Financial Dashboard Application

HOOKS

Framework: React

JavaScript

HTML5

CSS3

Bootstrap

Project Presentation

By

Vikram Guhilot

V1.0

Introduction

The financial Dashboard application is an application that was created for a user to be able to get information of various financial performances in real time by fetching current market performances from 3rd party APIs that are tracking an maintaining the data on a regular basis, this application has touched on markets like currencies, precious metals, crypto currencies and country information – All these features can be further developed and are in beta phase as of know

Motivation:

The main reason for building this application was to put my React skills to the test, in previous classes I learned technologies like HTML, CSS and JS. However, I had not used a powerful framework that used the technologies mentioned above to create robust and maintainable applications. I had a choice of frameworks to use i.e. (Angular, Vue and React), I chose React as all my friends in industry recommended it as React was extremely popular and gaining more momentum with the introduction/addition of Hooks, React also has a strong community for support online, it is maintained by Facebook and is super lightweight.

React framework features implemented:

Hooks: useState, useEffect, useContext

Router

Context

JSX

Implementation:

In this project you will notice, that I have not used a single class-based component, or I have not implemented any classes. Why should I or why would I? With the introduction of hooks, there is absolutely no reason to use classes. The useState hook is used to maintain and declare state for which we previously used classes (this.state), also other lifecycle methods like componentDidMount, componendDidUpdate, etc .. that used classes can be easily implemented using hooks like useEffect, useRef etc, which we can configure to fire at any lifecycle event. For the longest time the React community turned towards REDUX for state management as Reacts own context/state management was a little complex to implement, however that has all changed with the hooks, it has become extremely easy to implement and we can minimize or even eliminate the use of Redux

Data Flow:

In react data flows from top to bottom, from parent to child and not backwards or sideways, so now you must have questions, one of them definitely being what if a component somewhere far away in the tree that is dependent on data from a component on the other side of the tree, how are you going to get access to it, since data only flows in one direction top to bottom

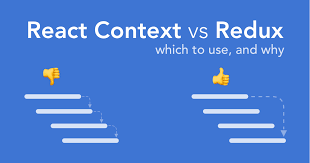


Figure1

A picture containing holding, light

Description automatically generated

Figure2

A picture containing light

Description automatically generated

Figure3

A close up of a screen

Description automatically generated

Figure4

A picture containing monitor, holding, light

Description automatically generated

Figure5

A picture containing light

Description automatically generated

Figure6

A picture containing clock, meter, light

Description automatically generated

**Create Context**

To begin, we create a new Context. As we want the entire app to have access to this, we go to index.js and wrap the app in ContextProvider

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import {BrowserRouter as Router} from "react-router-dom"

import {ContextProvider} from "./Context"

ReactDOM.render(

<ContextProvider>

  <Router>

    <App />

  </Router>

</ContextProvider>

, document.getElementById('root')

);

**We also wrap <App/> with Router as the App component is from where we will be routing to different pages**

**Creation of context provider component**

**A picture containing drawing

Description automatically generated**

**Code …. to manage/update state**

**A screenshot of a computer

Description automatically generated**

Here **ContextProvider** is the provider and **Context** is the consumer, we export both of them so that we could use them in different components

Eg ContextProvider wraps the App component signifying that data will be passed from this level.

Context will be imported into components where they will have access to data

eg

import React, {useContext} from "react"

import {Context} from "../Context"

import ContSearch from "../components/ContSearch"

import '../App.css';

function Country(){

const {countryInfo, flag, moreData}=useContext(Context)

return(

JSX code ……….

