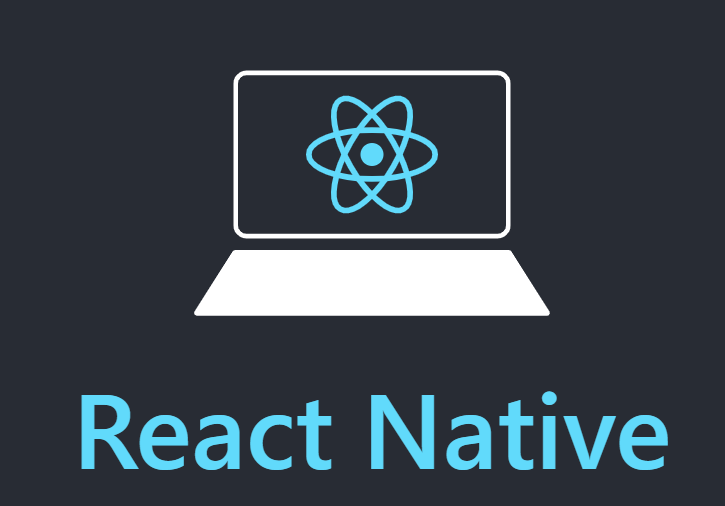
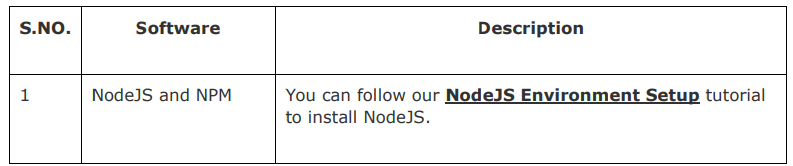
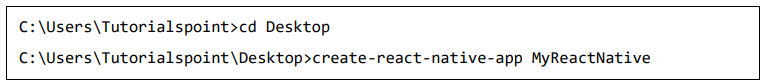
**Native Mobile Application Development(Covid19)**

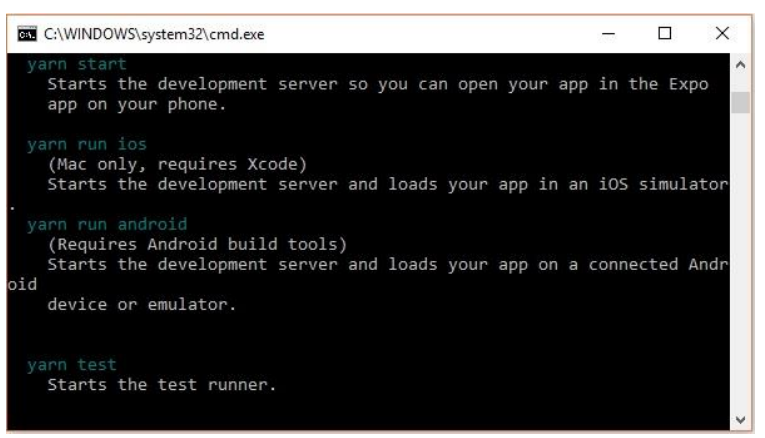
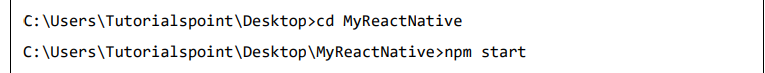
**Introduction**

**About Project**

**Overview  
Problem Statement:-Native Mobile Application Development**This report discusses the work done in development of COVID19 App with the help of React Native.The basic idea of the app is to show the live data of the Countries all over the world suffering from Covid Pandemic.  
 **React Native  
**React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components. With React Native, you don't build a mobile web app, an HTML5 app, or a hybrid app; you build a real mobile app that's indistinguishable from an app built using Objective-C or Java. React Native uses the same fundamental UI building blocks as regular iOS and Android apps. You just put those building blocks together using JavaScript and React. In the summer of 2015, Facebook released a framework called React Native which is used to build an application in React and then compile it to either Android or iOS.  
  
