Srinivasa_visvanathan

# Currency Rate Checker

Software Design Document

Names:

Adithya
Srinivasa,

Manju
Visvanathan

Date: (04/28/2017)

# TABLE OF CONTENTS

**1. INTRODUCTION 2**
1.1 Purpose 2
1.2 Scope 2
1.3 Overview 2
**2. SYSTEM OVERVIEW 2**
**3. SYSTEM ARCHITECTURE 3**
3.1 Architectural Design 3
3.2 Decomposition Description 4
3.3 Design Rationale 4
**4. DATA DESIGN 5**
4.1 Data Description 5
4.2 Data Flow Diagram 5
**5. HUMAN INTERFACE DESIGN 6**
5.1 Overview of User Interface 6
5.2 Screen Images 6
**6. REFERENCES 9**

# 1. INTRODUCTION

## 1.1 Purpose

This software design document describes the architecture and system design of a Currency Rate Checker application. This document will serve as a reference to any queries regarding the design of this application.

## 1.2 Scope

This application notifies the user when the currency pair rate reaches the target rate. The pair rate and target rates are obtained from a dynamic XML Document that keeps track of all the required data.
This application has a good scope in financial organizations that deal with flow of money between different countries, and organizations that are either investing or making a profit in multiple countries. This application also helps in stock market analysis in real time.

## 1.3 Overview

This document is organized into the following major parts:

- System Overview
- System Architecture
- Data Design
- Human Interface Design

# 2. SYSTEM OVERVIEW

This is a system that notifies them when a currency pair rate reaches the target rate. User configures the currency pair and target rate.

The URL to get the Rates in XML form: "http://rates.fxcm.com/RatesXML"

Example of the rate:
<Rate Symbol="EURUSD">
<Bid>1.38022</Bid>
<Ask>1.38042</Ask>
<High>1.38266</High>
<Low>1.37634</Low>
<Direction>0</Direction>

<Last>10:42:43</Last>
</Rate>

Here "Bid" is the current rate. "Ask" is the suggested sell rate. "High" is day high, "Low" is day low, "Direction" is 1=>up, -1=>down, 0=>no change compared to last rate, and "Last" is the time for last tick.

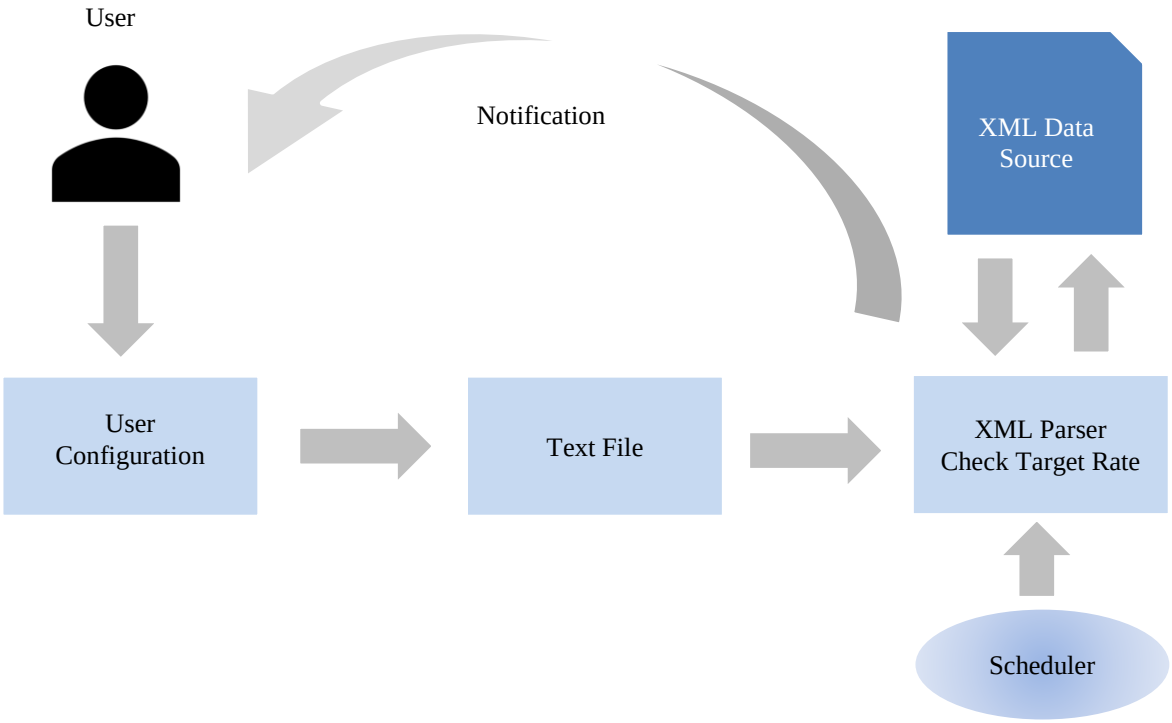
Functionality:

