PART 3

LAB 10

- ✓ **REST API**

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

Lab 10: Rest API

In this lesson, we will learn how to use the axios library to make a REST API call. The example will demonstrate calling an API service from CoinMarketCap to get the latest ETH price in USD. This is followed by an exercise to get historical ETH price from another API service from CryptoWatch and use it to create moving averages. Finally, we will provide the moving average services as a REST API service.

Create a Node Project

1. Create lab directory and change directory into it.

```
$ md lab && cd $_
```

2. Initialise the project

```
$ npm init -y
```

The directory will contain a single file called package.json which will grow when you start installing packages.

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

Register an account with CoinMarketCap

1. Sign up for an account with CoinMarketCap (https://coinmarketcap.com) and login your account.
2. Under API Key box, copy the API key.

   eg. XXXXXXXX-XXXX-XXXX-XXXXXXXX

Install axios package

1. Open terminal and install axios using npm i. axios (https://www.npmjs.com/package/axios) is a Promise-based HTTP client for node.js and browser.

```
$ npm i axios
```

2. Open package.json and you should find a line for the package dependency.

```
"dependencies": {
   "axios": "^0.27.2"
}
```

NOTE: The version number is important. It is very common for packages to fail due to breaking changes introduced by the dependent packages and their dependencies. To ensure that your node application is behaving consistently, make sure you are using the same version of package.

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

Getting Data from API Service

1. Create the file 10a-ethusd-latest.js and import axios.

```
const axios=require('axios');
```

2. Copy the sample for node.js from (https://coinmarketcap.com/api/documentation/v1/#section/Quick-Start-Guide) into the file.

```
let response = null;
new Promise(async (resolve, reject) => {
    try {
        response = await axios.get('https://sandbox-api.coinmarketcap.com/v1/cryptocurrency/listings/latest', {
            headers: {
                'X-CMC_PRO_API_KEY': 'b54bcf4d-1bca-4e8e-9a24-22ff2c3d462c',
            },
        });
    } catch(ex) {
        response = null;
        // error
        console.log(ex);
        reject(ex);
    }
    if (response) {
        // success
        const json = response.data;
        console.log(json);
        resolve(json);
    }
});
```

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

Getting Data from API Service

3. Go to the API reference from CoinMarketCap (https://coinmarketcap.com/api/documentation/v1/#operation/getV2CryptocurrencyQuotesLatest). Look for the API endpoint that will return latest cryptocurrency quote.
4. Update the API URL.
   - According to the API specification, the base URL for the sample is `https://sandbox-api.coinmarketcap.com` but you should use the actual base URL when making real API calls `**https://pro-api.coinmarketcap.com**`.
   - The endpoint we are interested in is the one we will use to fetch latest ETHUSD quotes which, in this case, is `**/v2/cryptocurrency/quotes/latest**`.
   - Finally, the API specification requires a symbol to be provided as a query parameter. Since we are interested in the current price for ETH, we will append `**?symbol=eth**` to the endpoint.
   - Therefore, the final API URL is `**https://api-pro.coinmarketcap.com/v2/cryptocurrency/quotes/latest?symbol=eth**`.
   - Change the sample file's API url to the following.

```
response = await axios.get('https://pro-
api.coinmarketcap.com/v2/cryptocurrency/quotes/latest?symbol=eth',
```

   - According to CoinMarketCap, this API endpoint is available for free.