)

}

export default Country

**Managing of State by the useState Hook**

const [url, setURL] = useState("")

    const [coinData,setCoinData] = useState([])

    const [country,setCountry] = useState("")

    const [countryData, setCountryData] = useState([])

    const [countryInfo, setCountryInfo] = useState([])

    const [flag, setFlag] = useState("")

    const [moreData, setMoreData] = useState([])

    const [rate, setRate] = useState("")

    const [currUrl, setCurrUrl] = useState("")

    const [metal, setMetal] = useState([])

    const [too, setToo] = useState("")

**useEffect Hook**

useEffect(()=>{

         fetch(currUrl, {

            "method": "GET",

            "headers": {

                "x-rapidapi-host": "currencyconv.p.rapidapi.com",

                "x-rapidapi-key": "74582e3d89msh6a4a888a5292854p1d279cjsne4a41671a67f"

                 }

            })

            .then(res=>res.json())

            .then(data=>setRate(data))

        },[currUrl])

useEffect(()=>{

            fetch(metalUrl)

                .then(res=>res.json())

                .then(data=>setMetal(data))

        },[metalUrl])

The useEffect hook takes two parameters a call back function and []

The hook fires at any change of the values present inside the array, if it is an empty array if fires only once, if you do not provide a [] it fires continuously

The fetch function: Data is converted to JSON format and then passed to useState function to update state

**useContext Hook**

const {getCurrency,rate} = useContext(Context)

We are de-structuring to pluck required objects/variables/functions from the provider

**Alright enough with the definitions and background, let see some action**

import React, {useContext, useState} from "react"

import {Context} from "../Context"

function Search(){

    const [coin, setCoin] = useState({Coin:""})

    const {processCode} = useContext(Context)

    function handleChange(e){

        e.preventDefault()

        const {name, value} = e.target

        setCoin(prev=>{

            return{...prev,[name]:value}

        })

    }

    function handleSubmit(e){

        e.preventDefault()

        processCode(coin.Coin)

    }

    return(

        <div className="Rcontainer">

            <div className="row">

                <div className="col-sm-12 col-md-12 col-lg-12">

                    <form className="iform" onSubmit={handleSubmit}>

                        <input

                        type="text"

                        name="Coin"

                        value={coin.Coin}

                        placeholder="Enter crypto code ex. LTC, BTC"

                        onChange = {handleChange}

                        className="form-control fm1"

                        />

                        <button className="btn btn-primary bpad">Search</button>

                    </form>

                </div>

            </div>

        </div>

    )

}

export default Search

const addr = "https://api.nomics.com/v1/currencies/ticker?key=demo-26240835858194712a4f8cc0dc635c7a&ids="

function processCode(id){

        let upper = id.toUpperCase()

        let att = addr.concat(upper)

        setURL(att)

    }

const [url, setURL] = useState("")

useEffect(()=>{

        fetch(url)

            .then(res=>res.json())

            .then(data=>setCoinData(data))

    },[url])

const [coinData,setCoinData] = useState([])

const newCoinData = coinData.map(coin=>{

        return(

            <div className="dContainer" key={coin.name}>

                <div className="row dd">

                    <div className="col-sm-2 logo">

<img

     src={coin.logo\_url}

     alt="Coin logo"

     height={30}

    width={30}/></div>

                    <div className="col-sm-6 price">$ {coin.price}</div>

                    <div className="col-sm-4 name">{coin.name}</div>

                </div>

            </div>

        )

    })

<Context.Provider

        value={{

            processCode,

            newCoinData,

            moreCoinData,

            extraData,

            countryInfo,

            processCountry,

            flag,

            moreData,

            getCurrency,

            rate,

            metal

            }}>

            {props.children}

 </Context.Provider>

import React, {useContext} from "react"

import {Context} from "../Context"

import Search from "../components/Search"

import '../App.css';

function Crypto(){

    const {newCoinData, moreCoinData, extraData} = useContext(Context)

    //console

    return(

        <div className="container">

            <div className="row">

                <div className="col-lg-5 box"><Search /></div>

                <div className="col-lg-5 box ">{newCoinData}</div>

            </div>

  </div>

    )