 **React Native Features Following are the features of React Native:**   
**• React** – This is a Framework for building web and mobile apps using JavaScript.  
 **• Native** − You can use native components controlled by JavaScript.   
**• Platforms** – React Native supports IOS and Android platform.  
  
 **React Native Advantages**  
 Follow are the advantages of React Native:  
 **• JavaScript** − You can use the existing JavaScript knowledge to build native mobile apps.  
 **• Code sharing** − You can share most of your code on different platforms.   
**• Community** – The community around React and React Native is large, and you will be able to find any answer you need.

**React Native Limitations**   
Following are the limitations of React Native:  
 **• Native Components** − If you want to create native functionality which is not created yet, you will need to write some platform specific code.  
  
**Environment Setup**There are a couple of things you need to install to set up the environment for React Native. We will use OSX as our building platform  


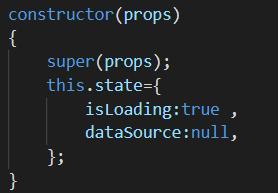
**Step1: Install create-react-native-app**   
After installing NodeJS and NPM successfully in your system you can proceed with installation of create-react-native-app (globally as shown below).  
  
**Step2: Create project**   
Browse through required folder and create a new react native project as shown below  
  
**Step3: NodeJS Python Jdk**  
Make sure you have Python NodeJS and jdk8 installed in your system if not, install them. In addition to these it is recommended to install latest version of yarn to avoid certain issues.   
Step4: Install React Native CLI  
 You can install react native command line interface on npm, using the install -g reactnative-cli command as shown below  

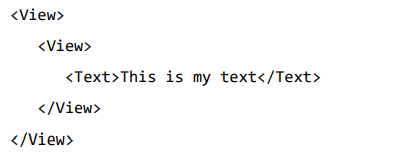

  
  
**Step5: Start react native**  
 To verify the installation browse through the project folder and try starting the project using the start command.  


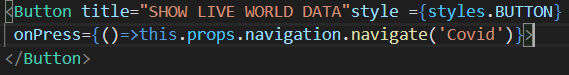
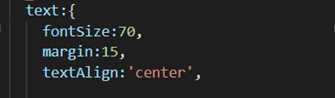
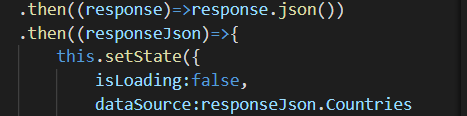
If everything went well you will get a QR code as shown below.  
 **As instructed, one way to run react native apps on your android devise is to using expo. Install expo client in your android devise and scan the above obtained QR code.**

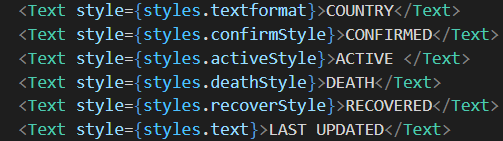
**Motivation behind building an app with ReactNative**There were two main reasons why I decided to build this project with ReactNative. The first one is that I am also learning web development. Since I enjoyed many of the practices and the building blocks of web apps, I thought that it would be very interesting and fun to be able to use a framework based on web development protocols in the making of an iOS app as well as android.  
  
My second motivation was that ReactNative is “Learn once, write anywhere”. Which means that I only have to learn how to work with the ReactNative framework once to be able to build mobile apps on many platforms, including iOS, Android and many others. This particular characteristic of ReactNative is very appealing. So unlike other frameworks, ReactNative provides a Javascript library to interact with native components to each platform. I had access to all the native iOS components ( camera, ListView, GridView, ImageView…) and was able to interact with them through Javascript. The other reason behind choosing ReactNative is the way I was able to structure my project.  
  
