

Team 5R

Assignment 3 – Currency Broker Alert

Software Design Document

Steven Theck

Ryan Epstein

TABLE OF CONTENTS

1. INTRODUCTION	3
1.1 Purpose	3
1.2 Scope	3
1.3 Overview	3
1.4 Reference Material	3
1.5 Definitions and Acronyms	3
2. SYSTEM OVERVIEW	4
3. SYSTEM ARCHITECTURE	5
3.1 Architectural Design	5
3.2 Decomposition Description	5
3.3 Design Rationale	5
4. DATA DESIGN	6
4.1 Data Description	6
4.2 Data Dictionary	6
5 COMPONENT DESIGN	7
6 User INTERFACE DESIGN	10
6.1 Overview of User Interface	10
6.2 Screen Images	10
7 REQUIREMENTS MATRIX	11

1. INTRODUCTION

1.1 Purpose

The purpose of this software design document is to describe the design of the Currency Broker Alert Center. It is aimed to users who are monitoring currency exchange rates.

1.2 Scope

The Currency Broker Alert Center is for a simple program that will notify the user if an exchange rate between a pair of currencies hits a target rate.

1.3 Overview

Provide an overview of this document and its organization. This document contains the necessary information to understand how the system was built, including a detailed description of the system's architecture, design, and an overview of the system itself.

1.4 Definitions and Acronyms

CBAC – *Currency Broker Alert Center* (name of the software)

Bid: the current rate of a currency pair

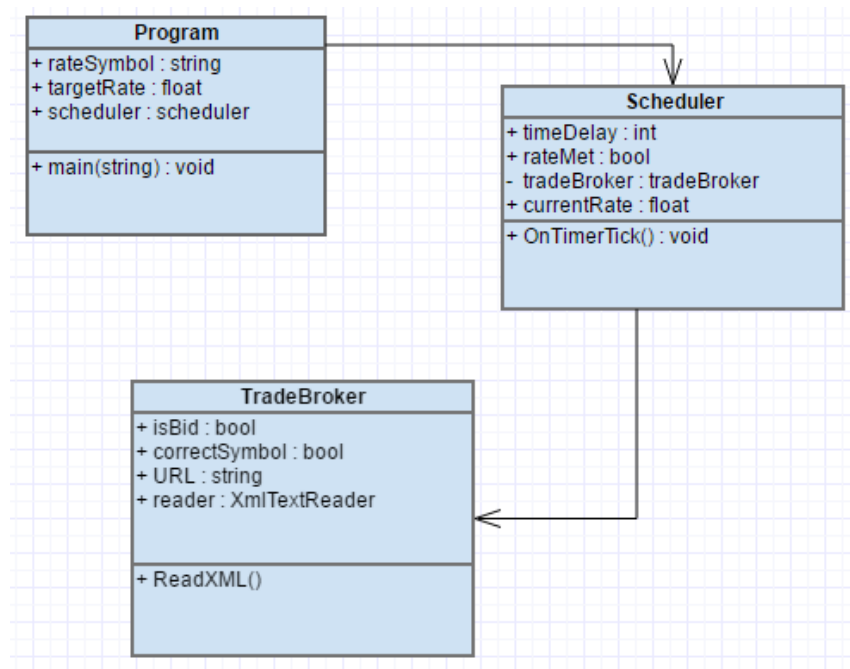
Ask: the suggested sell rate

2. SYSTEM OVERVIEW

The CBAC system is built to notify the user when an exchange rate of a currency pair hits a desired rate. The user will be prompted to enter a pair of stocks and a target rate for the pair to reach before being notified. CBAC will gather current trade rates from a webpage and finds the stock pair and stores its information. "Target has been reached!" is printed to the screen when the target value is reached.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design



3.2 Decomposition Description

Refer to UML above.

The Program will get input from the user to determine what stock and associated rate they are want to be notified on. The Scheduler intermittently (every 60000 milliseconds) runs the TraderBroker. The TradeBroker class gathers information from the XML regarding the stocks and confirms the user inputted rate versus the current rate of the stock indicated by the XML.

3.3 Design Rationale

We chose to run the scheduler in 1 minute intervals which is a trade off because the user will have slightly longer times before being notified if a stock is at a target rate but we did not want to make requests to the server too frequently.

4. DATA DESIGN

4.1 Data Description

Data is being pulled from the URL into XML form at every tick rate and is being compared against the user's input in real time.

4.2 Data Dictionary

Variables:

- rateSymbol (string): The symbol to indicate the stock pair the user is looking for
- targetRate (float): The user's target rate
- timeDelay (int): A time in milliseconds to indicate the tick rate of the program
- rateMet (bool): A flag to indicate if the target rate is met
- currentRate (float): The stocks current rate
- isBid(bool): A flag to indicate if the XML element is the stock's bid
- correctSymbol (bool): A flag to indicate if the user's stock pair is valid in the XML

Functions:

- OnTimerTick(): every time interval, it monitors a stock's rate until it reaches the target rate.
- ReadXML(): reads the rates from the webpage