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

5. Replace the API key in the sample with the one you are registered with.

```
headers: {
    "X-CMC_PRO_API_KEY": "XXXXXXXX-XXXX-XXXX-XXXXXXXX",
},
```

6. Run the script and the output should look like this

```
{
    status: {
        timestamp: '2022-05-20T07:40:43.893Z',
        error_code: 0,
        error_message: null,
        elapsed: 33,
        credit_count: 1,
        notice: null
    },
    data: { ETH: [ [Object] ] }
}
```

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

Using Data from API Call

1. Extract the pricing data from the response object.

   From the API reference, a 200 success response should return a JSON in the form of the following:

```
{
    "data": {
        "1": {
            ...
            "quote":{
                "USD": {
                    "price":...
                }
            }
        }
    },
    "status": {
        ...
    }
}
```

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

Using Data from API Call

1. Extract the pricing data from the response object.

   From the API reference, a 200 success response should return a JSON in the form of the following:

```
{
    "data": {
        "1": {
            ...
            "quote":{
                "USD": {
                    "price":...
                }
            }
        }
    },
    "status": {
        ...
    }
}
```

Lab 10a: Getting real-time ETHUSD price using CoinMarketCap API

2. We want to extract the USD price from the response data from any provider, so we will replace the line displaying the console.log(json) to the following.

```
// success
const json = response.data;
console.log(
    `The ETH price is ${response.data?.data?.ETH[0].quote.USD.price} quoted at
${response.data?.data?.ETH[0].last_updated}`
);
```

3. Run the script again. The result will show something like the following

```
The ETH price is 2027.8790002877977 quoted at
2022-05-20T07:55:00.000Z
```

Lab 10c: Create API Service

Lab 10c: Create API Service

In the previous section, we have learnt how to call API to get historical price for ETHUSD. In this lesson, we will learn to create our own API service to provide the 10-day moving average price.

Install Express package

1. Open terminal and install Node.JS Express using npm. Express (https://www.npmjs.com/package/express) is the library that will be used for creating the API service for this course.

```
$ npm -g i express
```

Lab 10c: Create API Service

Create and run an API Service at http://localhost:5050

1. Create 10c-ethusd-api.js and import the express function.

```
const express = require("express");
```

2. Create an API webservice that runs at port 5050.

   Create an express object and start the service using listen(). The listen method() takes in 2 parameters - the network port number for receiving API requests and the callback to invoke after the service is started.

```
const app = express();
var server = app.listen(5050, function () {
    console.log("Service running at port:", server.address().port);
});
```

3. Run the service in terminal

```
$ node 10c-ethusd-api.js
# Service running at port: 5050
```

4. Open your browser at 'http://localhost:5050'

   The page should show `Cannot GET /`. That means your service is running but the function is not provided yet.

Lab 10c: Create API Service

Create a DailyPrice Endpoint

1. Import getDailyPrices from lesson 8 maPrice.js.

```
const { getDailyPrices } = require("./maPrices.js");
```

2. Create an endpoint /daily using getDailyPrices to provide daily price API service using the data from CryptoWatch API.

```
const app = express();

// Create the daily endpoint
app.get("/daily", async (req, res) => {
    // get daily price from CryptoWatch
    const response = await axios.get(
        "https://api.cryptowat.ch/markets/bitfinex/ethusd/ohlc?periods=86400"
    );
    const json = response.data;
    const result = getDailyPrices(json);
    res.send(result);
});

var server = app.listen(5050, function () {
    console.log("Service running at port:", server.address().port);
});
```

3. Open the browser at `http://localhost:5050/daily` and you should see the JSON data returned from your API.

Lab 10c: Create API Service

Create Moving Average Endpoint with Parameter

In this section, we will add another endpoint that will return the moving average of ETHUSD for any number of days based on an optional parameter. For example, `http://localhost:5050/ma/10` for 10-day m.a. and return daily price if without parameter `http://localhost:5050/ma`

Lab 10c: Create API Service

1. Create the `ma` endpoint.
   - To create a parameter for the route, specify the parameter name preceded by a `:`. To make the parameter optional, superseded the name with `?`. Therefore, the route `/ma/:days?` will accept http://localhost:5050/ma/10 or http://localhost:5050/ma
   - To use the parameter, you can extract it from the `req.params` object.
   - Because the parameter is passed in as a string, it needs to be converted to a number using `parseInt()`.
   - If parameter is falsy (ie. null, omitted, 0), then we will return daily price instead.

```
app.get("/ma/:days?", async (req, res) => {
    try {
        // get daily price from CryptoWatch
        const response = await axios.get(
            "https://api.cryptowat.ch/markets/bitfinex/ethusd/ohlc?periods=86400"
        );
        const json = response.data;

        // if days is provided, get the moving average, else return dailyPrice
        let result;
        if (!req.param.days) result = getDailyPrices(json);
        else result = maPrices(json, parseInt(req.param.days));
        res.send(result);
    } catch (e) {
        console.error(e);
        res.send(e.toString());
    }
});
```

