API BUILD

PART 1 – WRITE UP FOR COVID -19 TRACKER

**Introduction**

For this API Assignment I’m going to create a chart application that consumes an API and adds value by showing COVID 19- cases, deaths around the world. I found this API watching a YouTube tutorial and decided to use this to learn and develop my React js, and Chart js skills. I documented the entire process for future projects.

This is the API below. URL = <https://covid19.mathdro.id/api>

**{**"confirmed": **{**"value": **2898082**,"detail": "<https://covid19.mathdro.id/api/confirmed>"**}**, this is the number of confirmed cases and the value

"recovered": **{**"value": **822165**,"detail": "<https://covid19.mathdro.id/api/recovered>"**}**,

"deaths": **{**"value": **203025**,"detail": "<https://covid19.mathdro.id/api/deaths>"**}**,

"dailySummary": "<https://covid19.mathdro.id/api/daily>", - daily data

"dailyTimeSeries": **{**"pattern": "[https://covid19.mathdro.id/api/daily/[dateString]](https://covid19.mathdro.id/api/daily/%5bdateString%5d)", - daily data for the day, not used in this API

"example": "<https://covid19.mathdro.id/api/daily/2-14-2020>"**}**,

"image": "<https://covid19.mathdro.id/api/og>",

"source": "<https://github.com/mathdroid/covid19>", - author of API

"countries": "<https://covid19.mathdro.id/api/countries>", -data of all countries for list and allows user to select from list

"countryDetail": **{**"pattern": "[https://covid19.mathdro.id/api/countries/[country]](https://covid19.mathdro.id/api/countries/%5bcountry%5d)","example": "<https://covid19.mathdro.id/api/countries/USA>"**}**, - specific country data

"lastUpdate": "2020-04-26T05:31:19.000Z" – time of last update

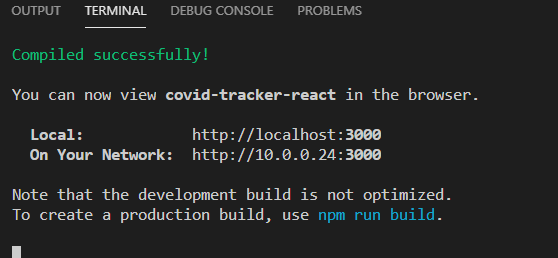
**Process**

To start creating this application we need to Create corona-tracker folder and Download react in this folder

npm install --save axios react-chartjs-2 react-countup in folder also add classnames

Axios helps pull in API. Chartjs is used to build a functional chart and react-countup is used to show numbers of people affected by COVID in an animation

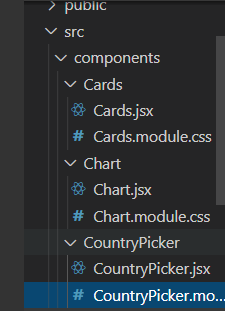
npm start to connect to localhost:3000

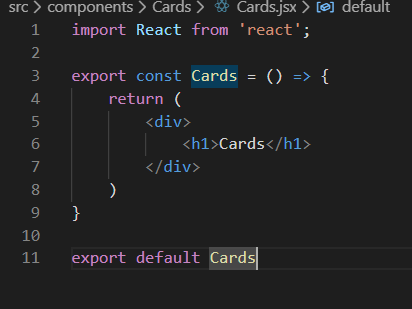


Now create components folder in src

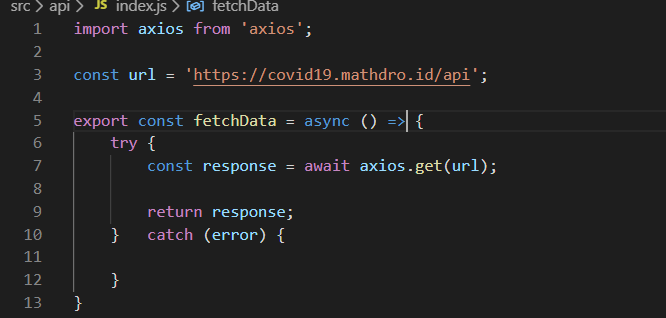
Create Cards, Chart and CountryPicker folders in the component folder

In cards folder create Cards.jsx file and Cards.module.css (make it module so it remains in the cards folder and does not become a global CSS file.

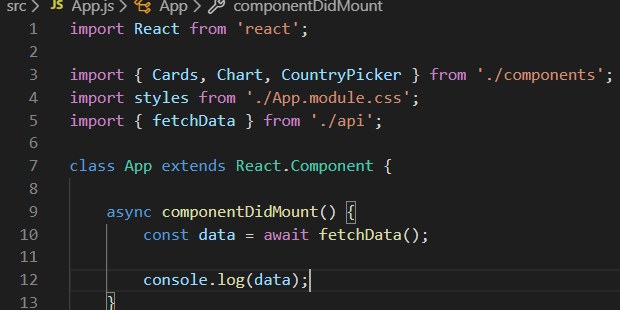
Now create the same for Chart and CountryPicker folders

Then create functional components in each of JSX files 

Now create your api folder that you’ll pull from with its own index.js file



Now we can call this function inside of app.js. Now we have fetchData function which is an asynchronous function returning a response from the API



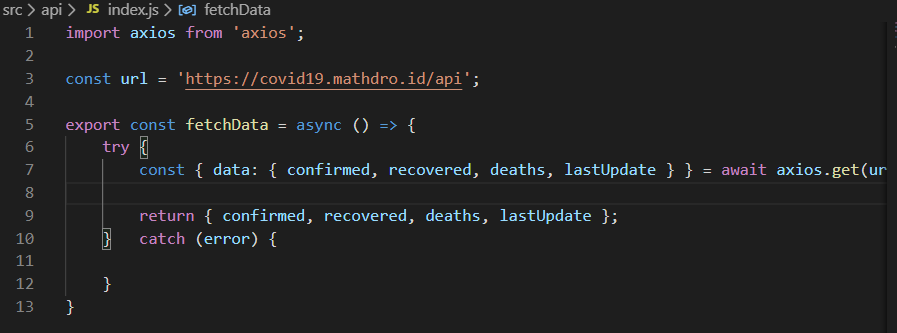
Put fetchData into the api class component and console.log it



**IMPORTANT:** Console.log now shows the data you are fetching which is the API data. Mission accomplished. API data is now being pulled into the application.