}

export default Crypto

**DEMO**

**HOME PAGE**

**A screenshot of a cell phone

Description automatically generated**

**CURRENCY CONVERTOR**

**A screenshot of a cell phone

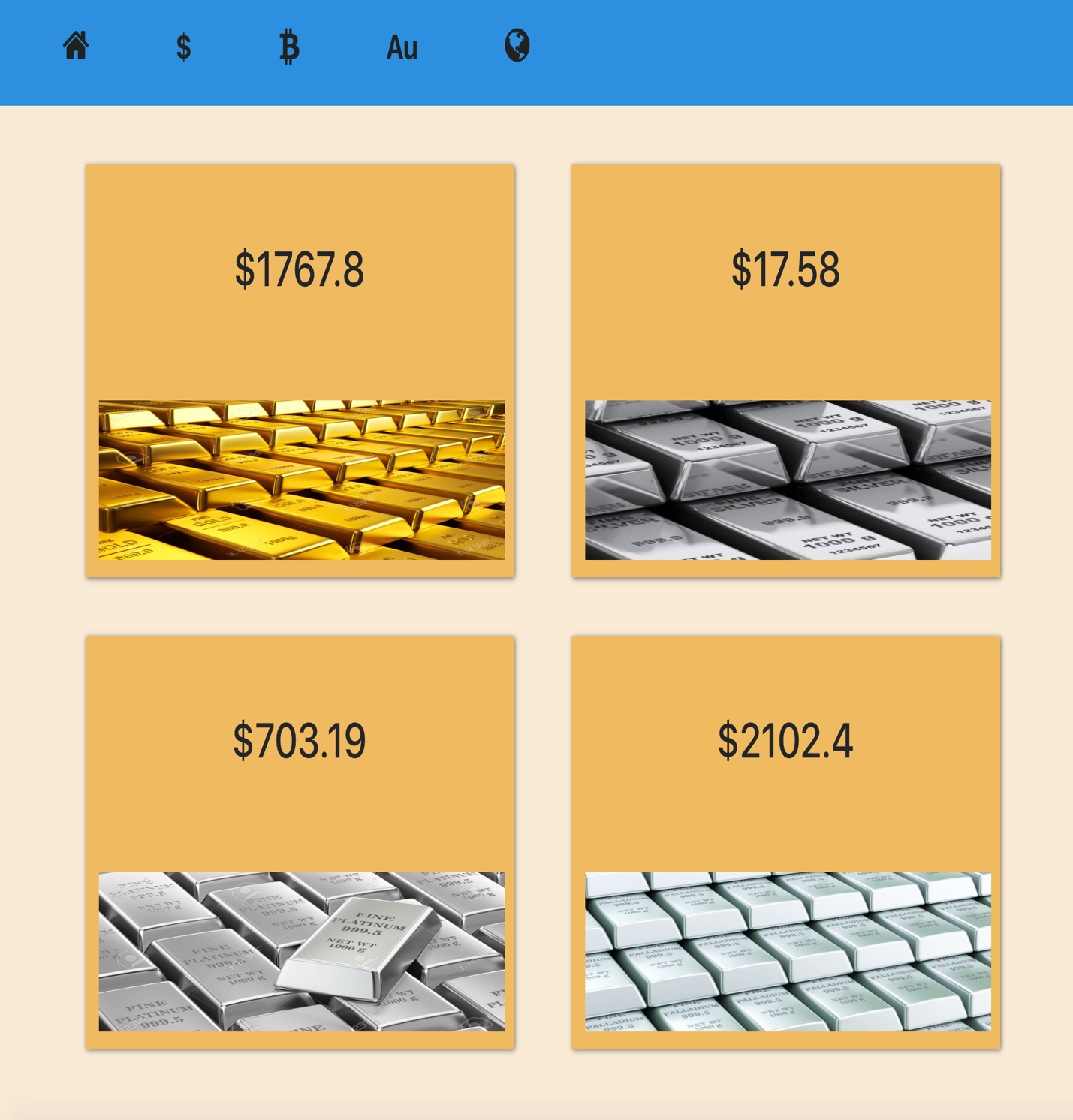
Description automatically generated**

**CRYPTO CURRENCY**

**A screenshot of a cell phone

Description automatically generated**

**PRECIOUS METALS**

****

**COUNTRIES DATA**

**A screenshot of a cell phone

Description automatically generated**

**FIN**