



Mobile App Development 2

Study diary

Md Forkan Hossain || 1902394

Contents

1	Week exercises	3
2	Week excercises	7
3	Week exercises	9
4	Week exercises	11
5	Week exercises	13
6	Week exercises	17
7	Week exercises	20
8	Week exercises	23
9	Week exercises	24
10	Week exercises	26
	Final project	27
	Sources used with exercises	28

1 Week exercises

1.1 Written answers to questions

React Native:

A technique called "multiplatform mobile development" enables you to create a single mobile application that works flawlessly across several different operating systems. Some or even the entire source code for cross-platform applications may be shared. As a result, developers can produce and distribute mobile assets that are compatible with both Android and iOS without having to recode them for each platform separately.

Framework for Android. a collection of Java interfaces, classes, and other pre-compiled code from which programs are constructed. The Android SDK's Android APIs may be used to access parts of the framework that are available to the public. Other components of the framework are only accessible to OEMs via the system APIs of the Android SDK.

For both Mac and iOS apps, Apple's IDE (Integrated Development Environment) is called Xcode. The graphical tool you'll employ to create iOS apps is called Xcode. You can design, create, write code for, and debug an iOS app using the tools, compilers, and frameworks included with Xcode.

JavaScript is used by React Native. Especially ReactJS, a JavaScript package used to create user interfaces. Working with React Native is straightforward for web developers. The widespread use of JavaScript is another clear benefit.

Best React Native tools for developers:

Atom, Android, iOS, and Linux, Nuclide, Sublime Text, Visual Studio Code, Expo, ESLint, Flow, Jest

Create a simple React Native "Hello World" application

1. `import React, {Component} from 'react';`
2. `import {Platform, StyleSheet, Text, View} from 'react-native';`
3. `type Props = {};`
4. `export default class App extends Component<Props> {`

5. `render() {`
6. `return (`
7. `<View>`
8. `<Text>Hello World</Text>`

Document my React app:

1. Install dependencies. First, install the required dependencies `npm install --save-dev react-styleguidist`.
2. Start Server. You don't need to configure anything if you created your app with `create-react-app`.
3. Start Basic Documenting.
4. Add More Description.
5. Add Interactive Example.
6. 20 Essential Parts of Any Large Scale React App.

React Native pros and cons:

- Hot Reload and Fast Refresh.
- Code reuse and ready-to-use components.
- Large community.
- Relatively complex UI.
- Requires some experience.
- Rely a lot on third-party libraries.
- Faster to develop.

Flutter:

Flutter uses a reactive programming language called **Dart**, making development faster and easier than traditional methods.

Best Flutter App Development Tools

Android Studio, Visual Studio Code, Flutter SDK, Dartpad, Firebase, Vysor, Bitrise, Codemagic.

Tutorial for Flutter:

- Flutter Course for Beginners ([freeCodeCamp](https://freeCodeCamp.org))
- Hello Dart: Introduction to Programming (code.makery.ch)
- Your First Flutter App: An App from Scratch (raywenderlich.com)

Flutter Dart Documentation

1. Comments.
2. DON'T use block comments for documentation
3. Generating docs.
4. Doc comments.

5. DO use `///` doc comments to document members and types
6. DO start doc comments with a single-sentence summary.
7. DO separate the first sentence of a doc comment into its own paragraph.

The Pros & Cons of Using Flutter

- Flutter enables you to make instant changes in the app which is a god-sent when it comes to fixing bugs.
- Flutter-based apps are very smooth in their performance which makes for great UX.
- With a single code base, quality assurance and testing usually takes much less time.

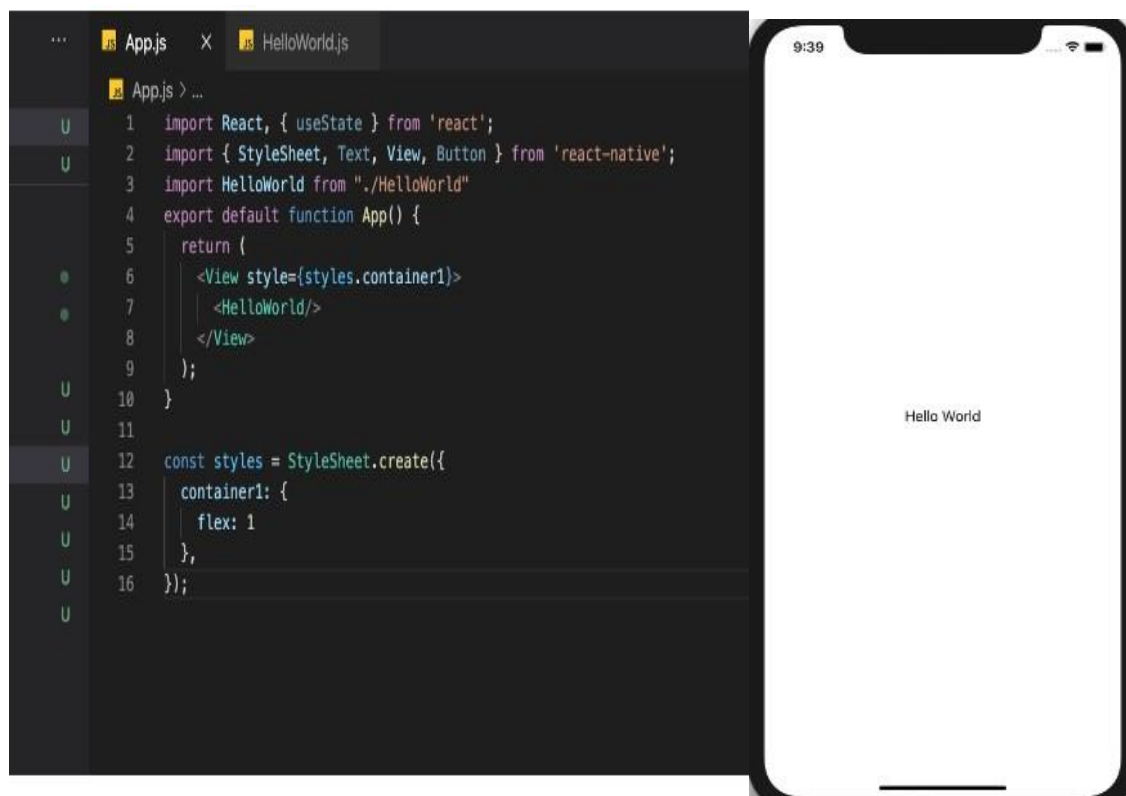
1.2 Android Studio setup and Hello World

Steps to Install React Native:

1. Install Chocolatey.
2. Install Node, Python2, and JDK8.
3. Install Android Studio.
4. Setup Android SDK.
5. Install React Native CLI.
6. Setting Up Android Device.

Install and Configure Flutter:

1. Download Flutter SDK.
2. Extract the Files.
3. Update Path Variable for Windows PowerShell.
4. Confirm Installed Tools for Running Flutter.
5. Download and Install Android Studio.



```

1  import * as React from 'react';
2  import { Text, View, StyleSheet } from 'react-native';
3  import Constants from 'expo-constants';
4
5  // You can import from local files
6  import AssetExample from './components/AssetExample';
7
8  // or any pure javascript modules available in npm
9  import { Card } from 'react-native-paper';
10
11 export default function App() {
12   return (
13     <Text style={styles.app}>
14       Forkan Hossain
15     </Text>
16   );
17 }
18
19 const styles = StyleSheet.create({
20   app: {
21     textAlign: 'center',
22     marginTop: 200,
23     fontSize: 20,
24   },
25 });

```

My Device iOS Android Web

Forkan Hossain

```

main.dart X
lib > main.dart > MyApp > build
1  import 'package:flutter/material.dart';
2
3  Run | Debug | Profile
4  void main() {
5    runApp(const MyApp());
6  }
7
8  class MyApp extends StatelessWidget {
9    const MyApp({Key? key}) : super(key: key);
10
11    @override
12    Widget build(BuildContext context) {
13      return const MaterialApp(
14        home: Center(child: Text('Forkan Hossain')),
15      ); // MaterialApp
16    }
17  }