- The application takes input from the user. The input is the desired currency and the desired target rate, both are strings.
- Once the user gives input, the input data is stored in a text file. And the XML Parser module periodically checks if the users desired currency and rate reached his target rate.
- Java Scheduler has been used to periodically and automatically check if the currency pair rate reached the target rate.
- If the pair rate reaches the target rate, the user is notified by a window on the screen. If not, there is no notification.

# 3. SYSTEM ARCHITECTURE

## 3.1 Architectural Design

The below diagram outlines the architecture of the system using a modular program design.



As you can see above, the system has been modularized into several subsystems each of which perform a specific task.

The User Configuration subsystem is responsible for the Graphical User Interface and takes the input data from the user and dumps it into a text file.

The XML Parser module checks the XML source file periodically, using a scheduler

The Check Target Rates module also checks the user's desired target rate with the current version of XML File.

The Scheduler module makes sure that the XML Parsing and Rate Checking happen periodically with fixed intervals to reflect on any changes in the XML Source.

## 3.2 Decomposition Description

Here, we provide a decomposition of the subsystems in the architectural design.

**User Configuration**: In this module, the user input is accepted and stored in appropriate data structures which will be explained in detail in the data design. This module also is responsible for the User Interface Design and error checking by throwing exceptions for wrong user inputs.

**XML Parser**: This module is responsible for going to the URL Link that contains the XML file and parsing the data which is stored in a certain format. Since we know the format in which the data is stored, the XML Parser module can be designed to read the XML Document from the URL and store it in appropriate data structure.

**Check Target Rate:** This module compares the user data with the parsed data from the XML Source. If the desired target rate is reached, a notification window pops up on the user screen, if not, nothing happens.

**Scheduler:** Although this is not considered as a separate module, the role of scheduler is very important, in this module we import the required packages from the java library and create an object from the java scheduler class and use the functions there to periodically run the XML Parser and the Check Target Rates module after every 30 seconds.

## 3.3 Design Rationale

The rationale behind choosing this architectural design was:

**Simplicity:** This design is intuitive and simple to understand and develop a code.

**Data Flow:** In this architecture, we can see that there is a clear demarcation between flow of data. We know which module uses what data and what data is moving between modules. This ease of data modelling makes this design a good choice.

