={1}

strokeColor="rgba(0,0,255,0.5)"

fillColor="rgba(0,0,255,0.1)"

/>

</MapView>

) : (

<Text>Fetching your location...</Text>

)}

{error && <Text>{error}</Text>}

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

},

map: {

width: '100%',

height: '100%',

},

});

export default LocationCircleApp;

**Q3: Fetch Junior Students and Senior Faculty**

import React, { useEffect, useState } from 'react';

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

const StudentsFacultyApp = () => {

const [data, setData] = useState(null);

const [error, setError] = useState(null);

useEffect(() => {

fetchData();

}, []);

const fetchData = async () => {

try {

const response = await fetch('http://comsats.edu/students?student=junior&faculty=senior');

const result = await response.json();

setData(result);

} catch (error) {

console.error('Error fetching data:', error);

setError('Error fetching data');

}

};

return (

<ScrollView contentContainerStyle={styles.container}>

{error ? (

<Text>{error}</Text>

) : (

data && (

<View>

<View style={styles.card}>

<Text style={styles.header}>Junior Students</Text>

{data[0].students[0].junior.map((student, index) => (

<Text key={index} style={styles.item}>{student}</Text>

))}

</View>

<View style={styles.card}>

<Text style={styles.header}>Senior Faculty</Text>

{data[0].Faculty[1].Senior.map((faculty, index) => (

<Text key={index} style={styles.item}>{faculty}</Text>

))}

</View>

</View>

)

)}

</ScrollView>

);

};

const styles = StyleSheet.create({

container: {

padding: 16,

backgroundColor: '#f8f9fa',

flexGrow: 1,

},

card: {

backgroundColor: '#ffffff',

borderRadius: 8,

padding: 16,

shadowColor: '#000',

shadowOffset: { width: 0, height: 2 },

shadowOpacity: 0.25,

shadowRadius: 3.84,

elevation: 5,

marginBottom: 16,

},

header: {

fontSize: 18,

fontWeight: 'bold',

marginBottom: 8,

},

item: {

fontSize: 16,

marginBottom: 4,

},

});

export default StudentsFacultyApp;