**Team Name:**  ENVISION

**Team Logo:**

****

**Team Members:**

Alekya pochampally

Manideep Chamala

Suma Soma

**Visualization of Real-time trades for stocks**

**Steps**

**Step 1:**

This is the website URL <https://finnhub.io/> from where we will be collecting data stream real-time trades for US stocks, forex and crypto.

**Step 2**: Verifying Live data

Of all Stream real-time trades for US stocks, forex and crypto, we will be showing only two Symbols data in our project.

Bitcoin

Amazons

**Step 3:** Web Sockets

We would create a web socket using the URL.

**this.socket = new WebSocket('wss://ws.finnhub.io?token=bujpqa748v6tkps41v00');**

**Step 4**: Now we would create web hook to bind the data to the chart created in the application.

**Step 5:** whenever new data is reflected in the website, using add event listener we would get last updated prices for trades.

**Step 6:** For each symbol there are multiple data records, we would take only two fields time and price of trade.

**Step 7**: using node JS and react we would show the real time data for those two trades in a web application.

**Data Source:**

This is the website URL <https://finnhub.io/> from where we will be collecting data stream real-time trades for US stocks, forex and crypto.

Create an API key in Finnhub.io that will use for authenticating requests.

Using symbols for trades we could collect the data from the URL.

**Number of records and columns:**

**Response Attributes:**

**Data**

List of trades or price updates.

**S**

Symbol.

**P**

Last price.

**T**

UNIX milliseconds timestamp.

**V**

Volume.

**C**

List of trade conditions.

**VISUALIZATION:**

For visualization of data, we will be creating a web Application using React (Front end) and node JS for backend.

Using Canvas JS Chart we are representing the real time last updated data for two trades Bitcoin and Amazon.

We are using three different charts to represent the live data.

1. Spline Line chart
2. Line Chart
3. Step Line Chart

We would hit the API by creating web sockets for the URL specifying the symbols of both trades.

Adding event listener to the socket, whenever there is an update in the data it would hit the URL and gets the latest prices for the trades.

Now we are extracting only necessary data i.e. price and time from the JSON response.

Then creating hook to socket we could render the data to the chart.

**Sample code for Line Chart:**

import React, { Component } from 'react';

import CanvasJSReact from '../assets/canvasjs.react';

var CanvasJSChart = CanvasJSReact.CanvasJSChart;

class LineChart extends Component {

constructor(props) {

super(props);

this.state = {

options: {

animationEnabled: true,

exportEnabled: true,

theme: "dark1",

title: {

text: "Line chart"

},

axisY: {

title: "USD",

prefix: "$",

valueFormatString: '#########.###',

includeZero: false

},

data: [{

name: "Binance Bitcoin",

type: "line",

showInLegend: true,

options: {

maintainAspectRatio: false

},

toolTipContent: "{x}: ${y}",

dataPoints: [

]

},

{

name: "Amazon",

type: "line",

showInLegend: true,

options: {

maintainAspectRatio: false

},

toolTipContent: "{x}: ${y}",

dataPoints: [

]

}

]

},

isHookActive : false

}

this.unsubscribe = this.unsubscribe.bind(this);

this.setWebHook = this.setWebHook.bind(this);

}

componentDidMount(){

this.setWebHook();

}

setWebHook() {

this.socket = new WebSocket('wss://ws.finnhub.io?token=bujpqa748v6tkps41v00');

// Connection opened -> Subscribe

this.socket.addEventListener('open', function (event) {

this.socket.send(JSON.stringify({ 'type': 'subscribe', 'symbol': 'BINANCE:BTCUSDT' }));

this.socket.send(JSON.stringify({ 'type': 'subscribe', 'symbol': 'AMZN' }));

}.bind(this));

// Listen for messages

this.socket.addEventListener('message', function (event) {

var response = JSON.parse(event.data);

console.log(response);

if(response && response.data && response.data[0]){

var data = response.data[0];

if(data && data.s === "BINANCE:BTCUSDT"){

this.state.options.data[0].dataPoints.push({ x: new Date(new Date(data.t).toISOString()), y: parseFloat(data.p) });

}else if(data && data.s === "AMZN"){

this.state.options.data[1].dataPoints.push({ x: new Date(new Date(data.t).toISOString()), y: parseFloat(data.p) });

}

}

if(this.chart){

this.chart.render();

}

}.bind(this));

this.setState({isHookActive:true,options:this.state.options});

}

unsubscribe() {

this.socket.send(JSON.stringify({ 'type': 'unsubscribe', 'symbol': 'BINANCE:BTCUSDT' }));

this.socket.send(JSON.stringify({ 'type': 'unsubscribe', 'symbol': 'AMZN' }));

this.socket.close();

this.setState({isHookActive:false,options:this.state.options});

