**Tech Stack**: .NET Core (backend), React (frontend), Kafka (real-time data streaming), Redis (caching), WebAPI

**Features**:

* Users can create a portfolio by adding stocks.
* Real-time stock price updates using Kafka.
* Historical data storage using Redis for quick access.
* API integration with stock market data providers (e.g., Alpha Vantage, Yahoo Finance).
* Notifications on price changes or portfolio value fluctuations

**Features & Functionality**

**1️.User Portfolio Management**

* Users can **create an account** and log in.
* They can **add stocks** they own along with purchase price and quantity.
* System calculates **portfolio value** based on real-time stock prices.

**2️.Real-Time Stock Price Updates**

* Fetch real-time stock prices using an **external API** (e.g., Alpha Vantage, Yahoo Finance, or a mock data feed).
* Use **Kafka** to handle real-time price updates and push them to the frontend.
* Store recent stock prices in **Redis** for quick access.

**3️.Price Alerts & Notifications**

* Users can set a **target price** for each stock.
* If the stock price crosses the threshold, send an alert (email, push notification via AWS SNS).

**4️.Performance Insights & Visualization**

* Show **historical price trends** using charts (React with Chart.js or D3.js).
* Compare current portfolio value vs. **initial investment**.
* Calculate metrics like **daily % change, overall gain/loss, and dividend income**.

1)Tables/Entities :- Using EntityFramework

1.1)User Table

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id(auto generated) | Int(PK) | Unique User id |
| FirstName | nvarchar(20) | User first name |
| LastName | nvarchar(20) | User Last Name |
| Email | Nvarchar(50) unique | User Unique Email |
| Password | Nvarchar(50) Bcrypt | User Password encrypted |
| CreatedAt | datetime | Account creation Date |
| UpdatedAt | datetime | Account update Date |

1.1)User Entity:-

-id , firstname , lastname

-email ,password

-List<Portfolio>

1.2)Portfolio Table:-

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id(auto generated) | Int(PK) | Unique Portfolio id |
| userId | Int(FK) | Foreign key reference to user |
| Name | nvarchar(20) | Portfolio Name |
| CreatedAt | datetime | Portfolio creation Date |
| UpdatedAt | datetime | Portfolio update Date |

1.2)Portfolio Entity:-

-id,Name,userId

-user(Navigation property)

-List<Stock>

1.3)Stock Table:-

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id(auto generated) | Int(PK) | Unique User id |
| PortfolioId | Int(fk) | Foreign Key to portfolio table |
| Name | nvarchar(20) | Stock Name |
| Ticker | Nvarchar(50) | Ticker symbol of stock |
| PurchasedPrice | Decimal(18,2) | Purchase price of at given time |
| Quantity | Decimal(10,2) | Purchased stock quantity |
| PurchasedAt | datetime | Stock Purchase Date |

1.3)Stock Entity:-

-id ,PortfolioId ,Name ,Ticker

-Quantity, PurchasePrice , PurchasedAt

-PortFolio(Navigation property)

1.4)StockPrices:-

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id(auto generated) | Int(PK) | Unique Portfolio id |
| Ticker | Nvarchar(50) | Ticker symbol of Stock |
| CurrentPrice | Decimal(18,2) | Latest Price of stock |
| LastUpdatedAt | datetime | Portfolio update Date |

1.4)StockPrice Entity:-

-id ,ticker

-CurrentPrice

-Last Updated

1.5)Notification Table:-

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id(auto generated) | Int(PK) | Unique Notification Id |
| userId | Int(FK) | Foreign key reference to user |
| Ticker | nvarchar(20) | Ticker of target stock |
| TargetPrice | Decimal(18,2) | Target Price For alert |
| NotificationType | Nvarchar(15) | Type(email,sms) |
| SentAt | datetime | Notification Sent Time |

1.5)Notification Entity:-

-id ,UserId, Ticker

-TargetPrice, NotifactionType,SentAt

-User(Navigation property)