Now we can go back into the API and configure the response however we want

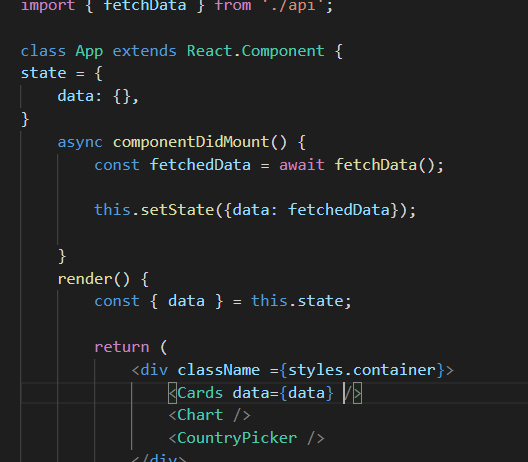
We can choose what type of data we want. Which is confirmed cases, recovered cases, deaths and last updated information by DE structuring the data in our API indexjs folder like so:



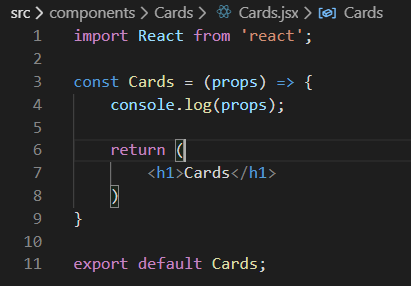
And now in our console we have confirmed, deaths, lastUpdate and recovered cases.

Now we can decide what we want to do with this data

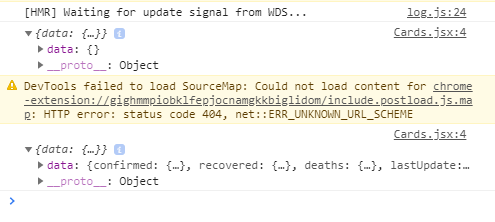
Add this.state and fetchedData to your app.js file. And ask it to send data to Cards.jsx



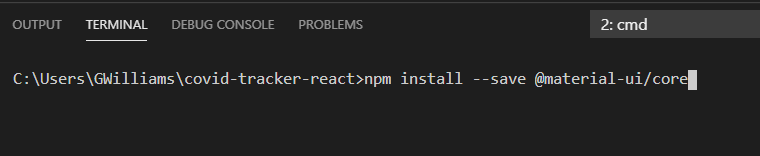
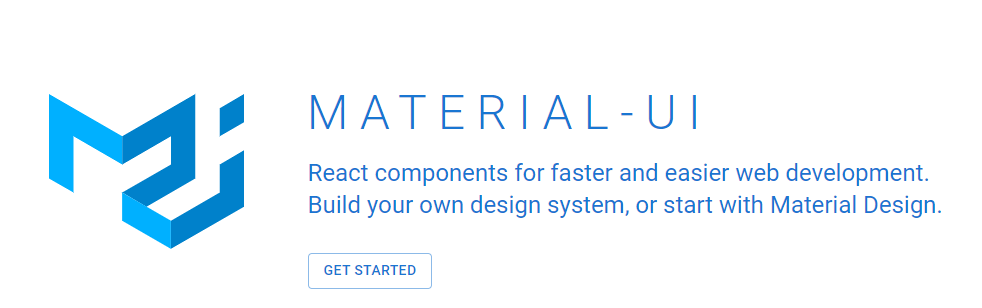
Add a props to your Cards.jsx file

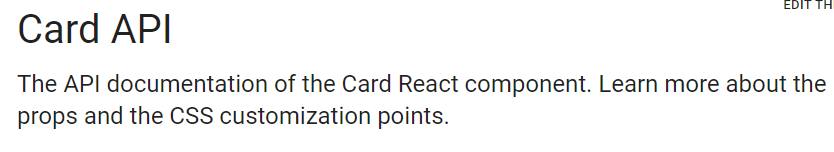


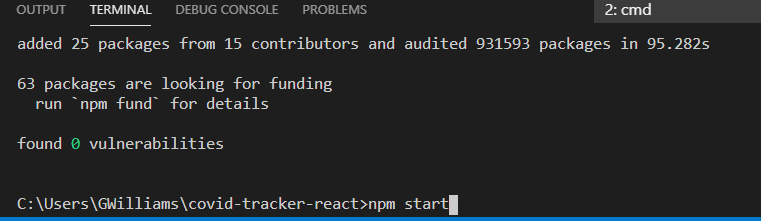
When you console.log props you get this in the terminal showing the Data coming through



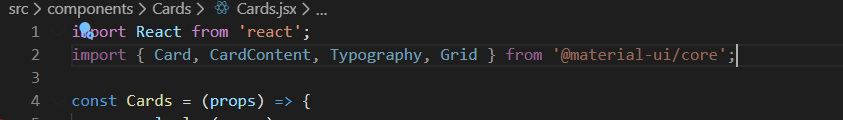
Stop current program running and

Install Material core/ui in your terminal in your folder so we can build the cards visually. It’s like bootstrap but more suited for react. We’re going to use the cards UI. Instead of coding it in CSS we can call it as a className in our application and get the UI.