}

render() {

var primaryButtonStyle ={ marginTop : '20px', borderRadius: '16px', border: '1px', padding: '4px 12px', background: '#64ad6a', cursor: "" };

var secondaryButtonStyle ={ marginLeft: '10px', marginTop : '20px', borderRadius: '16px', border: '1px', padding: '4px 12px', background: '#dd3439', cursor: "" };

primaryButtonStyle.cursor = this.state.isHookActive ? "default" : "pointer";

secondaryButtonStyle.cursor = this.state.isHookActive ? "pointer" : "default";

return (

<div style={{marginTop:"20px"}}>

<CanvasJSChart options={this.state.options}

onRef={ref => this.chart = ref}

/>

<div style={{textAlign:"center"}}>

<button style={primaryButtonStyle} onClick={() => { this.setWebHook() }} disabled={this.state.isHookActive}>Subscribe</button>

<button style={secondaryButtonStyle} onClick={() => { this.unsubscribe() }} disabled={!this.state.isHookActive}>Unsubscribe</button>

</div>

</div>

);

}

}

export default LineChart;

**Steps to run the Application:**

We need to install node JS in our system.

Open the project folder in windows explorer, from there open command prompt.

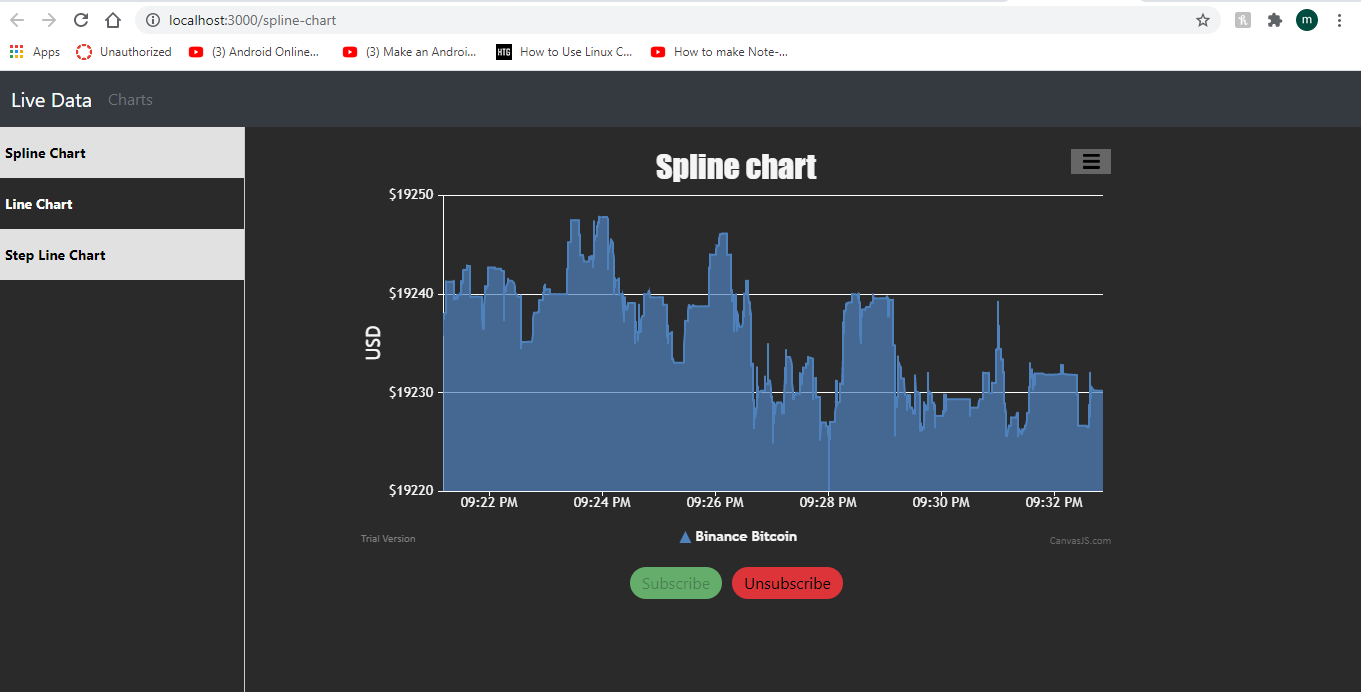
Run the following commands:

npm install;

npm run build;

npm start;

Now an application starts in our localhost at 3000 port as the below image.

