# فهرست

## طرح سوال:

You are tasked with designing and implementing a high-performance system for handling real-time market data.

Requirements

Your system should:

1.Consume market data feeds

- Simulate the feed with random price updates for multiple symbols.

2.Calculate moving averages in real-time

- For each symbol, maintain a moving average of the latest N price updates.

3.Detect price anomalies

- Identify and report price spikes greater than 2% within any 1-second interval.

4.Handle high throughput

- Efficiently process 10,000+ price updates per second.

5.Utilize concurrent programming

- Use thread-safe collections and async/await patterns to ensure scalability and correctness.

Task

Describe your approach and provide code samples for the main components.

Explain your design choices, especially regarding concurrency and performance.

---

Please organize your answer using the provided structure

- Problem Understanding
- High-Level System Design
- Detailed Design
- Code Implementation
- Testing & Validation
- Performance & Scalability Considerations
- Summary

## درک مسئله:

سیستم با مشخصات زیر تولید شود:

1. **هزاران قیمت در ثانیه** برای نمادهای مختلف دریافت کند.
2. برای هر نماد:
   - **میانگین متحرک زنده** محاسبه کند (مثلاً از آخرین 20 قیمت).
   - اگر در **۱ ثانیه نوسان بالای ٪۲** رخ داد، هشدار بدهد.

### اجزای کلیدی:

**دریافت سریع داده‌های بازار**

- باید بتوانیم قیمت‌های زیاد (مثلاً 10,000+ در ثانیه) را هم‌زمان از منابع دریافت کنیم.
- این کار شبیه‌سازی یک بازار واقعی است (مثل بازار بورس یا رمزارزها).

**محاسبه میانگین متحرک**

- فقط آخرین N قیمت (مثلاً 20 عدد) برای هر نماد نگهداری می‌شود.
- برای سرعت، به جای محاسبه مجدد کل میانگین، از **مجموع تجمعی** استفاده می‌کنیم.

**تشخیص ناهنجاری (مثلاً جهش قیمت)**

- اگر قیمت جدید بیش از ٪۲ با قیمت‌های یک ثانیه گذشته تفاوت داشت، هشدار بده.
- نیاز به حافظه‌ای داریم که قیمت‌های اخیر با زمان‌شان را نگه دارد و سریع بررسی کند.

### نکات مهم در طراحی:

**کارایی بالا(High Throughput)**

**برنامه‌نویسی هم‌زمان و مقیاس‌پذیر**

**مصرف حافظه بهینه**

**تاخیر پایین(Low Latency)**

**قابل توسعه بودن(Scalable Design)**

## Entity ها سامانه:

برای پیاده سازی این سامانه ما entity ها زیر رو داریم

نمادها:

```
public Guid Id

public string Name

public DateTime CreatedAt

public DateTime? UpdatedAt
```

قیمت نماد ها:

```
public Guid Id

public Guid SymbolId
public Symbol Symbol

public decimal Price

public DateTime RecordedAt
```

تاریخچه ای قیمت نماد ها:

```
public Guid Id

public Guid SymbolId

public decimal Price

public DateTime RecordedAt
```

Moving Average

```
public Guid Id

public Guid SymbolId

public decimal AveragePrice
public DateTime CalculatedAt
```

## بخش ها

از چند زیربخش تشکیل شده که با هم تعامل دارند:

### 1.Data Feed Subsystem (زیرسامانه تولید داده)

تولید دیتا تستی(شبیه سازی دیتا)

- تولید قیمت‌های تصادفی برای نمادهای مختلف با نرخ بالا.

در لایه `Infrastructure` یک کلاس `MarketDataFeedService` وجود دارد

دو تابع دارد

تابع `GenerateSymbol` نماد ها تستی رو ایجاد میکنند

تابع `GenerateRandomPrices` قیمت نماد ها تستی رو ایجاد میکنند

دو تا endpoint دارد

اول `GenerateSymbol`

لیست نمادهای برمیگرداند

دوم`GetRandomPrices`

قیمت نمادها رو تولید میکنند

2.بروز رسانی قیمت های نمادهای

در پروژه MarketFeedProcessorUI یک بک گراند MarketFeedProcessor داریم

ابتدا سرویس مربوط به api شبیه سازی رو فراخوانی میکنند و سپس تابع HandlePriceUpdateAsync در کلاس MarketDataService فراخوانی میکنند ،در این متد نماد و قیمت در دیتابیس و کش ذخیره میکنند

**3.Moving Average Engine**

میانگین قیمت نماد های

در پروژه MarketFeedProcessorUI یک endpoint داریم در کنترل SymbolAverageController و api ای GenerateSymbol برای محاسبه میانگین

**4.Anomaly Detection Engine**

شناسایی انحراف قیمت نمادها

در پروژه MarketFeedProcessorUI یک endpoint داریم در کنترل SymbolAverageController و api ای anomalies برای شناسایی انحراف قیمت نمادها

- مقایسه قیمت‌ها طی ۱ ثانیه اخیر.
- بررسی درصد تغییر برای شناسایی نوسان <٪۲.

## قابلیت ها سامانه:

استفاده از serilog برای ثبت لاگ ها

استفاده از EF برای ذخیره نمادهای در دیتابیس sql server

استفاده از redis برای کش نماد و قیمت و ...

استفاده از ConcurrentDictionary, ConcurrentBag برای دسترسی امن در پردازش همزمان

استفاده از Parallel.ForEachAsync و Task.WhenAll برای افزایش کارایی در پردازش

استفاده از xunit و Moq برای تست نویسی

## پیشنهاد برای توسعه آینده:

برنامه طوری بیاد نوشته شود اگر چند نود داشته باشیم پس باید لیست نماد ها رو بین نودهای تقیسم شود تا هر نود تعدادی رو پردازش کنند چون دیتا در کش ردیس هست این قابیلت را دارد اما در زمان پردازش ، نماد ها بین نودها تقیسم شود در صورت امکان از الگو inbox pattern یا قابلیت message broker برای اینکار استفاده شود