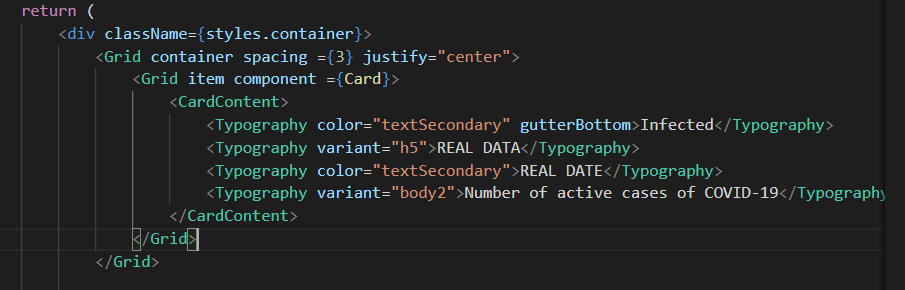


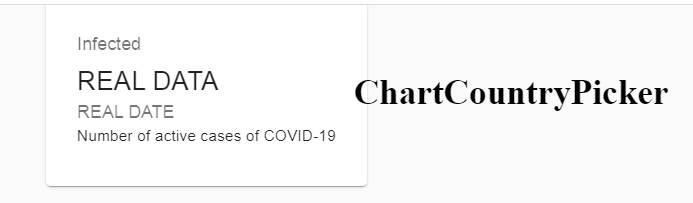
Now let’s start our application and implement our cards

Import into your Cards.jsx file from ‘@material-ui/core’;

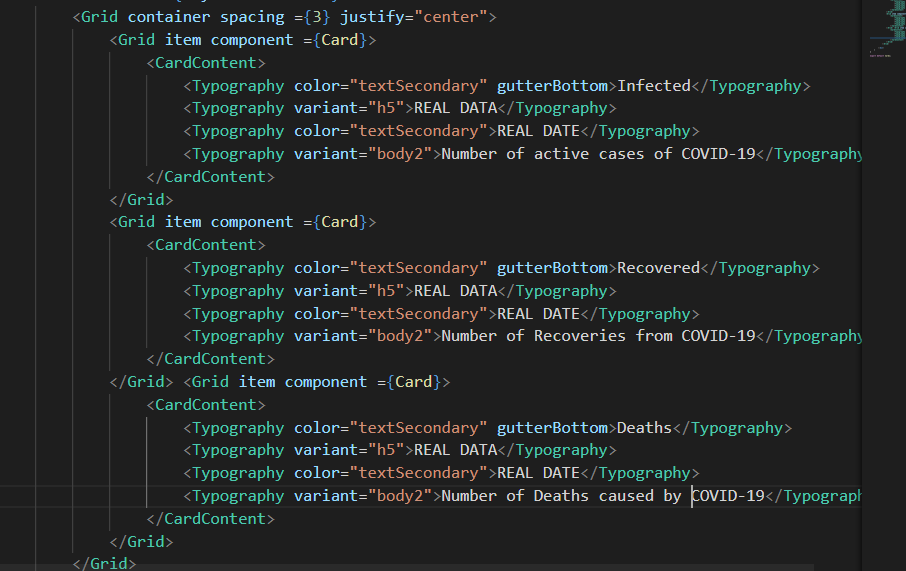


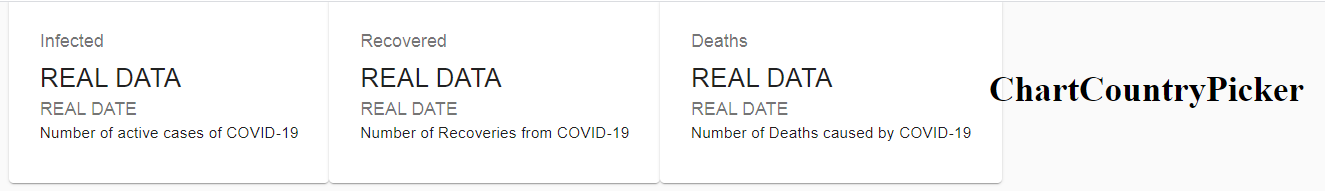
Add a Grid container justify center all items inside of it. Have one grid type of item and inside of their the component will be card which will make it look like its hovering. Inside Card we will have card content. Inside their we’ll have typography (used for text/heading) – gutterbottom will provide padding on the bottom and it’ll we call infected for infected people. Then another typography below we a variant of real data. Which will come from props. Then another typography with the date. Then the last typography will have a variant of body 2 which Is material UI and itll save number of cases of covid – 19



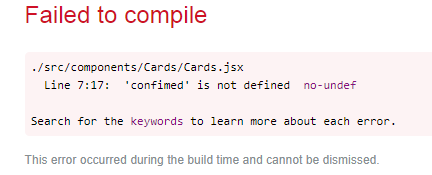


Do the same for recovered people and number of deaths. We can fix the spacing as well.

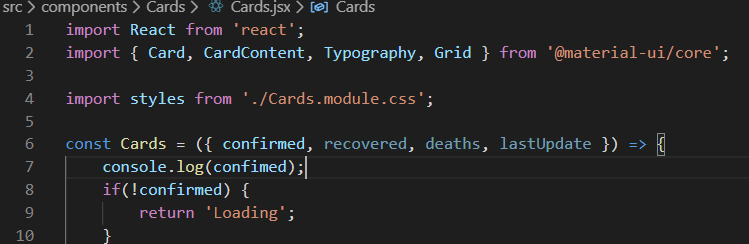




Now we need to get the actual Value of confirmed

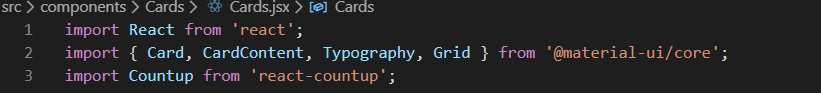


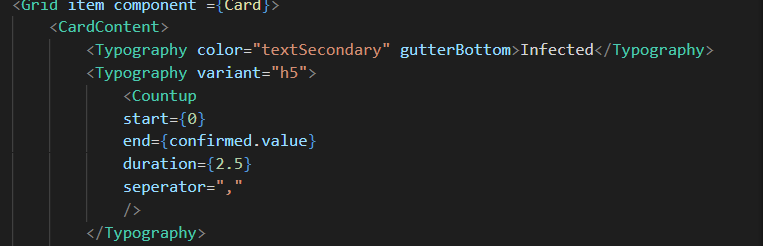
Confirmed data is undefined so we get an error because data has not yet been fetched so we need to add an If statement and console.log(confirmed)



