### **1. Web Scraping Tool Overview:**

This web scraping tool is built using Node.js and Express to create a server that exposes three API endpoints for data retrieval. The primary purpose of the tool is to scrape cryptocurrency market data from the CoinMarketCap website, save it as JSON, and allow users to download the data or retrieve information about a specific coin.

### **2. Technologies Used:**

* Node.js: A JavaScript runtime that allows server-side execution of JavaScript code.
* Express: A web application framework for Node.js that simplifies the process of building robust APIs.
* Axios: A promise-based HTTP client for making requests to external resources.
* Cheerio: A lightweight and fast jQuery implementation for parsing HTML on the server side.
* Cors: Middleware for handling Cross-Origin Resource Sharing to enable requests from different origins.
* Path: A module for handling file paths in a platform-independent manner.
* File System (fs): A module for interacting with the file system.
* Promises (fs.promises): A set of utility functions to work with promises for file system operations.

### **3. Code Structure:**

#### a. **Server Setup:**

const express = require("express");

const cors = require("cors");

const path = require("path");

const PORT = 8888;

const app = express();

app.use(cors());

app.listen(PORT, () => {

console.log(`Server is running on ${PORT}`);

});

});

The server is set up using Express on port 8888 with CORS middleware to handle cross-origin requests.

#### b. **API Endpoints:**

// Endpoint to get overall market data

app.get("/api/getdata", async (req, res, next) => { /\* ... \*/ });

// Endpoint to get data for a single coin

app.get('/api/getsinglecoin/:name', async (req, res, next) => { /\* ... \*/ });

// Endpoint to download the market data as a JSON file

app.get("/api/download", async (req, res, next) => { /\* ... \*/ });

These endpoints handle requests for overall market data, data for a single coin, and downloading the entire dataset.

#### c. **Error Handling Middleware:**

app.use((err, req, res, next) => { /\* ... \*/ });

A generic error-handling middleware to catch and handle errors, providing a standardized response format.

#### d. **Web Scraping Functions:**

// Function to fetch the CoinMarketCap page

async function fetchPage() { /\* ... \*/ }

// Function to scrape data from the HTML

async function scrapeData(html) { /\* ... \*/ }

// Function to save scraped data as a JSON file

async function saveAsJSON(data) { /\* ... \*/ }

// Function to get data for a single coin from the saved JSON file

async function getSingleCoin(coinName) { /\* ... \*/ }

// Main scraping function orchestrating the entire process

async function scrapper() { /\* ... \*/ }

module.exports = { scrapper, getSingleCoin };

These functions are responsible for fetching the CoinMarketCap page, parsing HTML, saving data as JSON, retrieving data for a single coin, and orchestrating the entire scraping process.

### **4. Explanation of the Scraping Process:**

* Fetch Page: The fetchPage function uses Axios to make a GET request to the CoinMarketCap website and retrieves the HTML content of the page.
* Scrape Data: The scrapeData function utilizes Cheerio to parse the HTML and extract relevant data from the cryptocurrency market table on the CoinMarketCap page.
* Save as JSON: The saveAsJSON function takes the scraped data and writes it to a JSON file in the 'data' folder. If the folder doesn't exist, it is created.
* Get Single Coin: The getSingleCoin function reads the saved JSON file and returns data for a specific coin based on the provided coin name.
* Main Scraper Function: The scrapper function acts as the main orchestrator, calling the above functions in sequence. It fetches the page, scrapes the data, saves it as JSON, and returns the scraped data.

### **5. Why This Approach:**

* Modularity: The code is modular, with separate functions for each step of the scraping process, making it easy to maintain and understand.
* Error Handling: The code includes error handling at various stages to ensure graceful degradation and meaningful error messages.
* Promises: The code extensively uses asynchronous operations and Promises for non-blocking execution, ensuring efficiency.
* Express Framework: Express simplifies the creation of APIs, making it an ideal choice for building a web server.
* Axios and Cheerio: Axios is used for making HTTP requests, and Cheerio simplifies HTML parsing, providing a lightweight alternative to a full-fledged DOM parser.
* Data Storage: The scraped data is saved as a JSON file for easy retrieval, and the folder structure is organized for clarity.

### **6. Conclusion:**

In summary, this web scraping tool efficiently retrieves cryptocurrency market data from CoinMarketCap, stores it as JSON, and provides a simple API for users to access the data or download the entire dataset. The chosen technologies and coding practices aim to ensure readability, maintainability, and robust error handling.