  
**These are the main concepts we have learnt about React/ReactNative which were implemented in our project:**   
  
**STATE AND PROPS**  
The data inside React Components is managed by state and props.

Difference between State and Props  
The state is mutable while props are immutable. This means that state can be updated in the future while props cannot be.  
Since state is mutable, we can update it by creating the deleteState function and call it using the onPress = {this.deleteText} event.  
Presentational components should get all data by passing props. Only container components should have state.  
- Using props, we can send data to child components, it is a powerful way to render dynamic data

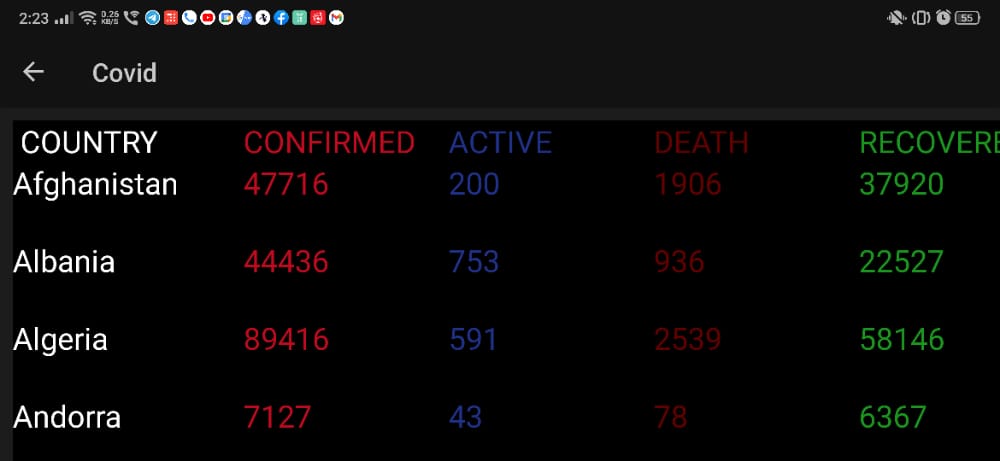
  
 SetState: -Helper method to update the state of a component.

**VIEWS**View is the most common element in React Native. You can consider it as an equivalent of the div element used in web development.  
 Use Cases  
 Let us now see a few common use cases.  
 • When you need to wrap your elements inside the container, you can use View as a container element.  
 • When you want to nest more elements inside the parent element, both parent and child can be View. It can have as many children as you want.   
• When you want to style different elements, you can place them inside View since it supports style property, flexbox etc.  
 • View also supports synthetic touch events, which can be useful for different purpo

**TEXT**  
This component can be nested and it can inherit properties from parent to child. This can be useful in many ways. We will show you example of capitalizing the first letter, styling words or parts of the text, etc.  
  
styles.text will be applied to all Text components.  
You can also notice how we set other styling properties to some parts of the text. It is important to know that all child elements have parent styles passed to them.  
  
**SCROLLVIEW**  
  
By using this component we will be able to see scrollable list of data.  
 **STATUS BAR**  
The Status bar is easy to use and all you need to do is set properties to change it.   
The hidden property can be used to hide the status bar. This is default value.   
The barStyle can have three values – dark-content, light-content and default. This component has several other properties that can be used. Some of them are Android or IOS specific. You can check it in official documentation.  
  
  
**BUTTONS**we can use the onPress prop for handling touch event. Facebook offers the Button component, which can be used as a generic button. Consider the following example to understand the same.  
  
  
**STYLING**There are a couple of ways to style your elements in React Native. You can use the style property to add the styles inline. However, this is not the best practice because it can be hard to read the code.At the bottom of the file, we will create our stylesheet and assign it to the styles constant. Note that our styles are in camelCase and we do not use px or % for styling.   
To apply styles to our text, we need to add style = {styles.Text} property to the Text element.  
  
  
**Navigation Container**  
To navigate from one screen to another we use navigation container and stack navigator.  
  
**Covid19 Data Extraction**We have extracted the Jason API from the covid19.org website.  
we stored the data in “dataSource:responseJason”.   
Now we have country wise data which we will display in the Tabluar form.

We are displaying Country,Confirmed cases,active,death,recovered & last updated.

**Screenshots**

**REFERENCE**

1. <https://reactnative.dev/>
2. **Microsoft session by Ashish Mishra**