```


flutter_application_0001

debug





Forkan Hossain


2 Week exercises

1 st Exercise

ashamed scone  All changes saved half a minute ago [See previous saves](#) ✓

Search API

Saved    

My Device iOS Android **Web** 

Open files
 App.js
 Project
 assets
 components
 App.js
 package.json
 README.md

```

1  import { Text, View, StyleSheet, TextInput, Button, } from 'react-native';
2  import Constants from 'expo-constants';
3  import React, {useState} from 'react';
4  const App=()=>{
5    const [Number1, setnumber1]=useState();
6    const [output, setResult]=useState();
7    function click(){
8      setResult((Number1 * 9/5) + 32+" °F");
9    }
10
11   return( <View>
12     <Text style={styles.viewer}>Weather Station</Text>
13     <TextInput style={styles.textInputStyle} placeholder="Enter temperature in C.." keyboardType="numeric" value =
14       {Number1} onChange={e=>setnumber1(+e.target.value)}></TextInput>
15     <Button title="Calculate in F" onPress={()=>click()}></Button>
16     <h2 style={styles.clicked}>{output}</h2>
17   </View>
18 );
19
20 const styles = StyleSheet.create(
21   {
22     textInputStyle:{
23       height: 50,
24       borderWidth: 4,
25       margin: 20,
26       padding: 15,
27       borderRadius: 10,
28       borderColor: 'Yellow',
29     },
30     viewer:{
31       padding: 80,
32       fontSize: 60,


```

Weather Station





CALCULATE IN F


✓ No errors

Prettier {} Editor Expo v46.0.0 Devices 1 Preview

ashamed scone  All changes saved 2 minutes ago [See previous saves](#) ✓

Search API

Saved    

My Device iOS Android **Web** 

Open files
 App.js
 Project
 assets
 components
 App.js
 package.json
 README.md

```

1  import { Text, View, StyleSheet, TextInput, Button, } from 'react-native';
2  import Constants from 'expo-constants';
3  import React, {useState} from 'react';
4  const App=()=>{
5    const [Number1, setnumber1]=useState();
6    const [output, setResult]=useState();
7    function click(){
8      setResult((Number1 * 9/5) + 32+" °F");
9    }
10
11   return( <View>
12     <Text style={styles.viewer}>Weather Station</Text>
13     <TextInput style={styles.textInputStyle} placeholder="Enter temperature in C.." keyboardType="numeric" value =
14       {Number1} onChange={e=>setnumber1(+e.target.value)}></TextInput>
15     <Button title="Calculate in F" onPress={()=>click()}></Button>
16     <h2 style={styles.clicked}>{output}</h2>
17   </View>
18 );
19
20 const styles = StyleSheet.create(
21   {
22     textInputStyle:{
23       height: 50,
24       borderWidth: 4,
25       margin: 20,
26       padding: 15,
27       borderRadius: 10,
28       borderColor: 'Yellow',
29     },
30     viewer:{
31       padding: 80,
32       fontSize: 60,

```


Weather Station

CALCULATE IN F

86 °F

✓ No errors

Prettier {} Editor Expo v46.0.0 Devices 1 Preview







ludicrous popsicle

All changes saved less than 20 seconds ago. [See previous saves](#) ✓

Q Search API

Saved



Open files

App.js

Header.js

WeatherScreen.js

package.json

Project

assets

download.jpg

snack-icon.png

components

Header.js

WeatherScreen.js

App.js


package.json

README.md

```
1 import {useState} from 'react';
2 import {View, Text, StyleSheet, Button, TextInput, Image, SafeAreaView} from "react-native";
3
4 import WeatherScreen from './components/WeatherScreen.js';
5
6 const App=()=>{
7   //const[hellomessage, sethellomessage] = useState("HelloWorld");
8
9   return(
10     <WeatherScreen></WeatherScreen>
11   );
12 };
13
14 export default App;
15
```

My Device iOS Android Web

Tampere Weather Station



7°C

20 mph

FIND

3 Week Exercise

1 st Exercise

```

1 // Md Forkan Hossain
2 import {useState} from 'react';
3 import {View, Text, StyleSheet, Button, TextInput, Image, SafeAreaView} from "react-native";
4 import Header from './components/Header.js';
5 import WeatherInfo from './components/WeatherInfo';
6 const WeatherScreen={()=>{
7
8   // Let helloMessage = "Hello World";
9   const[weatherData, setWeatherData]= useState({
10     //Location : '',
11     city : 'Tampere',
12     description: 'Sunny',
13     temperature: 15,
14     windSpeed: 3,
15     weatherIcon: "01d",
16   });
17   const url = 'https://api.openweathermap.org/data/2.5/weather?q=';
18   const apiKey = '&units=metric&appid=af604235e547bff3b148acbebe1c0a16'
19   //const iconImage = 'https://openweathermap.org/image/w/';
20   const fetchWeatherData = async (location)=>{
21     try{
22       //const response = await fetch(url + location + apiKey);
23       const response = await fetch(url + "Delhi" + apiKey);
24       const jsonWeatherObject = await response.json();
25       setWeatherData({
26         city: jsonWeatherObject.name,
27         description: jsonWeatherObject.weather[0].description,
28         temperature: jsonWeatherObject.main.temp,
29         windSpeed: jsonWeatherObject.wind.speed,
30         weatherIcon: jsonWeatherObject.weather[0].icon,
31       });
32     } catch(err){
33       console.error(error);
34     }
35   }
36
37   //const response = await fetch(url + location + apiKey);
38   const response = await fetch(url + "Delhi" + apiKey);
39   const jsonWeatherObject = await response.json();
40   setWeatherData({
41     city: jsonWeatherObject.name,
42     description: jsonWeatherObject.weather[0].description,
43     temperature: jsonWeatherObject.main.temp,
44     windSpeed: jsonWeatherObject.wind.speed,
45     weatherIcon: jsonWeatherObject.weather[0].icon,
46   });
47   } catch(err){
48     console.error(error);
49   }
50   }
51   return(
52     <SafeAreaView>
53     <Header headerText = {weatherData.city}></Header>
54     <WeatherInfo description =(weatherData.description)
55       temperature = {weatherData.temperature}
56       windSpeed =(weatherData.windSpeed)
57       weatherIcon ={weatherData.weatherIcon}
58     ></WeatherInfo>
59     <Button title={"Fetch weather of " + weatherData.city} onPress= {(())=>fetchWeatherData()}></Button>
60   </SafeAreaView>
61   );
62 };
63 const styles = StyleSheet.create(
64   {}
65 );
66 export default WeatherScreen;
67

```

My Device
iOS
Android
Web

Tampere

15 °C

Sunny
3 mph

FETCH WEATHER OF TAMPERE

My Device
iOS
Android
Web

Delhi

32.05 °C

haze
3.6 mph

FETCH WEATHER OF DELHI

Prettier {} Editor Expo v46.0.0
Devices 1 Preview

ENG
11:06 AM

2nd Exercise

```

1 import {useState} from 'react';
2 import {View, Text, StyleSheet, Button, TextInput, Image, SafeAreaView} from "react-native";
3 import Header from './Header'
4 import WeatherInfo from './WeatherInfo';
5 import UpdateLocation from './UpdateLocation';
6
7 const WeatherScreen={()=>{
8
9   const [weatherData, setWeatherData]= useState({
10     //location : '',
11     city : 'Tampere',
12     description: 'Sunny',
13     temperature: 15,
14     windSpeed: 3,
15     weatherIcon: "01d",
16   });
17
18   const url = 'https://api.openweathermap.org/data/2.5/weather?q=';
19   const apiKey = '&units=metric&appid=372f58cdd4578e431749980646541770'
20   const fetchWeatherData = async (location)=>{
21     try{
22       const response = await fetch(url + location + apiKey);
23       const jsonWeatherObject = await response.json();
24       setWeatherData({
25         city: jsonWeatherObject.name,
26         description: jsonWeatherObject.weather[0].description,
27         temperature: jsonWeatherObject.main.temp,
28         windSpeed: jsonWeatherObject.wind.speed,
29         weatherIcon: jsonWeatherObject.weather[0].icon,
30       });
31     }
32     catch(err){
33       console.error(error);
34     }
35   }
36 }

```

My Device iOS Android **Web** 

Tampere

-15 °C



Sunny
3 mph

UPDATE

```

1 import {useState} from 'react';
2
3 import {View, Text, StyleSheet, Button, TextInput, Image, SafeAreaView} from "react-native";
4
5 import WeatherScreen from './components/WeatherScreen';
6
7 const App={()=>{
8
9   //const [helloMessage, setHelloMessage] = useState("HelloWorld");
10
11   return(
12     <WeatherScreen/>
13   );
14 };
15
16 export default App;

