Quickstart guide

Introduction

The Token Details API demo will walk you through how users can seamlessly retrieve and display token price changes from the past week.

Prerequisites

- Node.js and npm installed
- Basic knowledge of JavaScript, React, and Express.js

Set up environment variables

To securely manage your API key, store it as an environment variable. For macOS:

1. Open the Terminal.
2. Add `API_KEY` variable to your shell configuration file. Replace `your_api_key` with your DevPortal API key:

```
export API_KEY=your_api_key
```

3. Save and reload your configuration file:

```
source ~/.zshrc   # For zsh (default for macOS)
```

Step 1: Initialization

1. Create a new directory for the project:

```
mkdir token-details && cd token-details
```

2. Initialize a new Node.js project:

```
npm init -y
```

3. Install Express, CORS, and Axios:

```
npm install express cors axios
```

4. Create a new file `api.js` and set up a basic Express server by pasting into it the following:

```
const express = require("express");
const axios = require("axios");
const cors = require("cors");
const path = require("path");

const app = express();
const PORT = 5001;

app.use(cors()); // To handle CORS issues when making requests to the front end

// Serve static files from the React app
app.use(express.static(path.join(__dirname, "client/build")));

// Add endpoint handler here

// We will route all other requests to the client build
app.get("*", (req, res) => {
  res.sendFile(path.join(__dirname, "client/build", "index.html"));
});

app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

5. Add an endpoint to fetch token prices:

```
const BASE_URL = "https://api.1inch.dev/token-details/v1.0/charts/interval";
const CHAIN_ID = 1; // Eth

app.get("/api/:tokeAddress/prices/:interval", async (req, res) => {
  const { tokeAddress, interval } = req.params;

  try {
    const constructedUrl = `${BASE_URL}/${CHAIN_ID}/${tokeAddress}?interval=${inter
val}`;

    const response = await axios.get(constructedUrl, {
      headers: {
        Authorization: `Bearer ${process.env.API_KEY}`,
      },
    });
```

```
    // Send the data from the API back to the client
    res.json(response.data.d);
  } catch (error) {
    console.error("Axios Error: ", error.response);
    res.status(500).json({ error: "Failed to fetch token price by interval" });
  }
});
```

Step 2: Setting up React frontend

1. Create a new React application:

```
npx create-react-app client
```

2. Navigate to the React application directory and install React Charts:

```
cd client
npm i react-charts
```

3. Create a component `TokenPrice.js` inside the `src` directory with the following content:

```
import React, { useState, useEffect } from "react";
import { Chart } from "react-charts";

const TokenPrice = ({ address, interval }) => {
  const [tokenPrices, setTokenPrices] = useState([]);
  const [isLoading, setLoading] = useState(false);

  const primaryAxis = React.useMemo(
    () => ({ getValue: (datum) => datum.date }),
    [],
  );

  const secondaryAxes = React.useMemo(
    () => [{ getValue: (datum) => datum.price }],
    [],
  );

  useEffect(() => {
    const fetchData = async () => {
      setLoading(true);

      const response = await fetch(
        `http://localhost:5001/api/${address}/prices/${interval}`,
      );
```

```
      if (!response.ok) {
        console.log("Fetch token prices error", response);
        return;
      }

      const prices = await response.json();

      setTokenPrices(prices);
      setLoading(false);
    };

    if (isLoading) {
      return;
    }

    fetchData();
  }, [address, interval]);

  if (isLoading) {
    return <div>Loading...</div>;
  }

  if (tokenPrices.length === 0) {
    return <div>No data available</div>;
  }

  return (
    <Chart
      options={{
        data: [
          {
            data: tokenPrices.map((d) => ({
              date: new Date(d.t * 1000),
              price: d.v,
            })),
          },
        ],
        primaryAxis,
        secondaryAxes,
      }}
    />
  );
};

export default TokenPrice;
```

4. Import and use `TokenPrice` in `src/App.js`:

```
import "./App.css";
import TokenPrice from "./TokenPrice";

function App() {
  return (
    <div
      className="App"
      style={{
        width: 800,
        height: 800,
      }}
    >
      <h1>Token Price</h1>
      <TokenPrice
        address="0x111111111117dC0aa78b770fA6A738034120C302"
        interval="7d"
      />
    </div>
  );
}

export default App;
```

Step 3: Running the project

1. Start the Express server:

```
node api.js
```

2. In a new terminal, navigate to the client directory and start the React app:

```
cd client
npm start
```

Now, you can view your 1inch token price over last week at http://localhost:3000.

Described above is a basic setup and you can expand upon this by adding more features, error handling, and styling to get to production.

Previous
‹ Introduction

Next
Returns details for native token ›