Building a Real-time Stock Price Aggregator

Background

You are tasked with developing a backend system that aggregates real-time stock price data from the alpha vantage API and provides comprehensive reporting functionalities for analyzing stock trends. This exercise is designed to evaluate your advanced backend skills, including API integration, caching, database optimization, and data processing.

Requirements

Data Aggregation and Updating

- Integrate with the Alpha Vantage API to fetch real-time stock price data for a predefined set of stocks (e.g: Apple, Google, Microsoft).
- Choose at least 10 stocks.
- Implement and automated mechanism to fetch the stock price data at regular intervals (e.g: every 1 minute).

API Consumption and Error Handling

- Perform HTTP requests to the Alpha Vantage API securely and efficiently.
- Implement error handling mechanisms to manage potential API rate limiting and connection issues.

Caching and Cache Updating

- Implement caching to store the latest stock price data for a short duration (e.g: 1 minute).
- Make sure to keep the cached data up-to-date.
- Implement and endpoint to fetch the latest stock price from the cache.

Database Schema and Performance

- Design a database schema optimized for storing and retrieving all the price data of the specified stock.
- Optimize database queries for efficient retrieval of stock data, considering frequent updates.

Real-time Reporting

Develop a reporting system that allows users to view real-time stock prices and percentage changes. Use the following formula to calculate the percentage change:

$$PercentageChange = \frac{Price_{Current} - Price_{Previous}}{Price_{Previous}} \cdot 100$$

Guidelines

1. Set up a new Laravel project.
2. Design the database schema and models to store real-time stock price data efficiently.
3. Fetch the data from the source and store in your database.
4. Implement caching mechanisms and make sure to keep your cache updated.
5. Optimize your queries for quick retrieval of data, ensuring data integrity.
6. Create a RestAPI that returns reports based on your data.

7. Thoroughly test your application, using manual and automated tests, considering API responses, caching behavior, and job processing.
8. Document the project's setup, your design decisions , and provide clear instructions for running the application.

Submission

Share a link to your git repository containing the Laravel project. Include comprehensive instruction for setting up and running the project locally, along with detailed documentation.

Bonus

1. Containerize the application using Docker & Docker compose.
2. Design and implement a user interface that shows the latest stock price with visual indicators (e.g: color-coded arrows) to display positive or negative changes.