```

My Device iOS Android **Web** 

Delli

31.92 °C



overcast clouds
4.18 mph

UPDATE

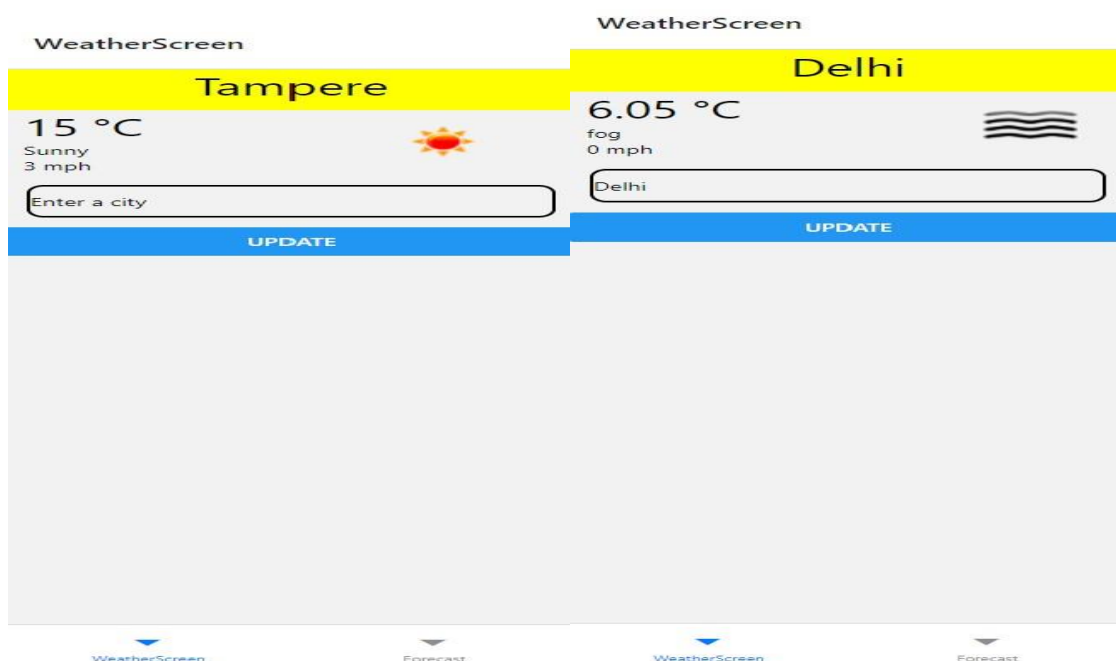
4 Week Exercise

1st Exercise:

When I want to navigate between pages in my React application, the go-to choice is **React Router**. If I am not building a single page application, I can use the `<a>` element or `Window.location` object to navigate to other pages.

React Navigation is the most popular navigation library in the React Native ecosystem and the best choice for most apps. It is maintained by the Expo team and supports Android, iOS, and the web.

In here, I used bottom tabs and 2 screens.



```

20 //const Tab = createMaterialBottomTabNavigator();
21 //const Tab = createMaterialTopTabNavigator();
22 //const Tab = createStackNavigator();
23 const Tab = createBottomTabNavigator();
24
25
26
27
28 const App = () => {
29   return (
30     <NavigationContainer>
31       <Tab.Navigator>
32         <Tab.Screen name="WeatherScreen" component={WeatherScreen}/>
33         <Tab.Screen name="Forecast" component={ForecastScreen}/>
34       </Tab.Navigator>
35     </NavigationContainer>
36   );
37 };
38
39 export default App;
40

```

2nd Exercise:

2 screen: weather and forecast



5 Week exercises

Platform Specific components in React Native

Name of some components and descriptions in Android:

- ✓ Fragments Represents a portion of user interface in an Activity.
- ✓ Intents Messages wiring components together.
- ✓ Views UI elements that are drawn on-screen including buttons, lists forms etc.
- ✓ Toast Android

```
const showToast = () => {
  if (Platform.OS == 'android') {
    console.log('Welcome Android World');
    ToastAndroid.show('Fetching data', ToastAndroid.LONG);
  } else if (Platform.OS == 'ios') {
    console.log('Welcome ios World');
  } else {
    console.log('Running else where');
  }
};

const createTwoButtonAlert = () =>
  Alert.alert('Alert Title', 'This is forecast', [
    {
      text: 'Cancel',
      onPress: () => console.log('Cancel Pressed'),
      style: 'cancel',
    },
    { text: 'OK', onPress: () => console.log('OK Pressed') },
  ]);

return(
  <SafeAreaView style={styles.container}>
    <Header headerText = {weatherData.city}></Header>
    <WeatherInfo description = {weatherData.description}
      temperature = {weatherData.temperature}
      windSpeed = {weatherData.windSpeed}
      weatherIcon = {weatherData.weatherIcon}
    </WeatherInfo>
  </SafeAreaView>
);
```

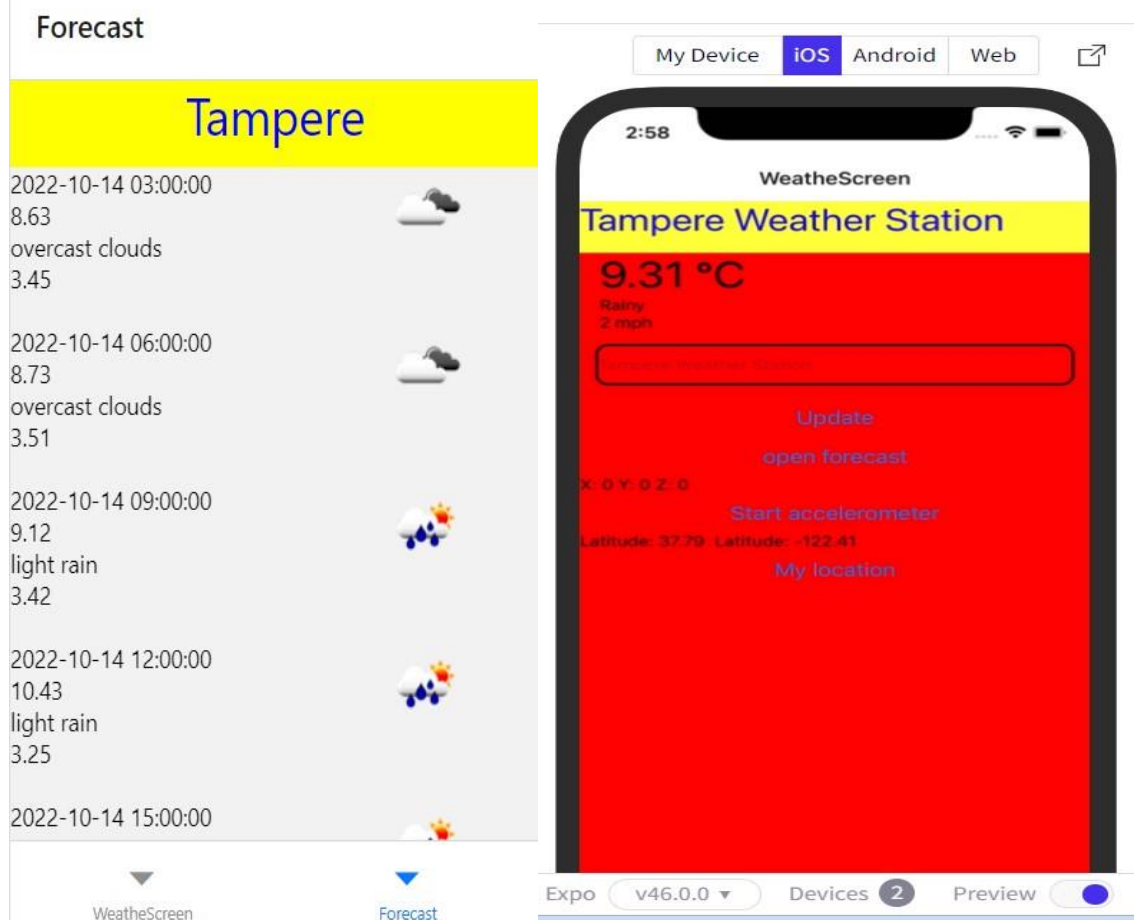
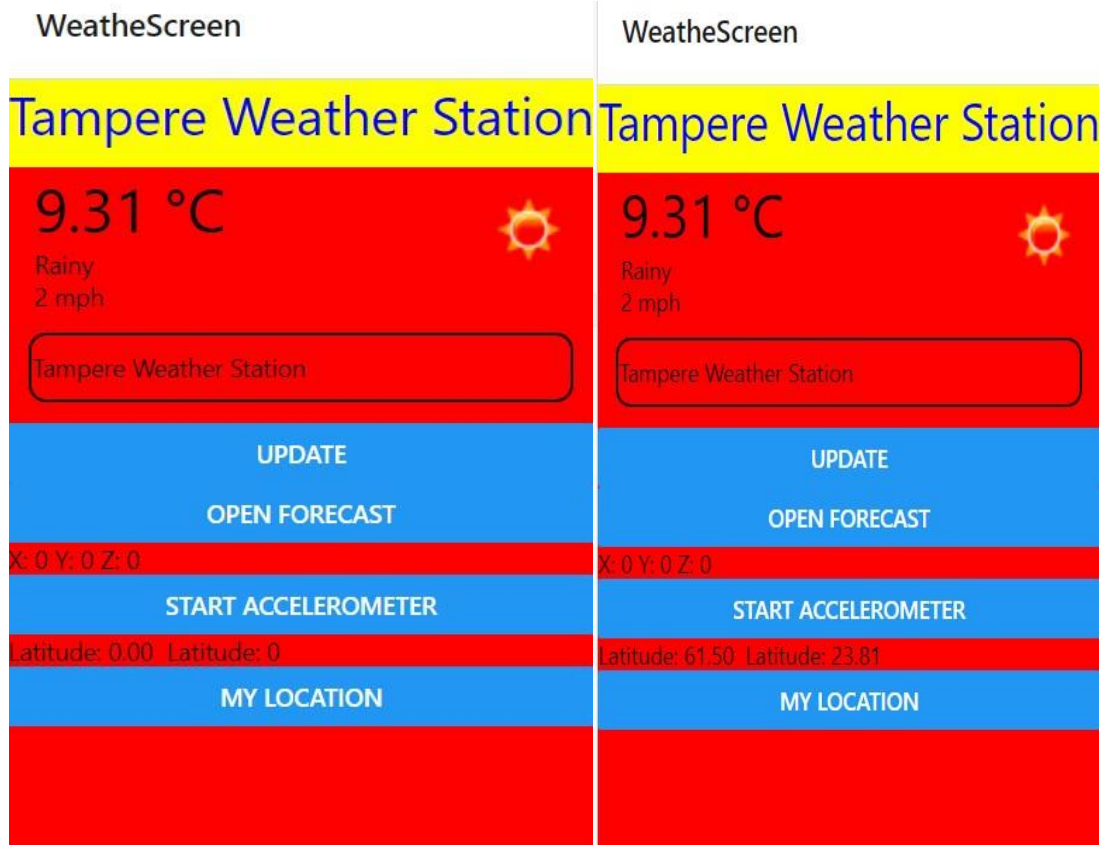
Name of some components and descriptions in Android:

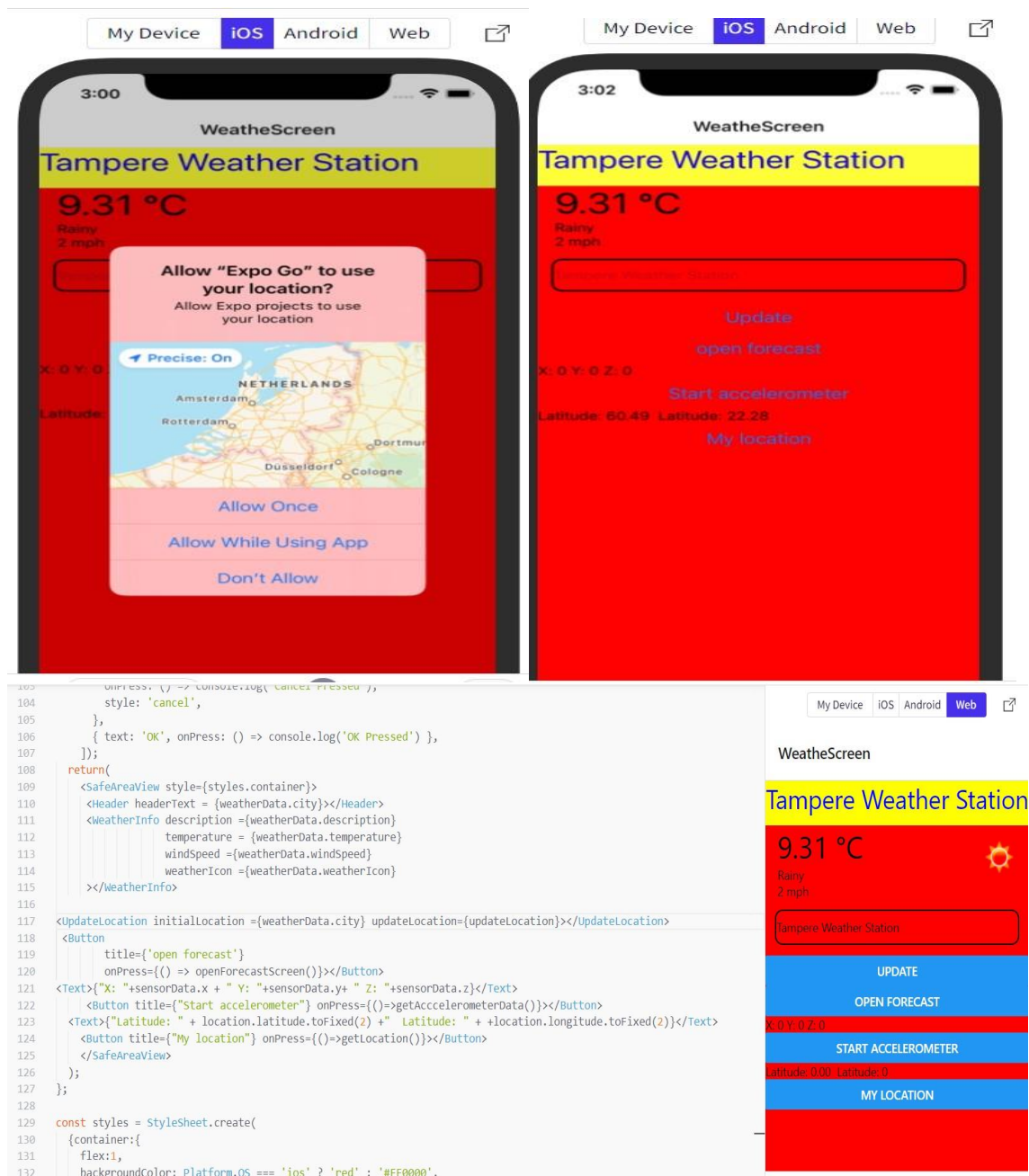
- ✓ Navigation Bar. The Navigation Bar is used for quick information about the screen.
- ✓ Search Bar.
- ✓ Toolbar.
- ✓ Tab Bar.
- ✓ Sidebar.
- ✓ Collection View.
- ✓ Activity View.
- ✓ Alerts.

```
const createTwoButtonAlert = () =>
  Alert.alert('Alert Title', 'This is forecast', [
    {
      text: 'Cancel',
      onPress: () => console.log('Cancel Pressed'),
      style: 'cancel',
    },
    { text: 'OK', onPress: () => console.log('OK Pressed') },
  ]);

return(
  <SafeAreaView style={styles.container}>
    <Header headerText = {weatherData.city}></Header>
    <WeatherInfo description = {weatherData.description}
      temperature = {weatherData.temperature}
      windSpeed = {weatherData.windSpeed}
      weatherIcon = {weatherData.weatherIcon}
    </WeatherInfo>
  </SafeAreaView>
);
```

Utilizing Device APIs in React Native





```

const getLocation=async()->{
  try {
    if (Platform.OS === 'android' && !Device.isDevice) {
      setErrorMsg(
        'Oops, this will not work on Snack in an Android Emulator. Try it on your device!'
      );
      return;
    }
    let { status } = await Location.requestForegroundPermissionsAsync();
    if (status !== 'granted') {
      setErrorMsg('Permission to access location was denied');
      return;
    }
    const location = await Location.getCurrentPositionAsync({});
    setLocation({
      latitude: location.coords.latitude,
      longitude: location.coords.longitude,
    });
  } catch (err) {

```

Publishing React Native Apps

Explain the steps of publishing your app in the Play Store (or Apple Store)

I have learned from your given course materials and internet that as an open platform, Android offers different choices to distribute apps. Even though other alternative Stores exist, Google Play is the android app store, the main platform to distribute an Android app.