# 4. DATA DESIGN

## 4.1 Data Description

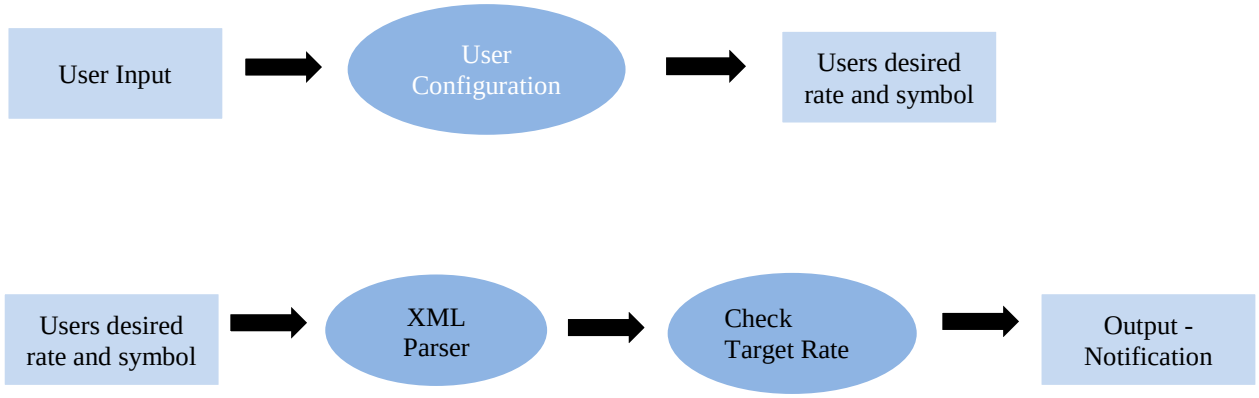
The various modules and subsystems store the data in a systematic way. Here, we see how the how the information domain of the system is transformed into data structures. Also, the details about how the major data or system entities are stored, processed, and organized.

The User Configuration module mainly involves the extraction of users input. The user first chooses the symbol which are stored in the array of strings data structure. The currency pair and the desired target rate of the user is stored as Array List <string>. This module then does error checking and GUI designing using Java Swing.

The XML Parser scans each line of the XML file to extract the required data only which are within the XML Tags. The data stored in this module are the currency pair symbol and the bid rate which are stored in the Array List <string> data structure.

The Check Target Rates module contains the currency pair rate and the target rate as Array List <string> and it compares this value to the value entered by thee user which is stored in the text file.

## 4.2 Data Flow Diagram



# 5. HUMAN INTERFACE DESIGN

## 5.1 Overview of User Interface

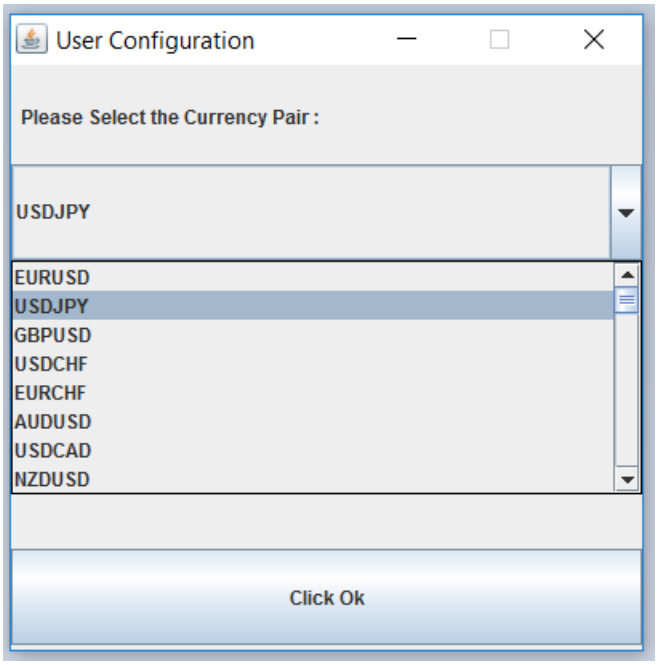
This section talks about the application from the user's perspective. The user will see a GUI window which is developed using the Java Swing APIs. When the user executes the user configuration module, the user will see a window asking the symbol (currency) and his desired currency rate (target rate).

The GUI will take these input values and validate whether to accept the input or not based on the correctness of the data that is given by the user. This makes the application robust.