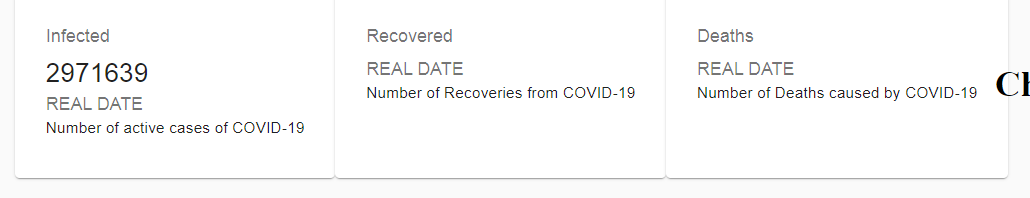
We added a data {props} to app.js and so we need to add it into the const cards and destructure the data with {}

Import react-countup into cards.jsxto get the counting effect



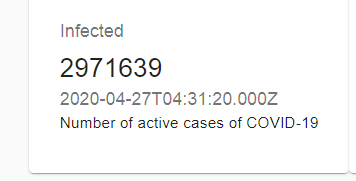


Add count up function after variant and it’ll work by counting up in the browser



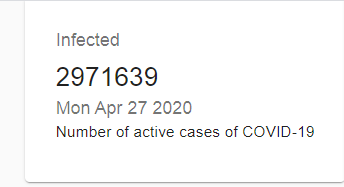
Real date comes from the last update





Looks strange so we change it to get a real format. We need to make a newDate object and call it toDate string () with the call function to return the date in a readable format.

 <Typography color="textSecondary">{new Date(lastUpdate).toDateString()}</Typography>



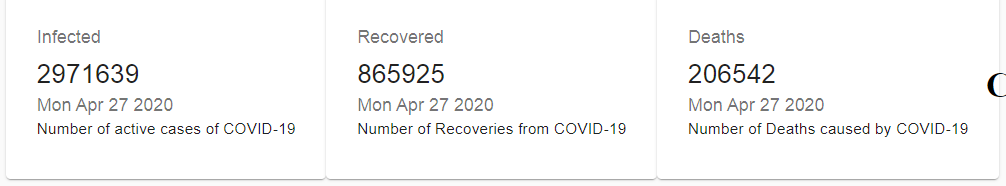
Now add the values for recovered and deaths to their card section. You can copy the count up section and replace confirmed.value with {recovered.value} and {deaths.value}

<Typography variant="h5">

                            <Countup start={0} end={confirmed.value} duration={2.

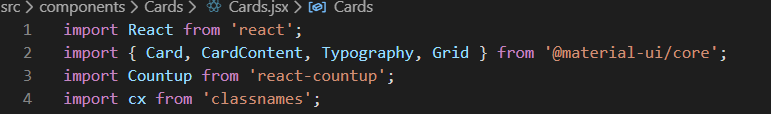
5} seperator="," />

You can just copy and paste the datestring since it’s the same date. Now you’ll get the dates.



Now let’s add some styles using the module.css

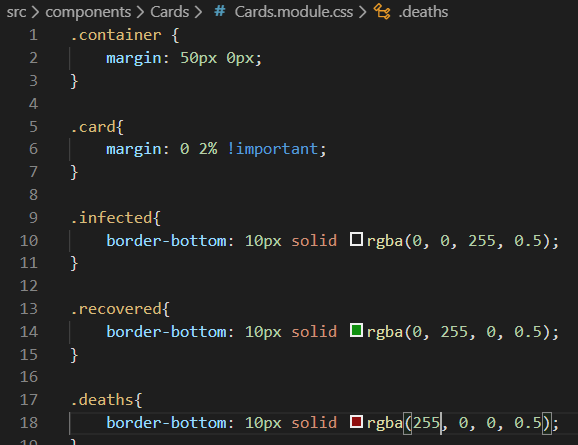
Import classnames in Cards.jsx



Add screen size in Cards.jsx xs{12} all 12 spaces of mobile screen. And medium devices and larger md{3} and 3 spaces of the screen and put this for all the grid item components. Replace the styles.infected with .recovered and .deaths

 <Grid item component ={Card} xs={12} md={3} className={cx (styles.card, styles.infected)}>

In the Cards.module.css



Now we have blue for infected, green for recovered, red for deaths



Now we can create the chart. We going in api index.js and add a second function to get the daily function. Notice the url is/daily for axios when we call from a different part of the API.

}

export const fetchDailyData = async () => {

    try {

       const { data } = await axios.get('${url}/daily');

       console.log(data);

    } catch (error) {

    }

}

To be able to see console.log we need to call() it in the chart.jsx because that’s where we should get the data

Import useState and useEffect which is another name for the hook. Add const [dailyData and setDailyData]

And add useEffect methodwhich accepts a callback

Import fetchDailyData from api

Const fetchDaily data from await which is a promise

Now the daily data is populated.