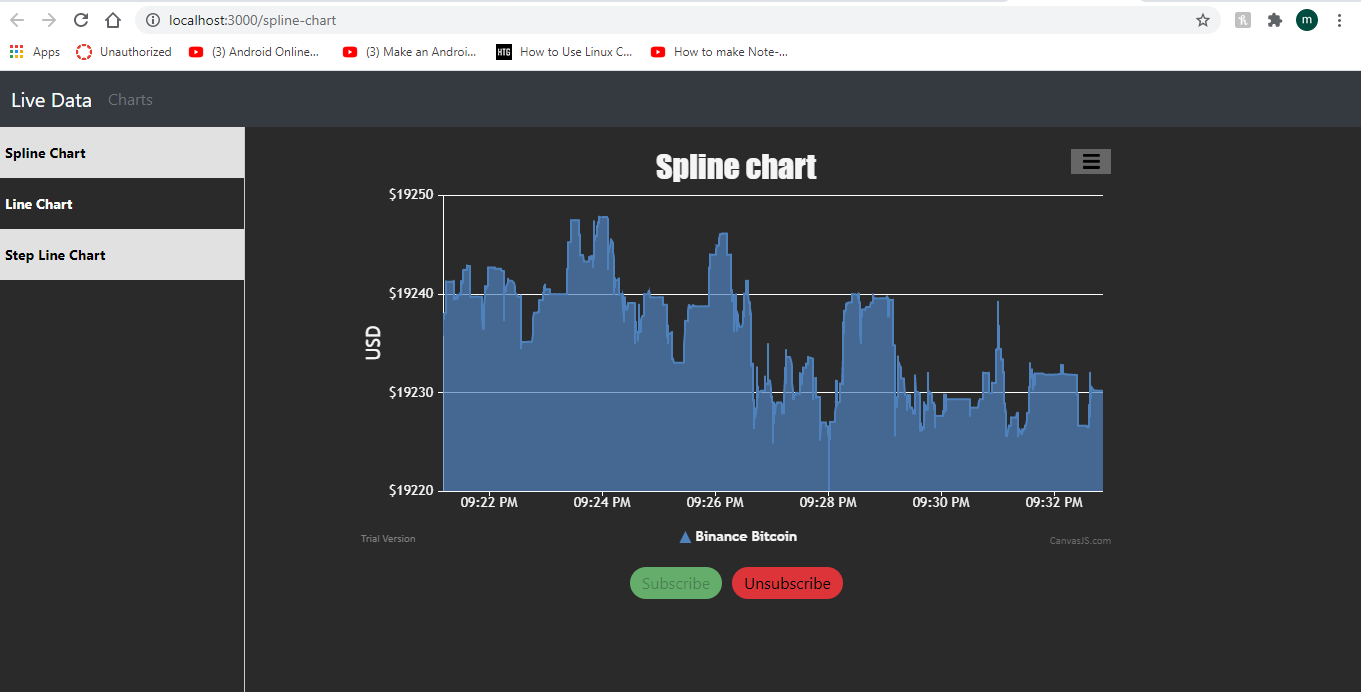


**GOAL**

To collect live streaming data, convert it to structured data and visualize it with different charts.