Step-by-Step Process to Upload App To Google Play Store

1. Google Play Developer Console.
2. Link Developer Account with Google Wallet Merchant Account.
3. Create Application.
4. App Store Listing.
5. Upload App Bundles or APK To Google Play.
6. Time For Content Rating.
7. Fix App Pricing and Distribution.
8. Finally, Publish the Application.

How can you become a publisher of Mobile Apps in the stores, explain the steps.

In mobile marketing, a publisher provides the capability and inventory that allows advertisers to run ads in their apps or on mobile sites. This can mean a publisher can be a website or an app.

A four-year bachelor's degree is a prerequisite for becoming a publisher. It would be best to major in English, journalism, communications, writing, or one of the liberal arts. Even while it's not required, prospective employers might find you more appealing if you have a master's or other postgraduate degree in publishing.

Which tests do you have to perform with your application before publishing? How it will be tested

I can test my Android app before publishing to the Google Play Store using the following methods:

1. Local Unit Tests using JUnit.
2. Instrumented Unit Tests using JUnit.
3. User Interface Tests using Espresso.
4. Distributing your app for Open, Closed or Internal Testing on the Google Play Store.

Knowing about the different types of mobile testing would be the first step towards formulating a comprehensive QA strategy.

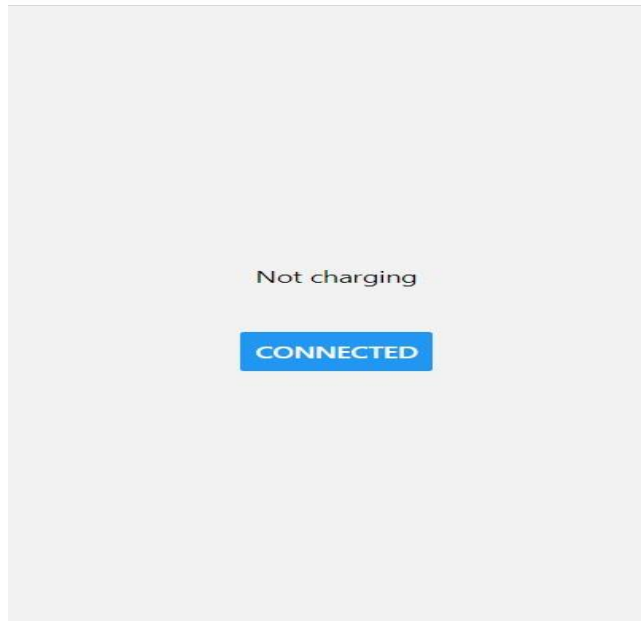
- Functional Testing.
- Interruption Testing.
- Localization Testing.
- Speed Testing.
- Memory Leak Testing.
- Usability Testing.
- Performance Testing.
- Security Testing.

6 Week exercises

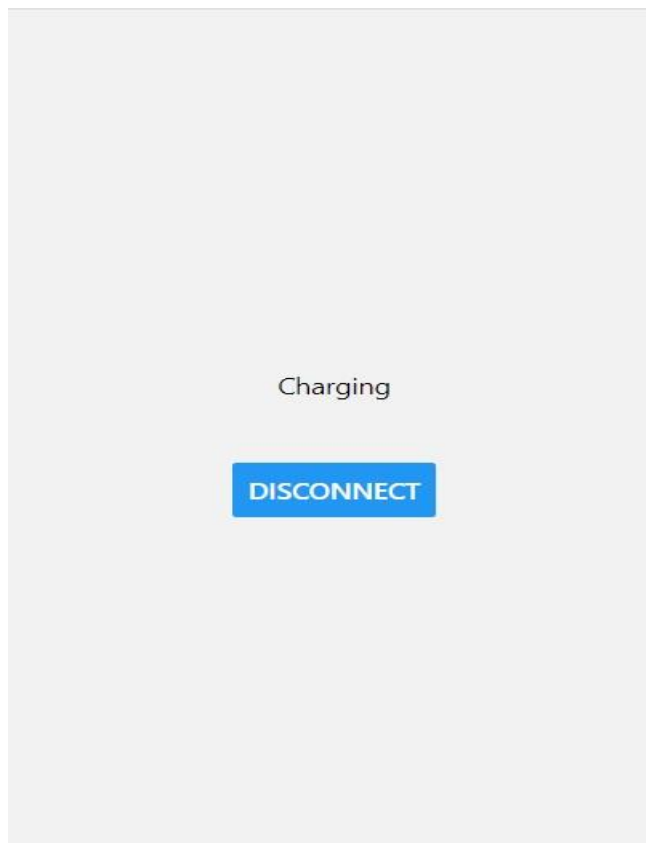
1. Android Essentials Recap (before the classes)

Android Enterprise Essentials is a lighter, leaner secure management service from Google. Designed to make it easier for company to protect and manage its mobile devices.

two



← one



2. Broadcast receivers in general (written Q&A)

Broadcast in android is the system-wide events that can occur when the device starts, when a message is received on the device or when incoming calls are received, or when a device goes to airplane mode, etc

Android Call State BroadcastReceiver Example

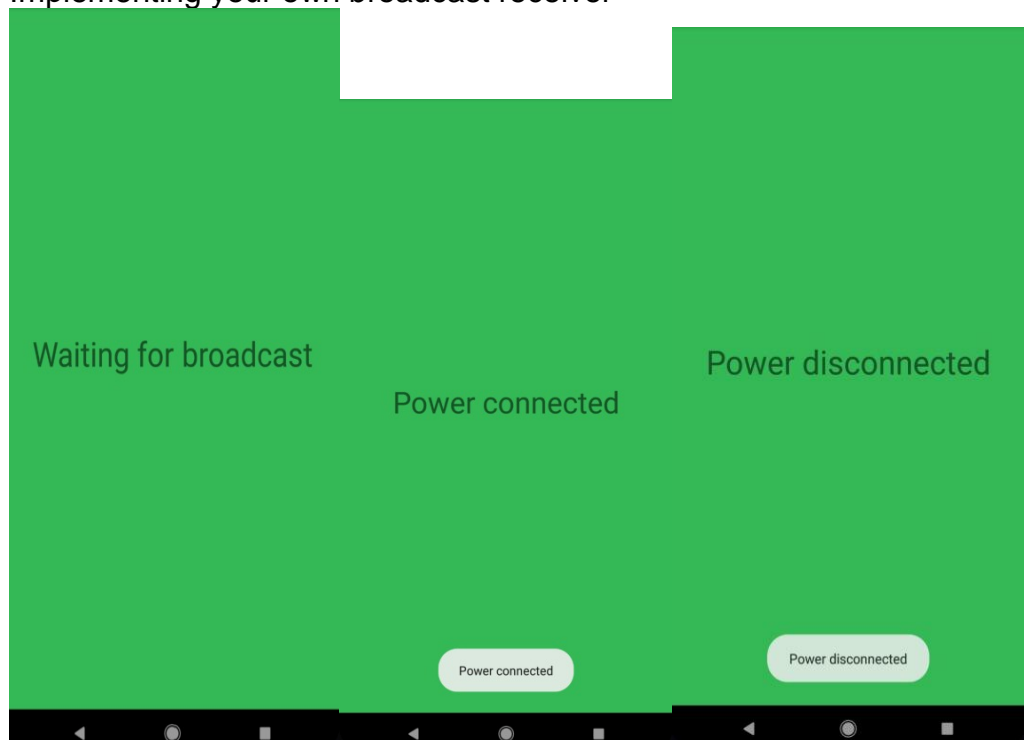
1. `android:layout_width="match_parent"`
2. `android:layout_height="match_parent"`
3. `android:paddingBottom="@dimen/activity_vertical_margin"`
4. `android:paddingLeft="@dimen/activity_horizontal_margin"`
5. `android:paddingRight="@dimen/activity_horizontal_margin"`

There are **two types of broadcast receivers**: Static receivers, which you register in the Android manifest file. Dynamic receivers, which you register using a context.

Android provides **three** ways for apps to send broadcast: The `sendOrderedBroadcast(Intent, String)` method sends broadcasts to one receiver at a time.

Also, The term 'broadcast media' covers a wide spectrum of different communication methods such as **television, radio, newspapers, magazines** and any other materials supplied by the media and press.