We’re going to create 2 types of charts. A bar graph and a line chart

Import both from react-chartjs-2 and import styles from chart.module.css



Import line from react- chart js.

const lineChart = (

        dailyData[0]

        ? (

        <Line

        data={{

            labels: '',

            datasets: [{}, {}],

        }}

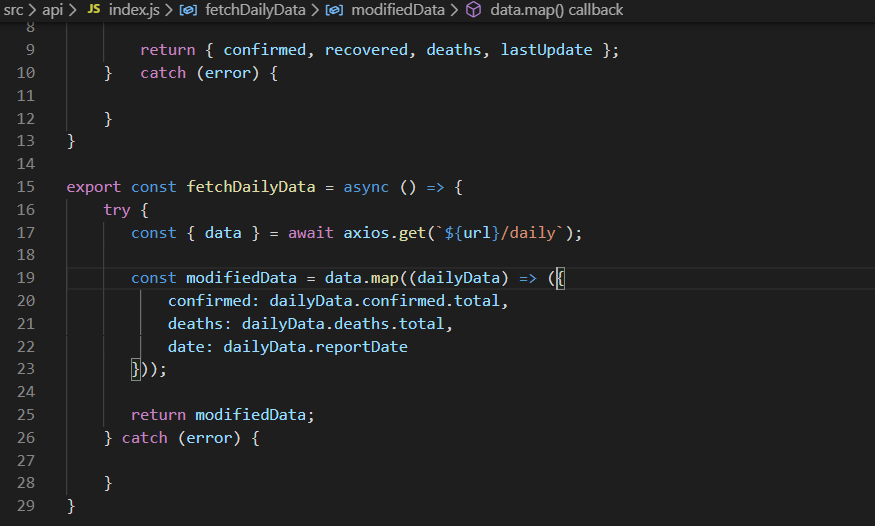
        />) : null

This is if we are going to fetchDailydata then return the linechart. And if nothing is available then return a null value.

Got a new error. So now must install chart.js for the react-chart to work

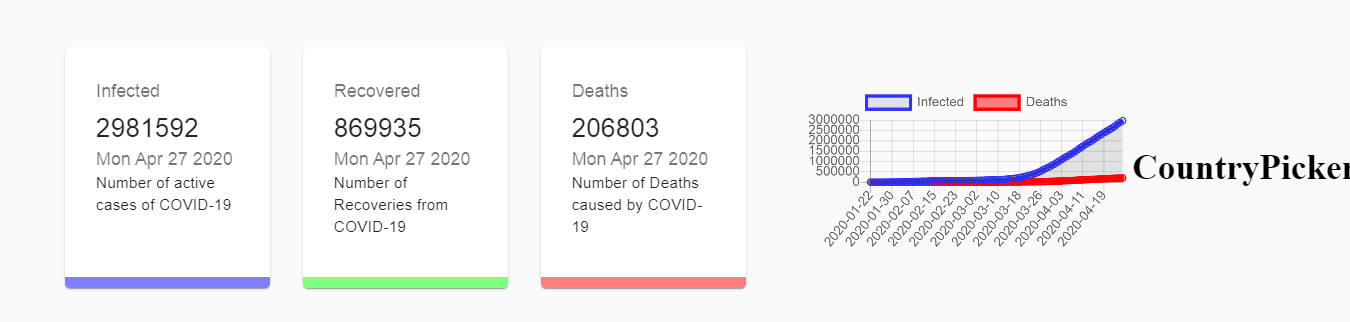
After that we can create the Line chart and display it

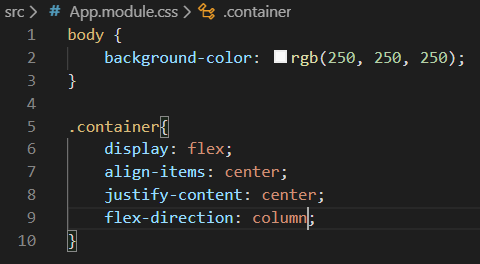
In API create a modifiedData variable using const



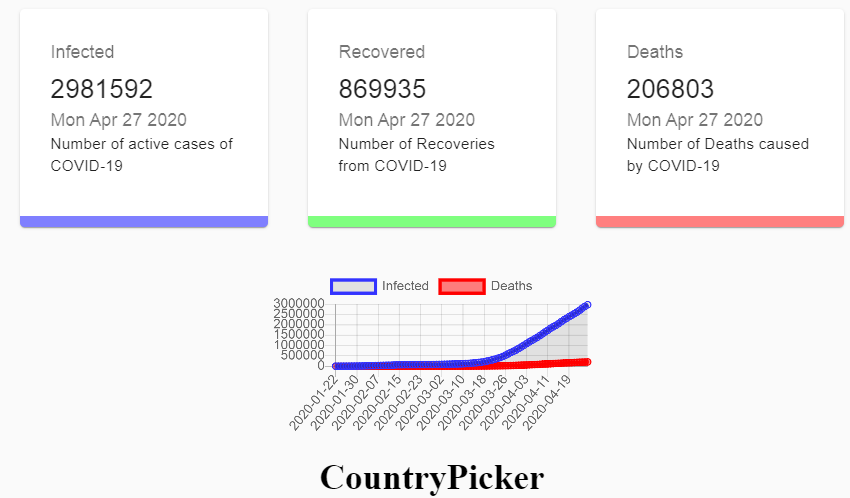
In Chart.jsx define what the datasets will look like.