Lab 10c: Create API Service

2. Open the browser at `http://localhost:5050/ma/10` and `http://localhost:5050/ma/30` will return the result for 10-day moving averages and 30-day moving averages respectively.

Lab 10d: Display API data in Front-End

Lab 10d: Display API data in Front-End

Most modern web applications are no longer developed using vanilla Javascript. Instead, they use web frameworks that can better capitalise on Javascript skills to develop powerful and aesthetic front-end UI.

For this final lab, we will not be going through how to develop a single-page app but if you are interested to find out more you can refer to the README.md file in the fin535-chart directory. We will download a ready made front-end that is created to illustrate how the client consumes the API service for ETHUSD daily and moving average prices that you have created.

Download and integrate to web front-end

1. Create a new directory /lab-react and change directory into it.
2. Clone the repository from github

```
$ git clone https://github.com/RoyLai-InfoCorp/fin535
$ cd fin535/fin535-chart
```

3. Install the packages and open the folder in Visual Code.

```
$ npm i
```

Lab 10d: Display API data in Front-End

4. Open the file /src/App.jsx and refer to the logic under the function handleClick. This function is triggered when the Show Chart button is clicked and will call the API Service endpoints daily and ma that was created in the earlier lab session using axios. It will display the data in a chart for the last 300 days.

```
const handleClick = async () => {
    const totalDays = 300;
    let response;

    response = await axios.get(`http://localhost:5050/daily`);
    const dates = response.data
        .slice(0, totalDays)
        .reverse()
        .map((x) => x.date);

    const dailySeries = {
        label: "daily",
        data: response.data
            .slice(0, totalDays)
            .reverse()
            .map((x) => x.price),
        borderColor: "rgb(132, 255, 99)",
    };

    response = await axios.get(`http://localhost:5050/ma/10`);
    const ma10Series = {
        label: "ma10",
        data: response.data
            .slice(0, totalDays)
            .reverse()
            .map((x) => x.price),
        borderColor: "rgb(132, 99, 255)",
    };
```

Lab 10d: Display API data in Front-End

```
    response = await axios.get(`http://localhost:5050/ma/30`);
    const ma30Series = {
        label: "ma30",
        data: response.data
            .slice(0, totalDays)
            .reverse()
            .map((x) => x.price),
        borderColor: "rgb(255, 99, 132)",
    };

    if (dates) {
        setData({
            labels: dates,
            datasets: [dailySeries, ma10Series, ma30Series],
        });
    }
};
```

5. Run the React app and open browser at http://localhost:3000 and click on Show Chart when the page is loaded.

```
$ npm start
```

Lab 10d: Display API data in Front-End

Cross Origin Resource Sharing (CORS)

You will encounter the error **... from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.**

Origin refers to the domain in the URL you specified in the browser when you access the service. Because the API is hosted at 'http://localhost:5050' but you are making the call from the react app hosted at 'http://localhost:3000', this results in the violation of the cross-origin security policy. To address this problem, the destination service, ie. the API Service has to allow its API to be accessible from the origin 'http://localhost:3000'.

1. Install `cors` extension for express in the API service directory.

```
$ npm i cors
```

2. Update `10c-ethusd-api.js` to import `cors` and include a line to `app.use(cors())`.

```
    const cors = require('cors')
    app.use(cors());
```

   NOTE: For simplicity sake, the above code will allow CORS from anywhere. In practise, you should limit the origin to specific domain names as well as the HTTP method used.

3. Restart your app again.