Project Objective:

This project will track real-time price movements of two leveraged ETFs, TQQQ (which tracks the NASDAQ-100 index with 3x leverage) and SQQQ (which tracks the same index but with -3x leverage), calculate the dollar change between consecutive price updates, and analyze the market trend (bullish or bearish). If the market data is unavailable or stopped, the system will stop updating dollar calculations and display appropriate alerts.

Mathematical Concepts Applied:

1. Dollar Change Calculation:

We will calculate the dollar change between the previous and current stock prices to determine how much the price has moved.

- **Formula:**

Dollar Change=Current Price−Previous Price\text{Dollar Change} = \text{Current Price} - \text{Previous Price}

For example:

- Previous Price: $50
- Current Price: $52

Dollar Change = $52 - $50 = **$2** (indicating an increase of $2 in price).

2. Percentage Change (Rate of Change):

To understand the price change in relative terms, we will calculate the percentage change. This helps to quantify how much the price has increased or decreased in terms of percentage.

- **Formula:**

Percentage Change=(Current Price−Previous PricePrevious Price)×100\text{Percentage Change} = \left(\frac{\text{Current Price} - \text{Previous Price}}{\text{Previous Price}}\right) \times 100

For example:

- Previous Price: $50
- Current Price: $52

Percentage Change = (52−50)50×100=4%\frac{(52 - 50)}{50} \times 100 = 4\%

This shows a 4% increase in the price of TQQQ.

3. Market Trend (Bullish or Bearish):

To determine if the market is bullish or bearish, we check the price movement of TQQQ and SQQQ:

- **Bullish Trend:** TQQQ's price increases, and SQQQ's price decreases.
- **Bearish Trend:** SQQQ's price increases, and TQQQ's price decreases.

4. Handling No Data (Market Stoppage):

If there is no data from the API (e.g., market closure, weekend, or API failure), the system will stop updating the calculations. We'll use timestamps to track the last successful data retrieval and stop calculations after a defined period of no updates.

Project Breakdown (15 Days)

Day 1-2: Initial Setup

- Set up the ASP.NET project (MVC/Web API) and create a simple SQL Server database schema.
- **Database Schema:**
  - **StockPrices Table:** Stores stock prices of TQQQ and SQQQ.
    - Columns: `Id`, `StockSymbol`, `Price`, `Timestamp`
  - **PriceChanges Table:** Stores the calculated dollar and percentage changes.
    - Columns: `Id`, `StockSymbol`, `DollarChange`, `PercentageChange`, `Timestamp`
  - **MarketStatus Table:** Tracks whether the market is bullish or bearish.
    - Columns: `Id`, `MarketStatus`, `Timestamp`
  - **ApiLogs Table:** Stores API request logs (successful and failed attempts).
    - Columns: `Id`, `ApiCallTimestamp`, `Status`

Day 3-4: Integrate API for Real-Time Data

- Choose an API like Alpha Vantage, Yahoo Finance, or IEX Cloud to fetch real-time TQQQ and SQQQ prices.
- Create a service in ASP.NET to interact with the external API and fetch real-time prices of TQQQ and SQQQ.
- Example endpoint for Alpha Vantage: `/query?function=TIME_SERIES_INTRADAY&symbol=TQQQ&interval=1min&apikey=your_api_key`

Day 5-6: Dollar and Percentage Change Calculation

- Once the real-time prices are fetched, calculate the dollar and percentage changes:
  - **Dollar Change**: Subtract the previous price from the current price.
  - **Percentage Change**: Calculate the percentage change using the formula above.
- Store the calculated values in the **PriceChanges Table**.
- Example C# Code to calculate dollar and percentage change:

```
public decimal CalculateDollarChange(decimal currentPrice, decimal previousPrice)
{
    return currentPrice - previousPrice;
}

public decimal CalculatePercentageChange(decimal currentPrice, decimal previousPrice)
{
    return ((currentPrice - previousPrice) / previousPrice) * 100;
}
```

Day 7-8: Market Trend Detection

- Based on the price movement of TQQQ and SQQQ, determine if the market is **Bullish** or **Bearish**:
  - **Bullish Market**: TQQQ price increases, and SQQQ price decreases.
  - **Bearish Market**: SQQQ price increases, and TQQQ price decreases.