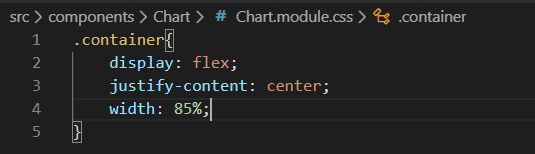
Boom we have our chart. We need to put it in the middle and make it bigger by going into the app.module.css and add flex direction: column



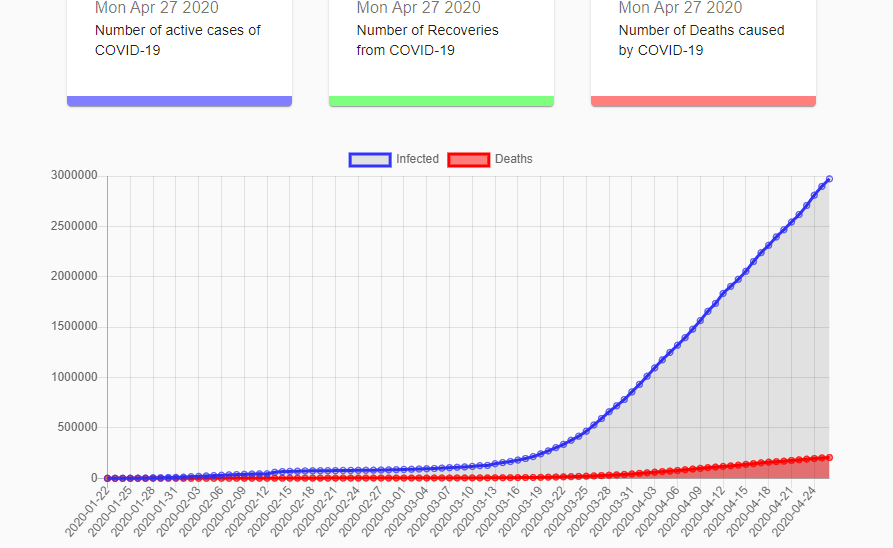
Put container chart In middle of page



To make it bigger we need to go into Chart.module.css and add styles

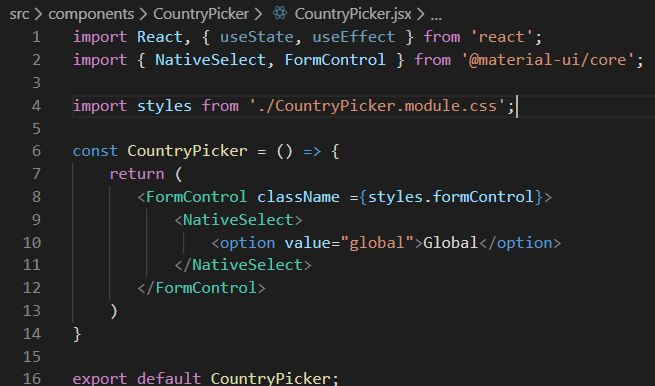


Now we have it populating and sized properly



Now we create the country picker starting in countrypicker.jsx

Import NativeSelect and formcontrol to enable selecting a country from @materialUi core