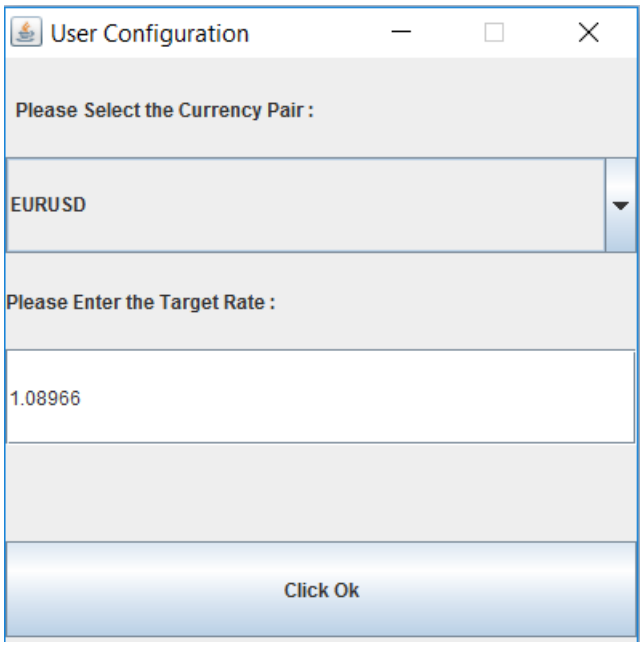
When the target rate has been reached, the user sees it on the window. If the user input is wrong, the user sees it on the window. So, the User Interface adds a lot of simplicity and robustness to this application.