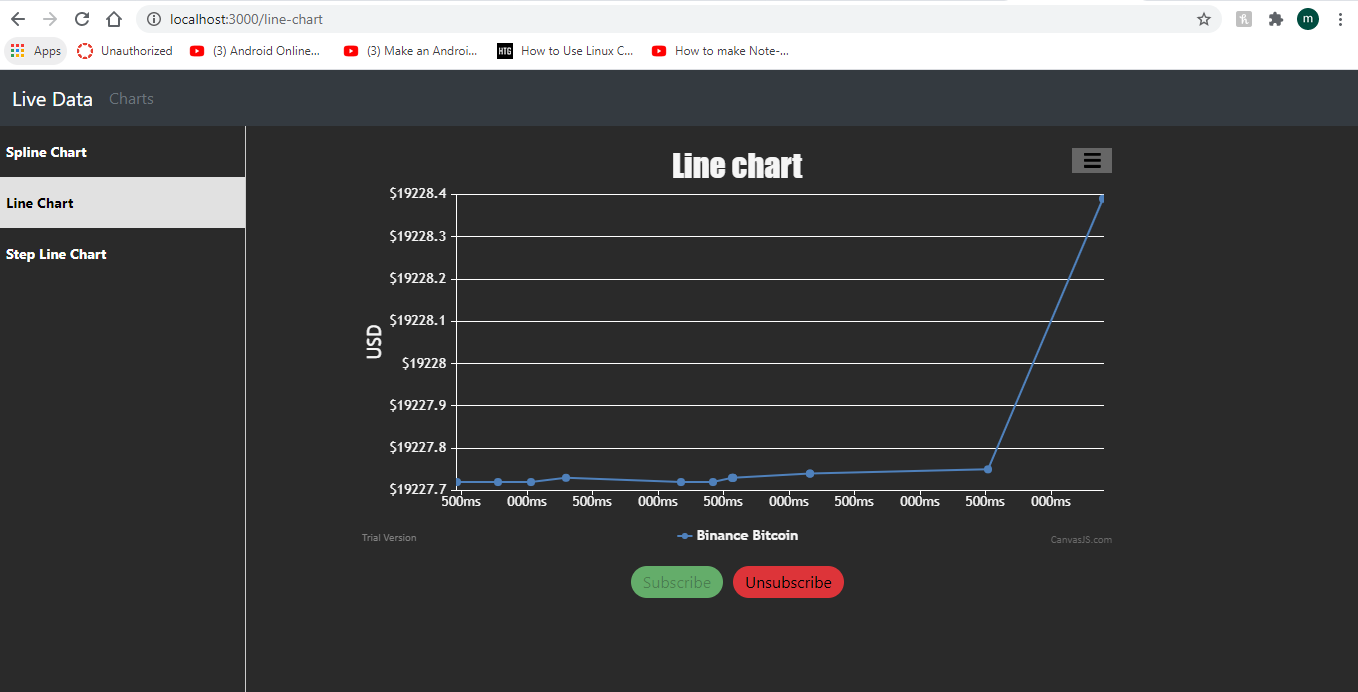
**Goal 1:**

Collecting the live data of price and last updated time for trades, representing it using Spline line chart.



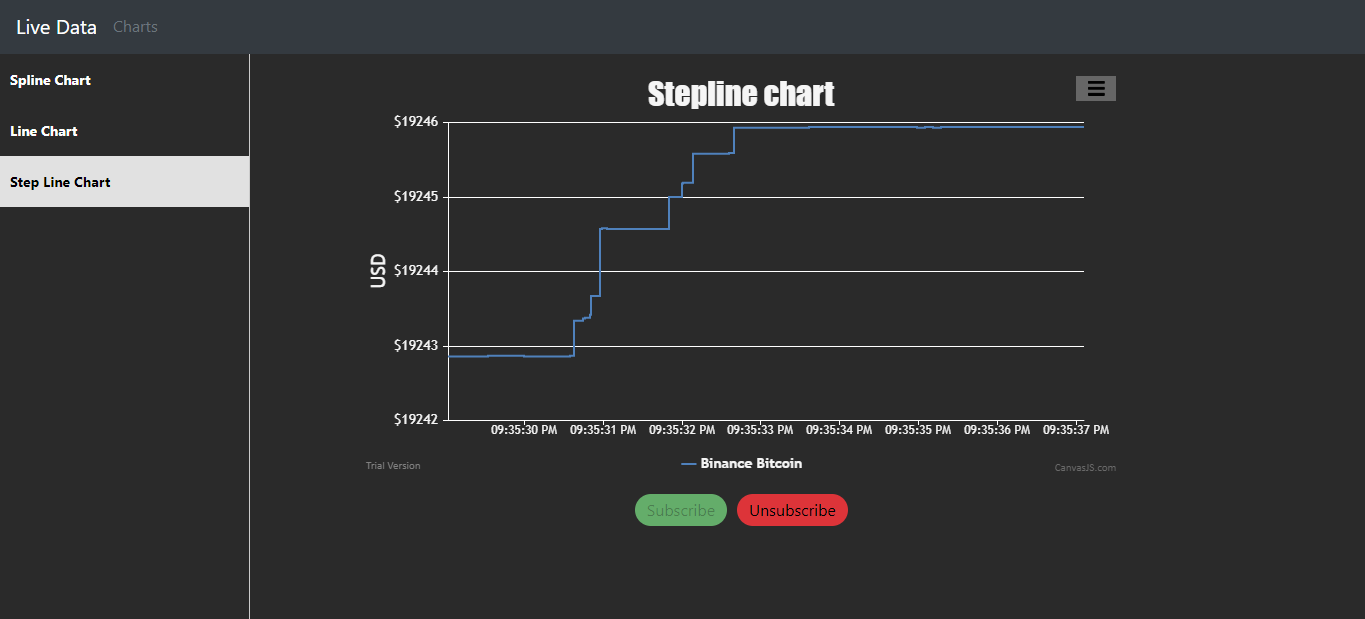
**Goal 2:**

Collecting the live data of price and last updated time for trades, representing it using Line chart.



**Goal 3:**

Collecting the live data of price and last updated time for trades, representing it using Step line chart.



**STORY:**

* From the above three graphs, we can see that there is increase and decrease in the value of price for every minute.
* Every minute after refreshing data we can clearly see that there is a fluctuation in the value of price for trades.

**CONCLUSION:**

In a few steps, collecting live streaming data, convert it to structured data and visualize it with different charts live dashboard using React App. While it may normally take a longer minutes, using event listeners in this React App allows users to perform live analysis on trade data within seconds of the events occurring.