And put make sure country is included in our app.js

class App extends React.Component {

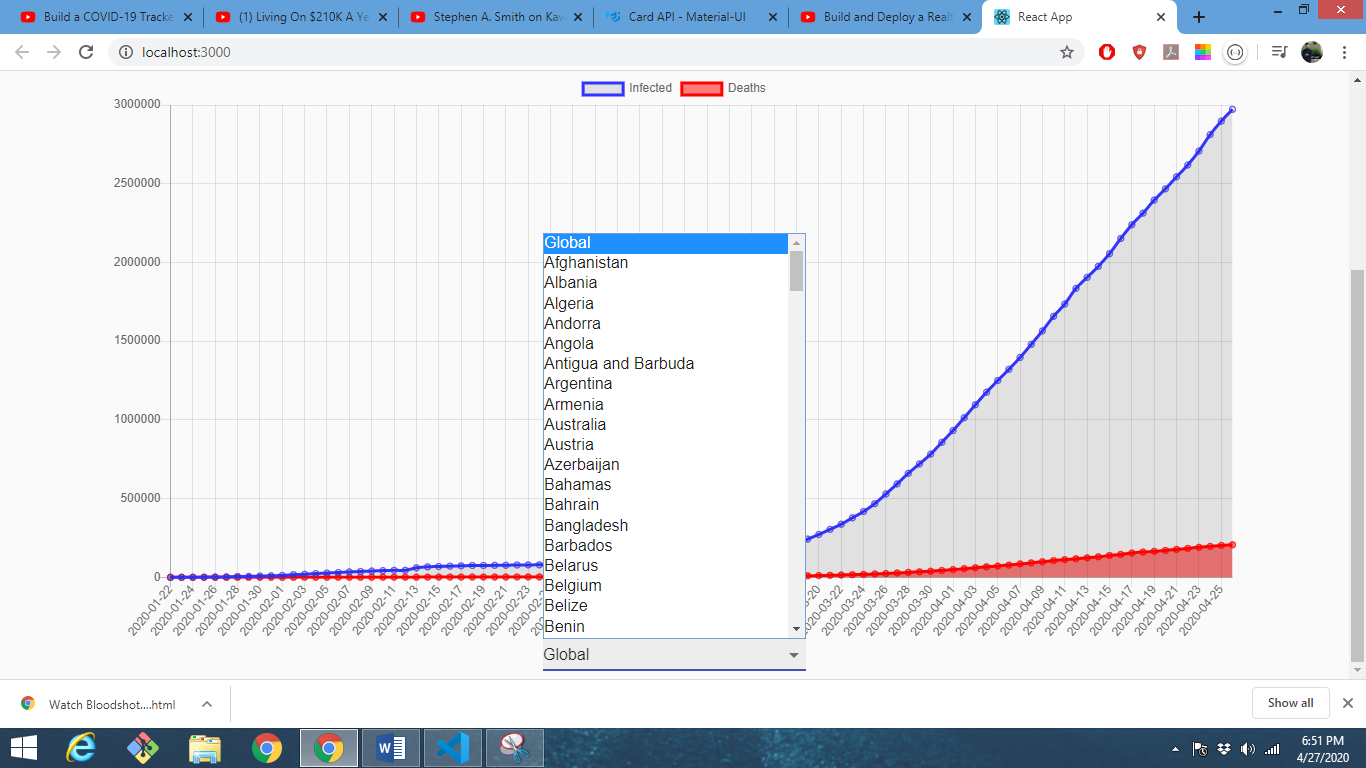
state = {

    data: {},

    country: '',

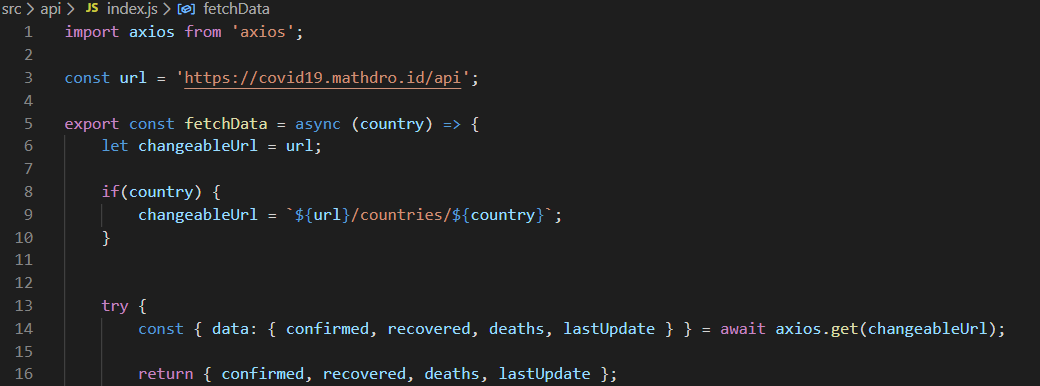
Now we can fetch the countries from the API

We can populate all 181 countries using a single line  {fetchedCountries.map((country, i) => <option key={i} value={country}>{country}</option>)}

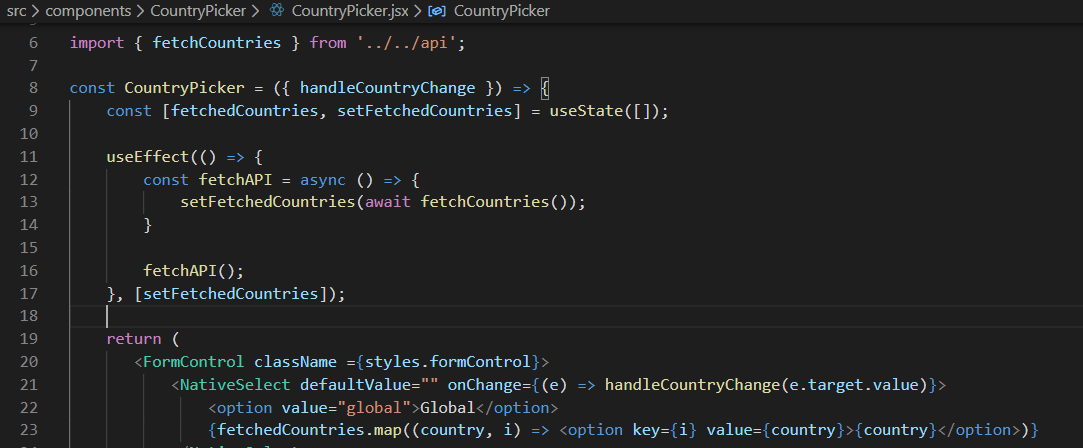


Okay now we need to make sure the countries will show information in the cards.

We need to fetch the data from the api and set the state by putting a changeable Url in the api (index.js) file



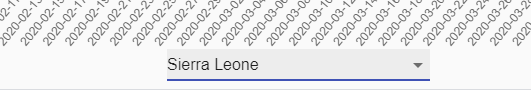
We need to put handleCountryChange into our CountryPicker.jsx file

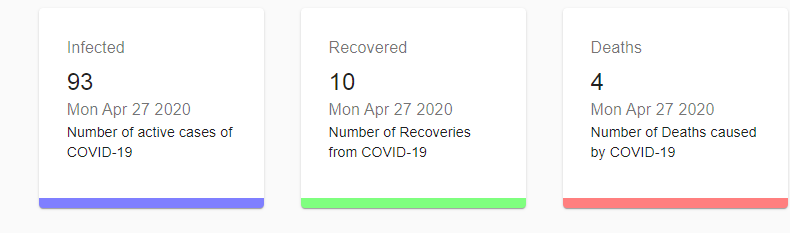


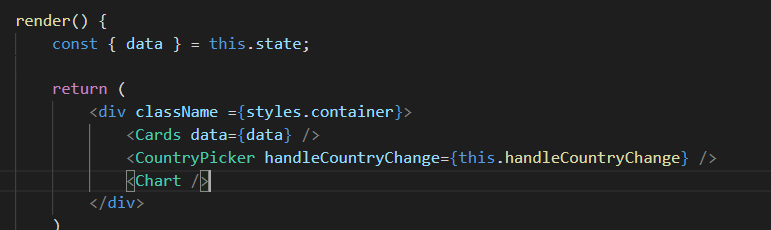
We need to put handleCountryChange into our app.js