5. COMPONENT DESIGN

```
class Program
{
    String rateSymbol
    float targetRate

    void Main(string[])
    {
        Initialize scheduler class
        Get stock pair from user
        Store stock pair

        While input is invalid (target rate less than 0)
        {
            Get target rate from user
            Store target rate
            if target rate is invalid
            {
                Print "invalid rate"
            }
        }
        do
        {
            while user has not escaped the program
            {
                Run scheduler
            }
        } while user has not pressed Esc key
    }
}
```

```

class Scheduler
{
    int timeDelay = 60000
    bool rateMet
    Initialize tradeBroker class

    void OnTimerTick()
    {
        Read XML and get current rate of stock
        while currentRate is invalid
        {
            Print "invalid pair"
            Get stock pair from user
            Store stock pair from user
            Get current rate for stock pair
        }
        if current rate has been met
        {
            Print "Target has been reached!"
            Set rateMet to true
        }
        if current rate was met but has changed and no longer satisfies target
        {
            Set rateMet to false
        }
        Sleep for timeDelay seconds
    }
}

```



```

class TradeBroker
{
    float ReadXML()
    {
        bool correctSymbol
        bool isBid
        String URLString
        XmlTextReader reader

        while XML is being read
        {
            switch
            {
                case XML is an element
                    if XML element is the Rate
                    {
                        while XML is still reading
                        {
                            if XML value is equal to user input for stock symbol
                            {
                                Set correctSymbol to true
                            }
                            else
                            {
                                Set correctSymbol to false
                            }
                        }
                    }
                    if XML element is Bid
                    {
                        Set isBid to true
                    }
                    else
                    {
                        Set isBid to false
                    }
                    break
                case XML is text
                    if the user symbol is valid and the XML is currently reading bid
                    {
                        return the value of the Bid
                    }
                    break
                case XML is the end of an element
                    break
            }
        }
        return 0
    }
}

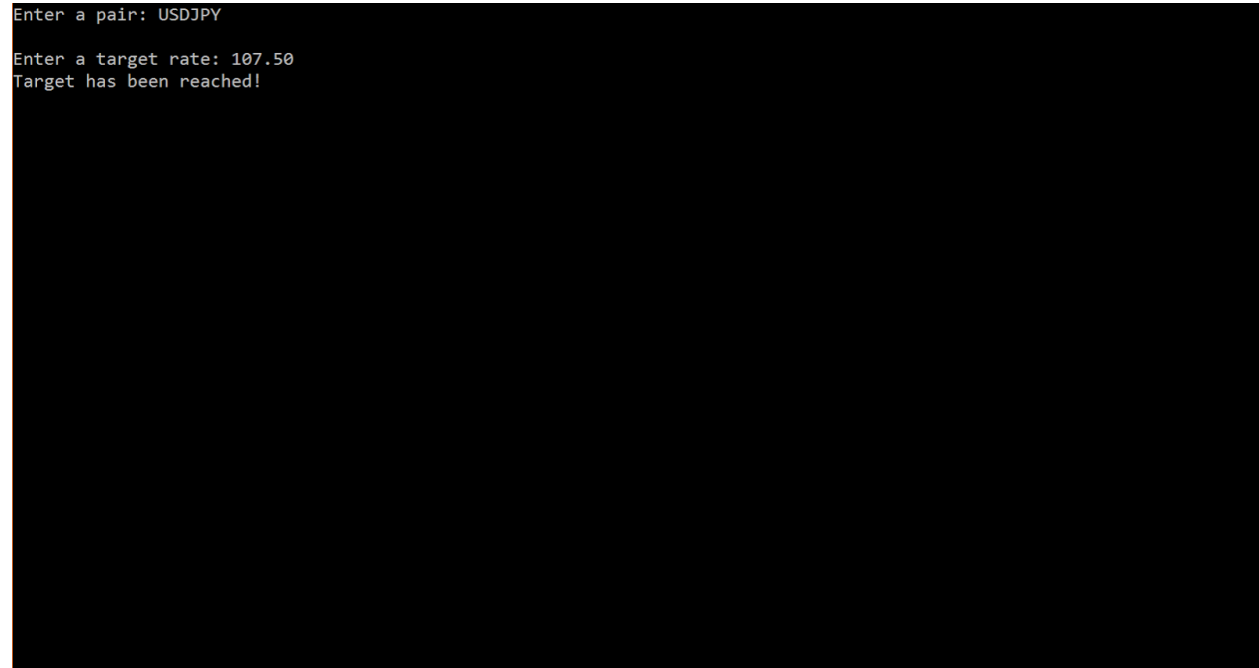
```

6. User INTERFACE DESIGN

6.1 Overview of User Interface

The program starts by prompting the user for a symbol to represent a pair of stocks. Then is prompted for a valid rate that they want to be notified if the rate reaches.

6.2 Screen Images



```
Enter a pair: USDJPY
Enter a target rate: 107.50
Target has been reached!
```

7. REQUIREMENTS MATRIX

1. You will need a parser to parse the XML. There are several examples online.
2. Start with hard coded user configuration. For example. Pair: EURUSD, Target rate: 1.381
3. Make it user entry once you have working code.
4. You will periodically check the rates, consider using a scheduler.
4. For notification, we will assume that there is a system that will be called to notify users. You only display on the screen if target has been reached.