2)Relationships:-

2.1️)One User → Many Portfolios  
2️.2)One Portfolio → Many Stocks  
2.3)One Stock Symbol → Many Price Records  
2.4)One User → Many Notifications

|  |
| --- |
| Portfolio |
| Add Portfolio |
| Update Portfolio ? |
| Delete Portfolio? |
| Get All Portfolios |
| Get Portfolio |

|  |
| --- |
| Stocks |
| Add Stock |
| Update Stock ? |
| Delete Stock? |
| Get All stocks |
| Get Current price |

|  |
| --- |
| Statistics |
| Price Trend |
| Investment |
| % change |
| Profit Loss |
|  |

|  |
| --- |
| User |
| Get User |
| Update User ? |
| Delete User ? |

3)Api

|  |
| --- |
| Auth |
| Login |
| Register |

Use Case:-

1. **Register:** Allows a new user to create an account within the system.
   * *Description:* The user provides their personal information (e.g., name, email, password) to create an account and gain access to the system's features.
2. **Login:** Allows an existing user to authenticate and access the system.
   * *Description:* The user enters their credentials (e.g., email, password) to verify their identity and gain access to their portfolio and other features.
3. **Create Portfolio:** Allows a user to create a new portfolio to track their stock holdings.
   * *Description:* The user provides a name for the portfolio, and the system creates a new portfolio associated with the user's account.
4. **View Portfolio:** Allows a user to view the details of a specific portfolio, including its name, the stocks it contains, their quantity, purchase price, and current market value.
   * *Description:* The user selects a portfolio to view, and the system displays the portfolio's details.
5. **Add Stock to Portfolio:** Allows a user to add a stock to a specific portfolio.

*Option1:-*. **Detailed Entry (Current Approach)**

**Description:** The user manually enters all the details of the stock, including the ticker symbol, company name, quantity, and purchase price.

**Pros:**

-Simple to implement (relatively).

-Allows for maximum user control.

**Cons:**

-Prone to errors due to manual data entry.

-Requires the user to know the ticker symbol and company name.

-Less user-friendly than other options.

*Option 2:-* **Search and Select**

**Description:**The user searches for a stock by ticker symbol or company name, selects it from a list of results, enters the quantity, and the system automatically calculates the price based on the current market price.

**Pros:**

-Reduces data entry errors.

-User-friendly.

-Provides real-time pricing information.

**Cons:**

-Requires integration with a Stock Data Provider API.

-Requires dynamic search functionality in the frontend.

*Option 3:-* ***Pre-Populated List with Search***

**Description*:***The system presents a list of the most popular stocks (or stocks from a specific sector), and the user can either scroll through the list or use a search box to filter the list and select a stock.

**Pros:**

-User-friendly.

-Reduces data entry errors.

- for users who are familiar with popular stocks but don't know the ticker symbols.

**Cons:**

-Limited to stocks in the pre-populated list.

-Requires pagination or infinite scrolling to handle large lists.

-Still requires integration with a Stock Data Provider API (for fetching the current price).

1. **Remove Stock from Portfolio:** Allows a user to remove a stock from a specific portfolio.
   * Description*:* The user selects the stock to remove, and the system removes it from the portfolio.
2. **View Real-Time Stock Price:** Allows a user to view the current price of a specific stock.
   * *Description:* The user provides the ticker symbol for the stock, and the system fetches the real-time price from the Stock Data Provider and displays it to the user.
3. **View Historical Stock Price Chart:** Allows a user to view a chart of historical stock prices for a specific stock.
   * *Description:* The user provides the ticker symbol and a date range, and the system fetches the historical prices from its database and displays them in a chart.
4. **Set Price Alert (Future):** (This use case is marked as future because it's not yet implemented.) Allows a user to set a target price for a specific stock and receive a notification when the price crosses the threshold.
   * *Description:* The user provides the ticker symbol, target price, and notification method (e.g., email), and the system sets up a price alert.
5. **Receive Notification (Future):** (This use case is also marked as future.) The user receives a notification when a price alert is triggered.
   * *Description:* The system monitors the stock price and sends a notification to the user when the price crosses the threshold.

Stock Api:-

Alpha Vantage :-

-5 API requests per minute and 500 requests per day.