3. Implementing your own broadcast receiver



Power disconnected

Power disconnected

plug in

Unplugged

```
private class HeadsetPlugging extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        final String action = intent.getAction();
        if (Intent.ACTION_HEADSET_PLUG.equals(action)) {
            if (intent.getIntExtra("state", -1) == 0) {
                Toast.makeText(context, "Unplugged", Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(context, "plug in", Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

7 Week exercises

1. Services – written Questions and Answers

A component built specifically to do tasks without a user interface is known as an Android service. A service might play music, download a file, or filter a picture. Interprocess communication (IPC) between Android applications may also be done via services.

An Android service is a component that is used to carry out background tasks including managing network transactions, playing music, connecting with content providers, etc. It is devoid of any UI (user interface). Even if the application is deleted, the service continues to execute in the background.

Types of Android Services

- Foreground Services:
- Background Services:
- Bound Services:

Services are diversified in three groups: **Business services, social services and personal services.**

The Life Cycle of Android Services:

In android, services have 2 possible paths to complete its life cycle namely Started and Bounded.

1. Started Service (Unbounded Service):

By following this path, a service will initiate when an application component calls the **startService()** method. Once initiated, the service can run continuously in the background even if the component is destroyed which was responsible for the start of the service. Two options are available to stop the execution of service:

- By calling **stopService()** method,
- The service can stop itself by using **stopSelf()** method.

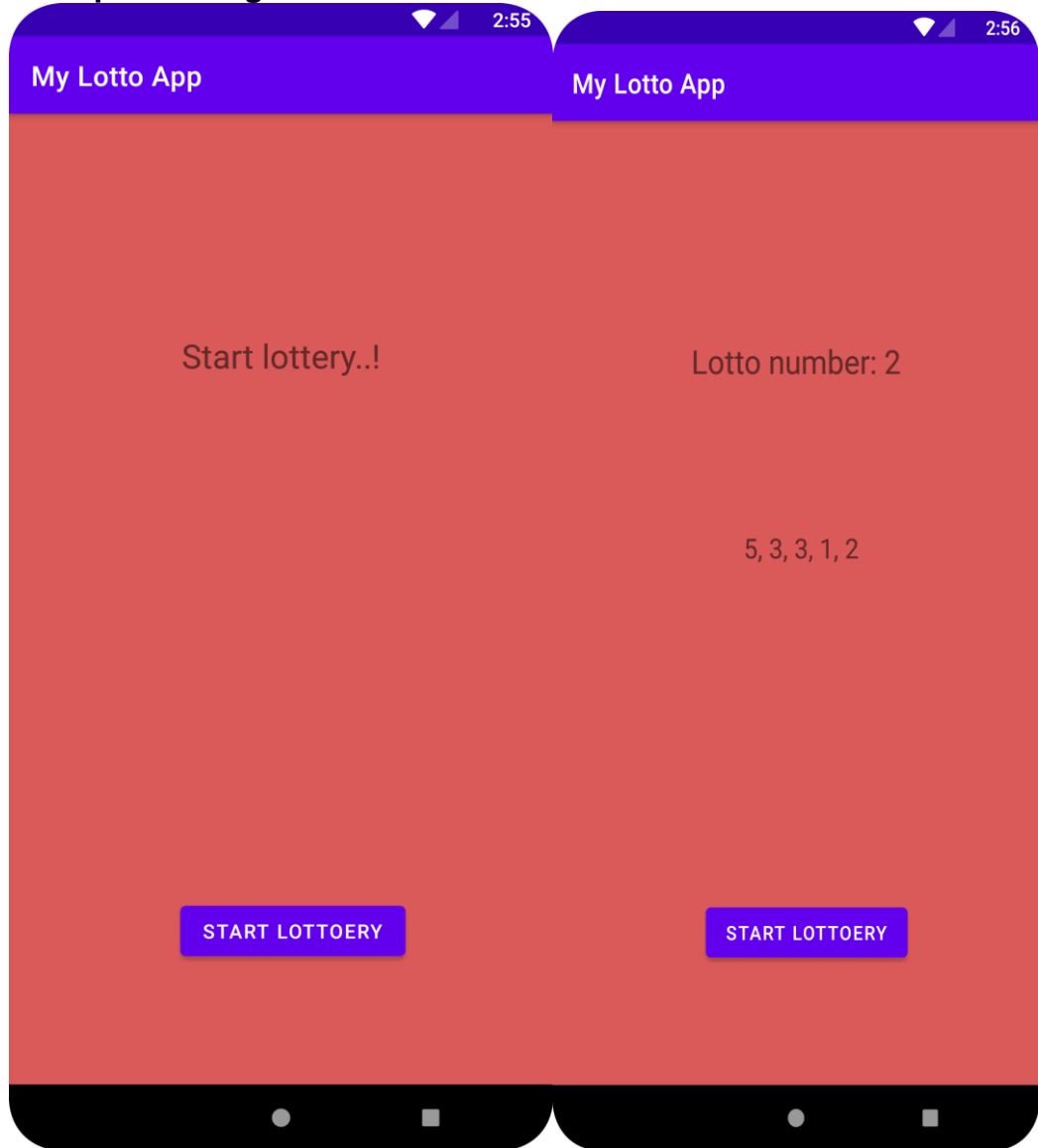
2. Bounded Service:

It can be treated as a server in a client-server interface. By following this path, android application components can send requests to the service and can fetch results. A service is termed as bounded when an application component binds itself with a service by calling **bindService()** method. To stop the execution of this service, all the components must unbind themselves from the service by using **unbindService()** method.

A regular class that extends the class Service can be used to establish a user-defined service. Furthermore, there are several callback methods that must be altered in order to perform service activities on apps. A few of the crucial Android Services methods are as follows:

The broadcast message itself is wrapped in an intent object whose action string identifies the event that occurred (for example `android.intent.action.AIRPLANE_MODE`).

2. Implementing a Service



```

26 }
27
28 public void StartLottoService(View view) {
29     Intent intent = new Intent( packageContext, this, MyLottoService.class);
30     intent.putExtra( name: "LOTTO_NUMBER_AMOUNT", value: 5);
31     startService(intent);
32     Handler handler = new Handler();
33     handler.postDelayed(new Runnable() {
34         @Override
35         public void run() {
36             TextView textView2 = findViewById(R.id.textView2);
37             textView2.setText(five + ", " + four + ", " + third + ", " + second + ", " + first);
38         }
39     }, delayMillis: 12000);
40 }
41
42 private class myLottoBroadcastReceiver extends BroadcastReceiver {
43     @Override
44     public void onReceive(Context context, Intent intent) {
45         if(intent.getAction() == "com.tam1.mylotteryapp"){
46             int number = intent.getIntExtra( name: "LOTTO_NUMBER", defaultValue: 0);
47             TextView maintextView = findViewById(R.id.maintextView);
48             maintextView.setText("Lotto number: " + number);
49             if(number!=0){
50                 five = four;
51                 four = third;
52                 third = second;
53                 second = first;
54                 first = number;
55             }
56         }
57     }
58 }

```

```

public int onStartCommand(Intent intent, int flags, int startId){
    mLottoNumberAmount = intent.getIntExtra( name: "LOTTO_NUMBER_AMOUNT",   defaultValue: 0);

    new Thread()->{
        try {
            for(int i=0; i<mLottoNumberAmount; i++){
                Thread.sleep( millis: 1000);
                // now i will make code to generate lotto number and send and broadcast
                Random rand = new Random();
                int rand_int1 = rand.nextInt( bound: 9);
                int number = rand_int1;
                Intent lottoNumber = new Intent( action: "com.taml.mylotteryapp");
                lottoNumber.putExtra( name: "LOTTO_NUMBER", number);
                sendBroadcast(lottoNumber);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }).start();
    return START_STICKY;
}

@Override


```

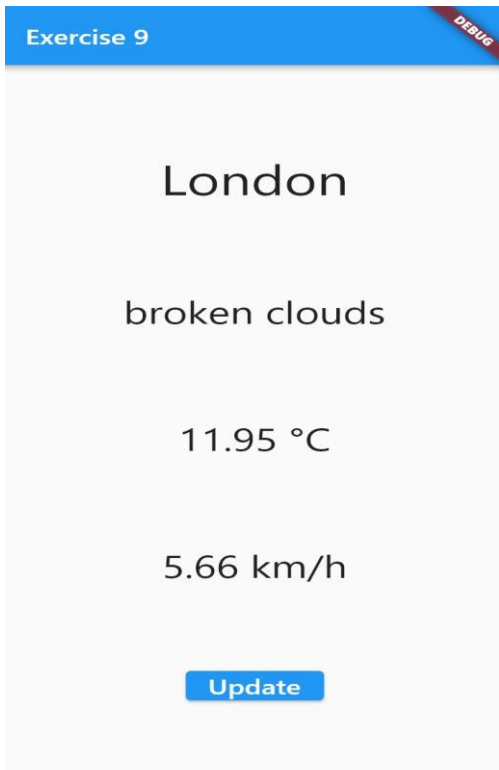
8 Week exercises

9 Week exercises

Flutter and REST APIs

Flutter allows us to create UIs and also combine them with backends. To show user data, the majority of applications employ APIs. We'll make use of the HTTP package, which offers sophisticated ways to carry out activities. Simple http requests are used by REST API to connect with JSON data because: Utilizing await and async functions.