- Implement a function to detect the trend:

```
public string DetermineMarketTrend(decimal tqqqPriceChange, decimal sqqqPriceChange)
{
    if (tqqqPriceChange > 0 && sqqqPriceChange < 0)
        return "Bullish";
    else if (sqqqPriceChange > 0 && tqqqPriceChange < 0)
        return "Bearish";
    else
        return "Neutral";
}
```

- Store the detected trend in the **MarketStatus Table**.

Day 9-10: Stop Updates if No Data (Market Closed)

- Check the timestamp of the last data update in the **StockPrices Table**. If no new data is fetched within a specified time (e.g., 30 minutes), stop the calculations and display an alert that no new data is available.
- Use a **Background Task** to check for data availability periodically (e.g., every 5 minutes).
- Example Code to check if data is stale:

```
public bool IsDataStale(DateTime lastUpdatedTime)
{
    return (DateTime.Now - lastUpdatedTime).TotalMinutes > 30; // 30 minutes timeout
}
```

Day 11-12: Frontend Development

- Create an **Index.cshtml** (or React component) to display the real-time data, including:
  - TQQQ and SQQQ prices.
  - Dollar and percentage change.
  - Market trend (Bullish/ Bearish).
- Use AJAX or WebSockets to fetch new data from the backend every minute without refreshing the page.
- Example HTML:

```
<div id="tqqq-info">
    <h2>TQQQ Price: $<span id="tqqq-price">0.00</span></h2>
    <h3>TQQQ Dollar Change: $<span id="tqqq-dollar-change">0.00</span></h3>
    <h3>TQQQ Percentage Change: <span id="tqqq-percent-change">0%</span></h3>
</div>
<div id="market-status">
    <h3>Market Trend: <span id="market-trend">Loading...</span></h3>
</div>
```

Day 13-14: Data Update Mechanism

- Create an **AJAX call** in JavaScript to get the latest TQQQ and SQQQ prices and update the page every minute:

```
function fetchStockData() {
    $.ajax({
        url: '/Stock/GetLatestData',
        method: 'GET',
        success: function(data) {
            $('#tqqq-price').text(data.tqqqPrice);
            $('#tqqq-dollar-change').text(data.tqqqDollarChange);
            $('#tqqq-percent-change').text(data.tqqqPercentChange);
            $('#market-trend').text(data.marketTrend);
```

```
            }
        });
    }

    setInterval(fetchStockData, 60000); // Update every minute
```

Day 15: Final Testing and Deployment

- Test the entire application for:
  - Real-time price updates.
  - Correct calculation of dollar change and percentage change.
  - Accurate market trend detection.
  - Handling of API failure or market closure.
- Deploy the application to a live server (e.g., Azure or AWS) and make sure everything works smoothly.

Folder Structure for ASP.NET Project:

```
/StockMarketProject
│
├── /Controllers
│   ├── StockController.cs
│   ├── MarketController.cs
│
├── /Models
│   ├── StockPrice.cs
│   ├── PriceChange.cs
│   ├── MarketStatus.cs
│
├── /Services
│   ├── StockPriceService.cs
│   ├── MarketTrendService.cs
│   ├── ApiLogService.cs
│
├── /Views
│   ├── /Home
│   │   ├── Index.cshtml
│
├── /wwwroot
│   ├── /js
│   │   ├── app.js
│   ├── /css
│       ├── styles.css
```

Conclusion:

This project will allow you to track the price changes of TQQQ and SQQQ, calculate dollar and percentage changes, and analyze market trends (bullish or bearish). The project will automatically stop updating if no data is available due to market closure or API failure. We will use mathematical formulas for calculating price changes, and the backend will integrate with a real-time stock price API, while the frontend will be updated dynamically using AJAX. The project will be developed in ASP.NET MVC/Web API with a SQL Server database to store and manage data efficiently.