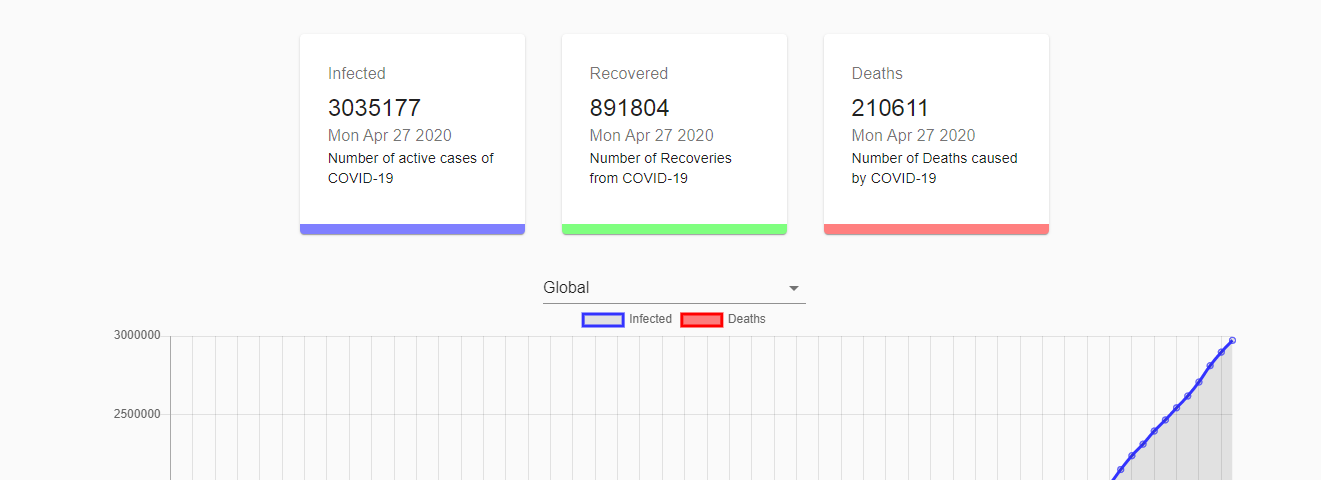
Now when I choose a country it populates the information in the cards.

For example 



I noticed my country picker was below my chart. This is because that was how it was ordered in the App.js file. So I switched their positions like so: 

Now my country picker is directly below the cards and above the chart



Now we can pass the data and country to the chart as well in our app.js so we can see visually the trends coming from the information into the chart.

  return (

            <div className ={styles.container}>

                <Cards data={data} />

                <CountryPicker handleCountryChange={this.handleCountryChange} />

                <Chart data={data} country={country}/>

            </div>

And destructure It in the state by putting country next to data in App.js

 render() {

        const { data, country } = this.state;

Now we can create the Bar Chart (graph. Below the map input

    const barChar = (

        data.confirmed

        ? (

            <Bar />

        ) : null

    )

Create the legend, title

const barChar = (

        data.confirmed

        ? (

            <Bar

            data={{

            }}

            options={{

                legend: { display: false },

                title: { display: true, text: `Current state in ${country}` },

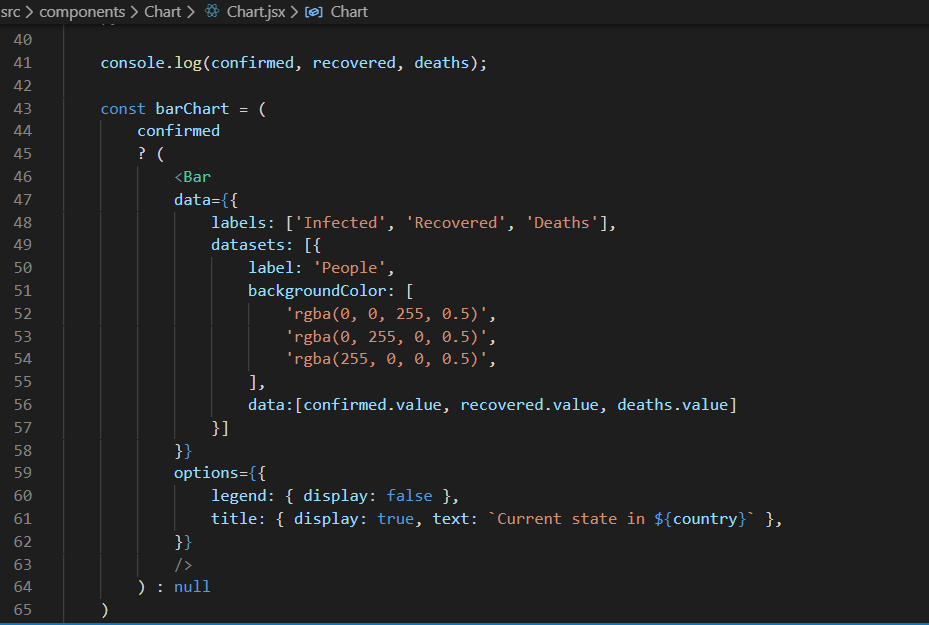
            }}

            />

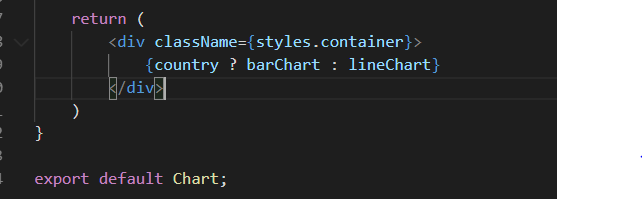
        ) : null

    )

Provide the data



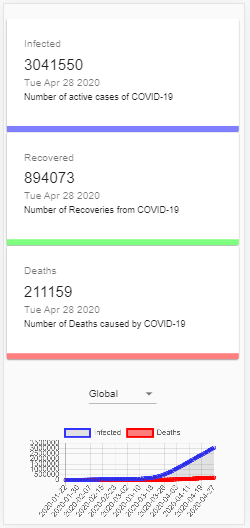
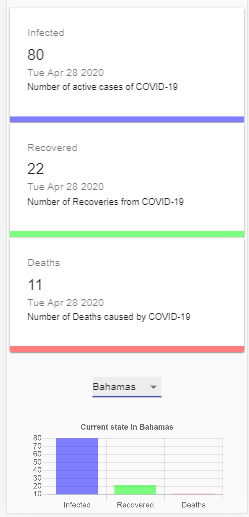
The code below is added to the return value. It says if there is a country show bar graph. If there isn’t show a line graph.



Now we get a bar graph when we choose a country



We can now see our final covid -19 tracker

**Conclusion**

This was a fun application to make. Building cards, a bar graph and a line chart took a lot of time but was well worth the final application. I now know how to use charts in future projects to render data from API’s.