```

void fetchWeather() async {
  Uri uri = Uri.parse(
    'https://api.openweathermap.org/data/2.5/weather?q=london&units=metric&appid=26bcb6dfb4cf05b2e96f6791e362e83a');
  var response = await http.get(uri);
  if (response.statusCode == 200) {
    var weatherData = json.decode(response.body);
    setState(() {
      cityName = weatherData["name"];
      description = weatherData["weather"][0]["description"];
      temperature = weatherData["main"]["temp"];
      wind = weatherData["wind"]["speed"];
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Exercise 9"),
    ), // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Text(cityName, style: const TextStyle(fontSize: 40)),
          Text(description, style: const TextStyle(fontSize: 16)),
          Text(temperature, style: const TextStyle(fontSize: 24)),
          Text(wind, style: const TextStyle(fontSize: 24)),
          Text("Update", style: const TextStyle(fontSize: 16, color: white, backgroundColor: blue)),
        ],
      ),
    ),
  );
}

```

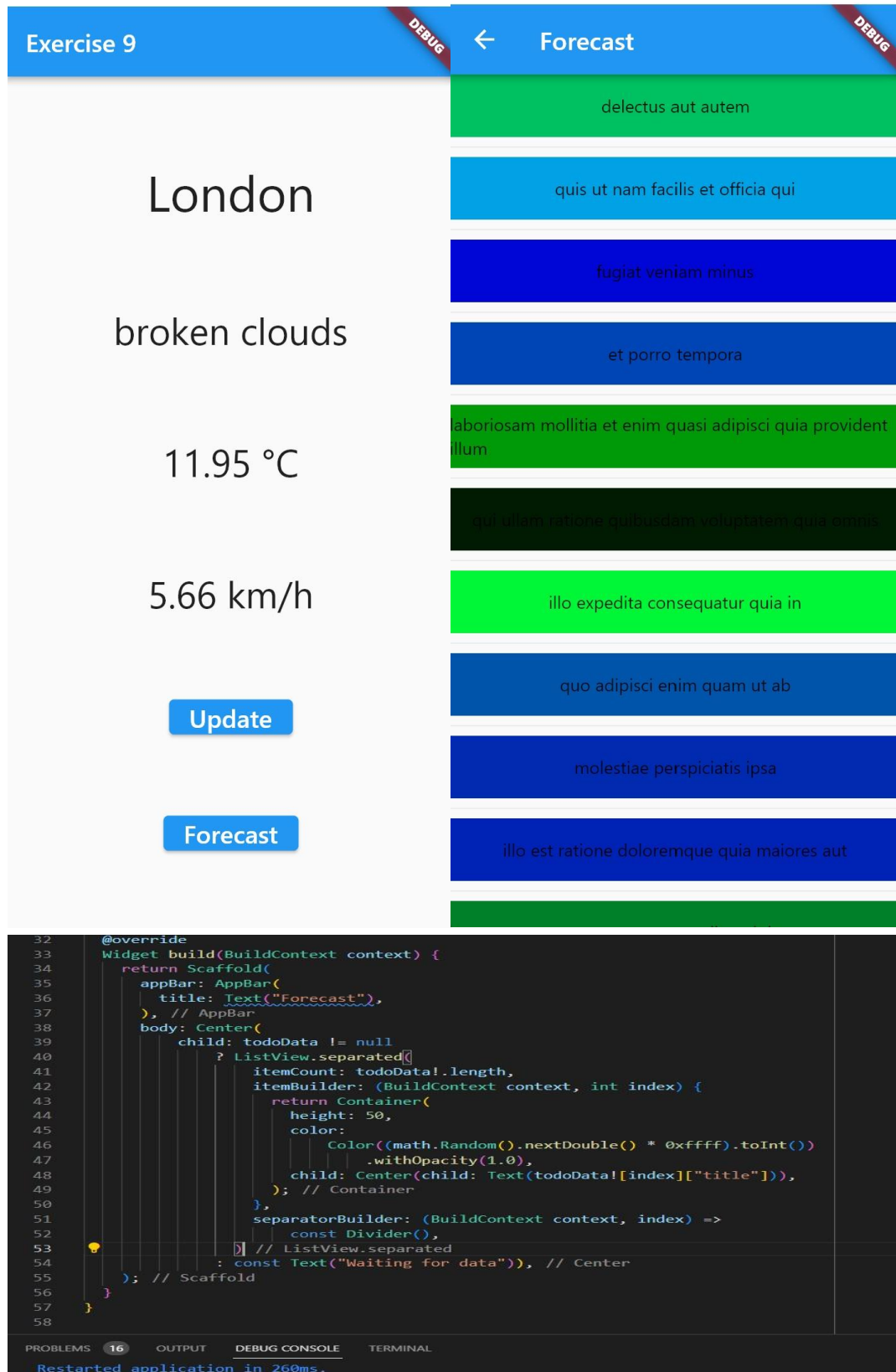
17 OUTPUT DEBUG CONSOLE TERMINAL

Started application in 260ms.

Flutter and ListView component

The constructor of ListView.builder:

1. Create a new flutter application.
2. For now, delete the code from the main. dart.
3. Copy the below code and paste it into your main. dart.

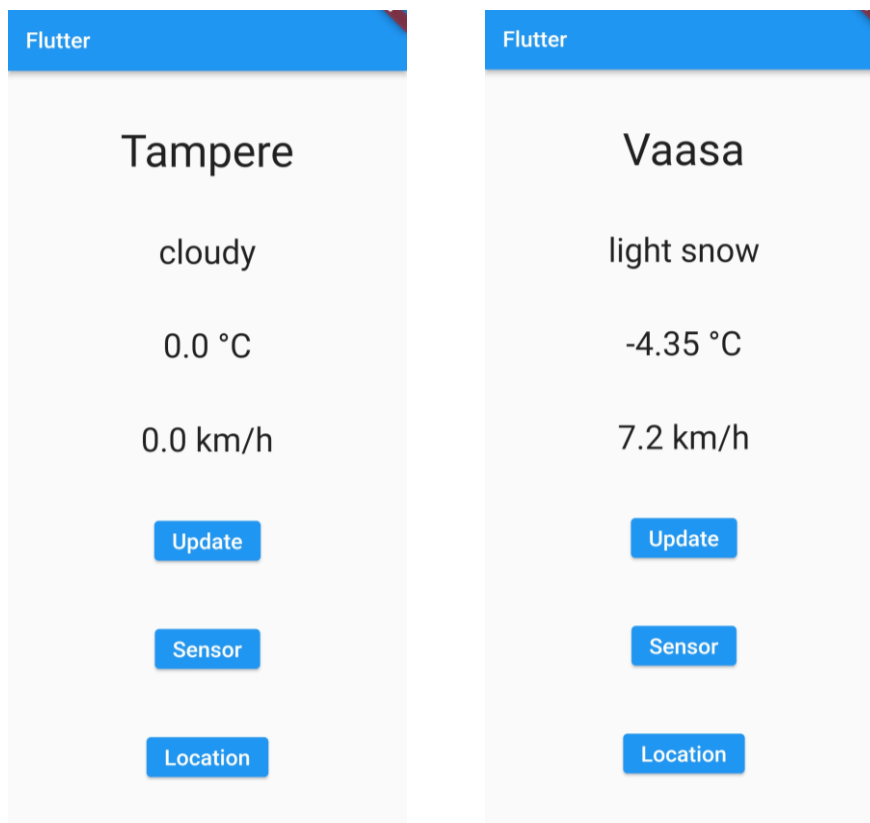


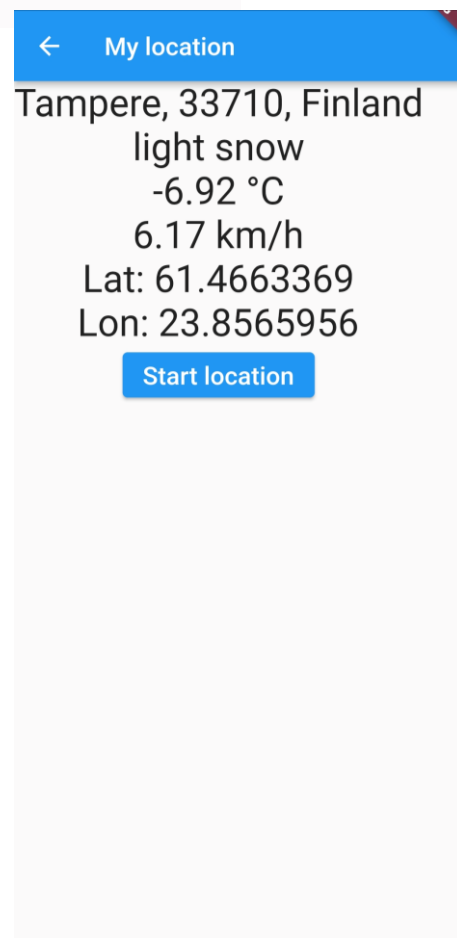
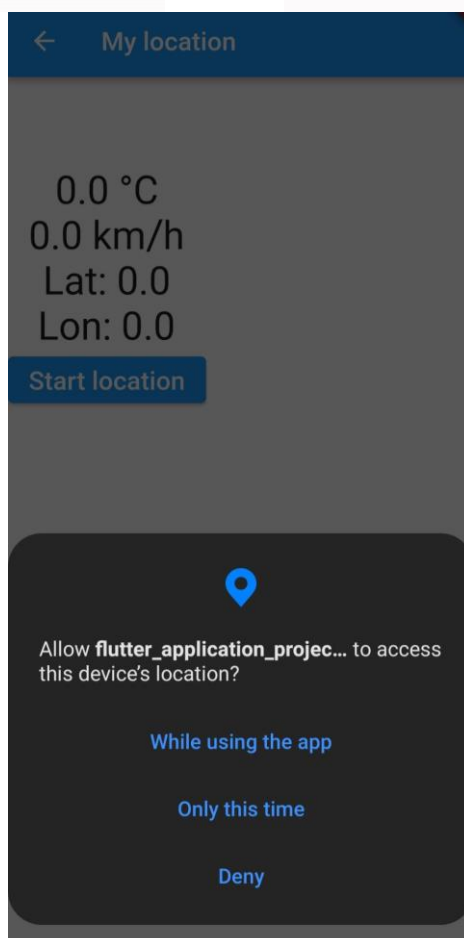
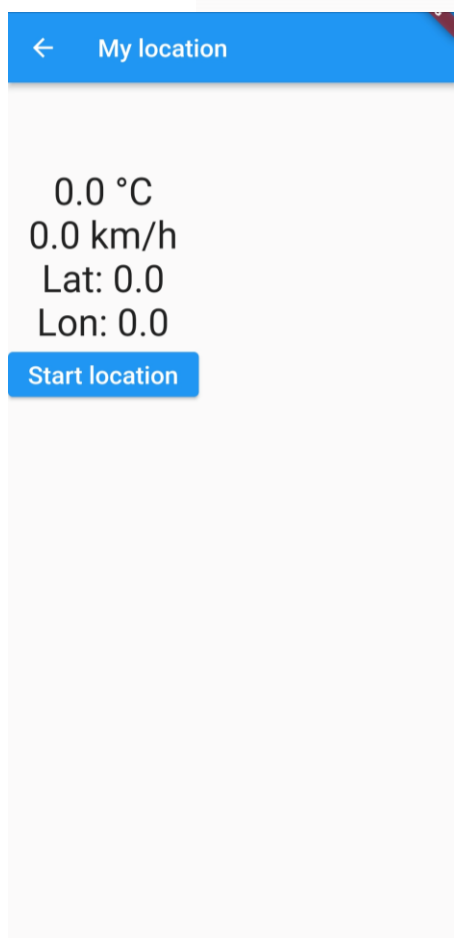
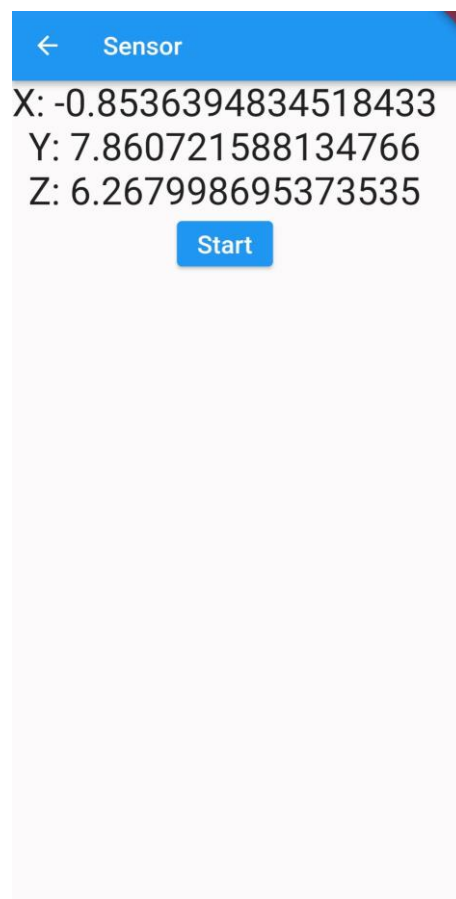
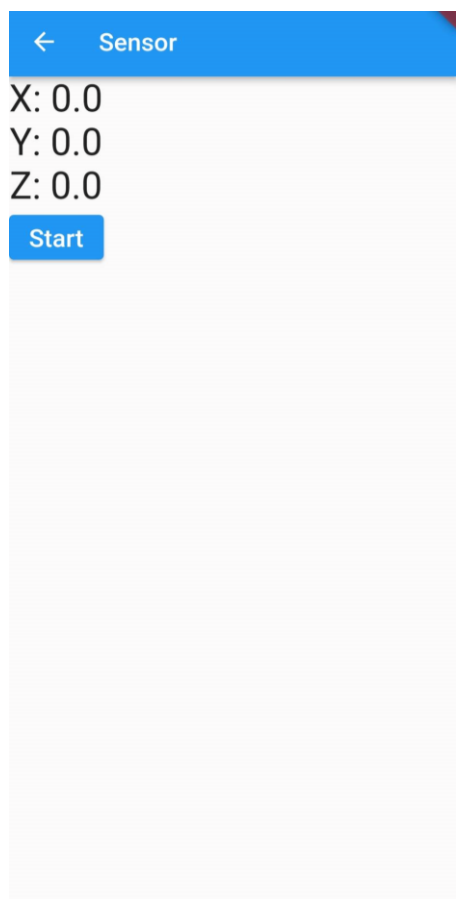
10 Week exercises

Final project