## 5.2 Screen Images

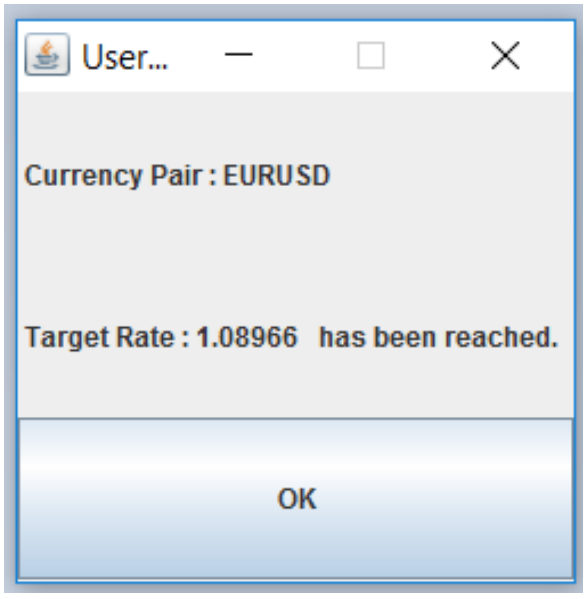
**List of Currency Pair:**



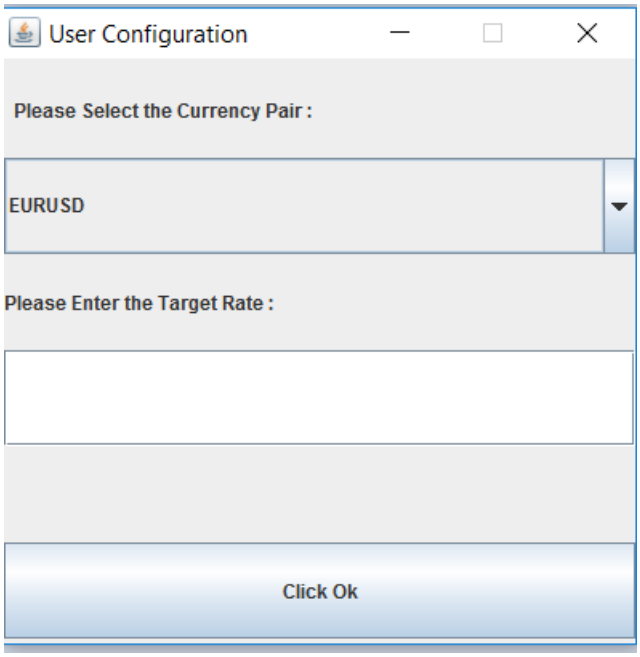
**Correct Input Value:**



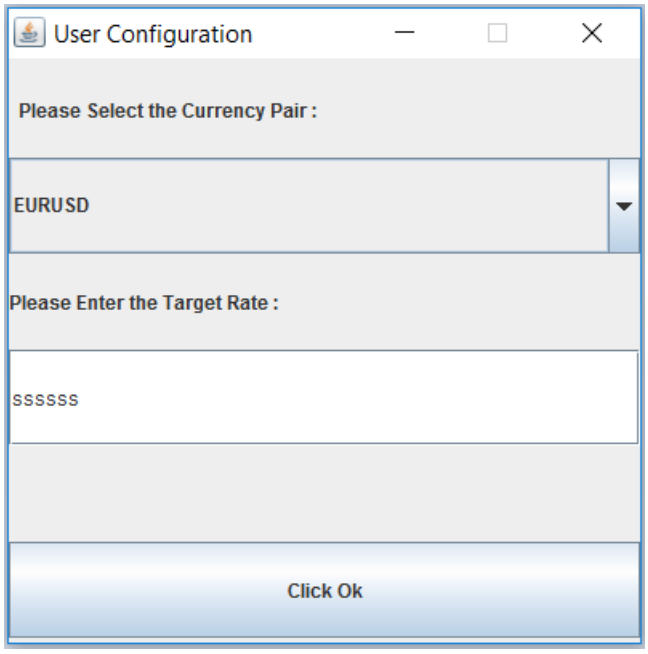

**Result Prompted to User:**



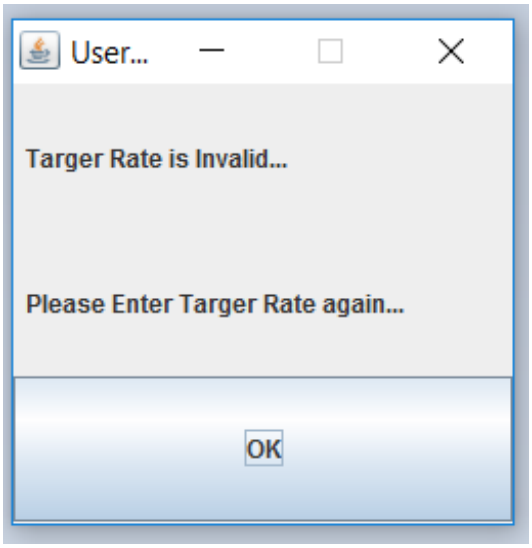
**Empty Value:**



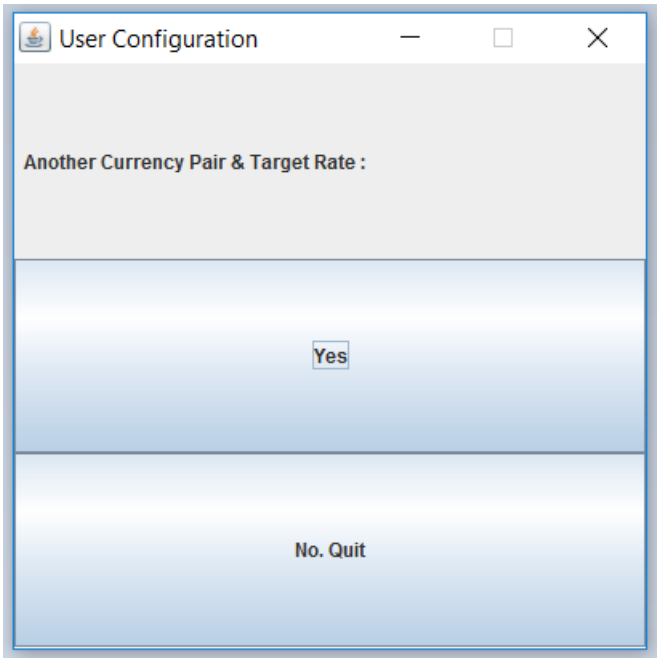

**Incorrect Value:**



**Prompt Message to User for Incorrect Value:**




**Prompt the User to Enter another Currency Pair & Target Value:**



# 6. REFERENCES

https://www.toptal.com/freelance/why-design-documents-matter

https://en.wikipedia.org/wiki/Software_design_description

http://www2.cs.uh.edu/~rsingh/

https://sovannarith.files.wordpress.com/2012/07/sdd_template.pdf