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 2017) using only Visual Studio Tools. (no third party tools such as ReSharper)

8. All projects can contain additional packages from the Nuget Package Manager, or packages that students have created themselves. 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 provides currency exchange rates between US Dollars and other currencies. Your team has been tasked with creating any application (Web Application, Desktop, Console, or a combination of the three), which will allow a user to keep a list of investments they have in other currencies. The app will also show the value of each of their investments in US Dollars as well as a total for all investments in US Dollars.

The investment list will be stored in a database. See the **Database** section for more information about the database.

The exchange rates are fetched from a web service. See the **Web Service** section for more information about the web service.

There is only one user of the system, and all investment information is for the sole 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 investments that they have in other currencies and what the US Dollar amount is.
2. The user can see a total for all of their investments in US Dollars.
3. The user can add funds to their cash balance in US Dollars.
4. The user can remove funds from their cash balance in US Dollars.
5. The user can use funds from their cash balance to exchange for a different currency. This will decrease the cash balance, and either create or update the amount invested for that particular currency.
6. The user can exchange investments back to US Dollars to decrease the total investment for that foreign currency, and increase the users cash balance. If the user converts an entire investment back to US Dollars the entry for that foreign currency should be removed from the database.

Every time the user looks at their investments, the program should use the most recent currency exchange rate for each investment. It should also show the total value of all investments converted to US Dollars.

You do not need to worry about keeping track of the transactions that occur nor the amount of money that has been made since adding an investment. It is the users job to know how much they invested and if they have made any money.

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 text document with your solution.

Documentation: Include any user documentation needed. Include any special instructions that may be needed to ensure the judges can run your program correctly. This should be in some sort of README file. Also make sure to use proper comments within the code.

Database

You can download the database for this competition by going to the following Github link and either cloning or downloading the repository. The database is a SQL Server database file with the extension .mdf and a log file with the extension of .ldf. You will need them both.

https://github.com/dbarneskvcc/MWC3_2019_VS_Database

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

Investments:

Id (int) [auto-incrementing]
Code (string) (unique) – Foreign Code
Amount (decimal) – Foreign Amount

Use a row with the code USD for storing the cash balance. Adding investments will decrement from this row and add / update a different one.

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

Some hints that might help you out but are not required:

1. Use the DataDirectory functionality in your connection string.
2. Set the DataDirectory to a path that is relative so that the judges do not have to change the connection string in order to get the app to run. This can be achieved by using and setting something like:
   ```
   AppDomain.CurrentDomain.SetData("DataDirectory", Environment.CurrentDirectory);
   ```
3. Ensure that the database files are located in the same location that Environment.CurrentDirectory points to.

Web Service

The following operations will require a current exchange rate to do the following calculations:

1. Determining how much to create or update an investment in a foreign currency.
2. Determining how much to remove or decrement an investment in a foreign currency.
3. Calculating the value in US Dollars of an investment.
4. Calculating the total value in US Dollars of all investments.

In order to get the current exchange rate for any given currency, the program will query a web service by both the native currency code (USD) and a foreign currency code. The results of the query are returned as a JSON string. From this string, you can get the exchange rate that will be used to covert from one currency to the other.

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

https://api.exchangeratesapi.io/latest?base=USD&symbols={SYMBOL}

Documentation for the API, can be found here:

https://exchangeratesapi.io/

If you would like to see what the returned data looks like, you can enter in the endpoint with a valid currency 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 return an error JSON response that looks like the following:

```
{"error":"Symbols 'ZZZ' are invalid for date 2019-03-22."}
```

The returned rate will be used to calculate how much of a foreign currency to add or remove. It will also be used to calculate investment values in USD.

A list of currency codes that can be used for testing can be found here:

https://www.xe.com/symbols.php

Grading Breakdown

Use Case	
Display list of investments and value in both foreign and USD	10
View total of investments in USD	5
Add funds to cash balance	10
Remove funds from cash balance	10
Exchange some cash balance for investment in foreign currency	10
Exchange foreign currency for some cash balance	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**