For my final project, I have chosen making a react native application (option 1). So, I have made a weather app using react native.

The main use case is, when the user press button update, front page will show the current weather update of city Vaasa. I have used async/await method to fetch the weather data from a web service. Also, there is total 3 screens in the app. So, I have used navigation here. After that when I go to sensor screen, I can have the accelerometer reading from that scree. And from Location screen, I can have my location that is latitude and longitude. To have them I have a permission request. When I approve it, it will show my latitude and longitude also the current weather information of that location. So, that was all about application. Next part is implementation.





Sources used with exercises

<https://docs.flutter.dev/get-started/install/windows>

<https://www.facetsui.com/docs/generate-rn-code>

<https://www.geeksforgeeks.org/how-to-build-a-weather-app-in-android/>

<https://medium.com/flutter-community/weather-app-with-flutter-bloc-e24a7253340d>

<https://www.thedroidsonroids.com/blog/guide-on-how-to-publish-your-app-on-the-app-stores>

<https://developer.android.com/guide/components/broadcasts>

<https://reactnative.dev/docs/navigation>

[https://www.linkedin.com/pulse/google-play-store-app-submission-guidelines-complete-guide-/?trk=pulse-article more-articles related-content-card](https://www.linkedin.com/pulse/google-play-store-app-submission-guidelines-complete-guide-/?trk=pulse-article_more-articles_related-content-card)

<https://developer.android.com/guide/components/broadcasts>

<https://blog.logrocket.com/creating-listviews-in-flutter/>