Visual Studio (VB, C#, C++)

Grading Rules

1. Do not place your name or your school anywhere in your code or data. Use only your team number (when needed). Failure to do so will cause you to be disqualified.

2. Code that has errors and will not run will not be judged. You can comment out non-working code for the judges to view, if there is a need for a tie breaker.

3. Applications can be developed in one of three approaches: Web, Desktop, or Console. In a tie breaker situation, Web will be considered the most complete, followed by Desktop, and lastly Console.

4. Applications that have more than one approach (Web, Desktop, Console) will be judged as more complete in a tie breaker situation.

5. Applications should be created using the three tier design process: Data, Presentation, Business Logic (Model/View/Controller). All working applications will be judged. However, applications using three tiers will receive more points.

6. The time you complete this assignment will also be used as a tie breaker.

7. All projects must be done using a current version of Visual Studio (at least 2015) using only Visual Studio Tools. (no third party tools such as ReSharper)

8. All projects can contain additional packages from the Nuget Package Manager. No other frameworks / libraries / packages are allowed from any sources.

9. All projects must use VB.NET, C#.NET, or C++.NET

10. All projects must use a .Net framework version of 4.5.1 or higher

11. Web code should be developed using ASP.NET 4.5 or 5

12. HTML code can be developed using XHTML or HTML5

13. Internet access is allowed for research, but you must write your own code.

Problem Statement

Midwest Financial is a company that among other things manages stock portfolios for various clients. Your team has been tasked with creating any application (Web Application, Desktop, Console, or a combination of the three), which will allow the financial advisor to keep track of their client's stock portfolio.

The client list, and stock list for each client is stored in a database. See the database section for more information about the database.

The stock prices are fetched from a web service. See the web service section for more information about the web service.

The only user of the system is the Financial Advisor, and is referred to from here on out as 'user'. The user does not need to authenticate to get access to the application.

Your application should implement the following Use Cases:

1. The user can view a list of the clients that they advise, which should include each client's First Name, Last Name, and Cash Balance.
2. The user can add a new client to keep track of.
3. The user can remove an existing client as long as all of their stock assets have been sold, and as a result, their total asset value is zero.
4. The user can view the portfolio for a single client, whose portfolio will include a list of the stocks the client has purchased, the asset value of each purchased stock, and a total asset value for all purchased stocks.
5. The user can add funds to a client's cash balance.
6. The user can remove funds from a client's cash balance.
7. The user can use funds from a client's cash balance to purchase stock. This will decrease the cash balance, and increase the total asset value.
8. The user can sell stocks from a client's stock asset portfolio to decrease the total asset value, and increase the client's cash balance.

Every time the user looks at a portfolio, the program should use the most recent stock price for each stock to calculate the asset value for each stock, and the total value of all assets.

You do not need to worry about keeping track of the operations that occur. You are not recording each transaction. You are only applying the associated math related to each transaction. You also do not need to concern yourself with figuring out how much money has been made by buying and selling either.

You should choose one approach, either Web, Desktop, or Console, and complete the program for one approach before attempting a second approach.

Assumptions: Make any assumptions necessary that are not included in this description. Include these assumptions (as a Word, text, or PDF document) with your solution.

Documentation: Include any user documentation needed. Also make sure to use proper comments within the code. Include any special instructions that may be needed to ensure the judges can run your program correctly.

Databases

You will be given a database file to use in conjunction with this application.

**DO NOT CREATE ANY OTHER DATABASES OR ANY OTHER TABLES ON THIS DATABASE. You can do everything with the database provided.

Clients:

- Id (int) [auto-incrementing]
- FirstName (string)
- LastName (string)
- CashBalance (decimal)

Stocks:

- Id (int) [auto-incrementing]
- Client_Id (int) [foreign key on Clients]
- CompanyTicker (string)
- Quantity (int)

There is a one to many relationship between clients and stocks where one Client can have zero to many Stocks.

You are free to use any way you would like to connect and interact with the database so long as it falls within the guidelines listed above in the Grading Rules.

Web Service

The following operations will require a current stock price to do the following calculations:

1. Determining how much to decrement the cash balance when stock is purchased. This should also pull the company name to store in the DB.
2. Determining how much to increment the cash balance when a stock is sold.
3. Calculating the client's entire asset value, which will change very often due to the variability of stock prices from one minute to the next.

In order to get the current stock price for any given stock, the program will query a web service by a stock ticker symbol. The results of the query are returned as a JSON string. From this string, you can get the current stock price per share, and the name of the company.

The web service end point can be accessed by making an HTTP request to the following location where {TICKER_SYMBOL} is a valid stock ticker symbol:

https://api.iextrading.com/1.0/stock/{TICKER_SYMBOL}/quote

Documentation for the API, and the specific end point you need can be found here:

https://iextrading.com/devloper/docs/#quote

If you would like to see what the returned data looks like, you can enter in the endpoint with a valid stock ticker symbol into a web browser, and retrieve the same JSON response that will be returned when you make the request in code.

If an unknown symbol is entered, the server will simply return 'Unknown symbol'.

The 2 pieces of information that you need to extract from the JSON data is the companyName, and latestPrice.

The companyName will be used when the list of purchased stocks are displayed. That way the user does not have to have all of the stock tickers and what company they are associated with memorized.

The latestPrice will be used for calculating how much the total value of all purchased stocks is, how much to increment and decrement the cash balance when stock is purchased and sold.

Grading Breakdown

Use Case	
Display list of clients	5
Add new client	5
Delete client with no stock	5
Add funds to client cash balance	5
Remove funds from client cash balance	5
View client portfolio and total of assets	10
Purchase a stock for a client	10
Sell a stock for a client	10
Technical	
Connect to the database	5
CRUD on the database	5
Pull data from web service	5
Parse data from web service	5
Proper OO design	5
Documentation	
Application	10
Code	10
Total	**100**