Currency Converter App Project Analysis

Revision 6

CMSC 495 6380

June 8, 2021

Group 4

Roy Auh, Ronald DeSears, Stephen Snelling

## **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Date** | **Description** | **Name** |
| 1 | 6/6 | Creation of Google Doc, added Introduction, Context Diagram, Risk and Mitigation, and Enhancement | Roy |
| 2 | 6/7 | Subsystem Diagram and minor formatting | Stephen |
| 3 | 6/8 | Subsystem-Requirements Mapping update with subsystems and a grammar pass | Stephen |
| 4 | 6/8 | Subsystem Diagram and Subsystem-Requirements Mapping Changes | Stephen, Ron |
| 5 | 6/8 | Added mathematical formula for currency conversion for subsystem requirement 5 | Ron |
| 6 | 7/5 | Updated to reflect design changes and feedback | Stephen |

## 

## **I. Introduction and Specifications**

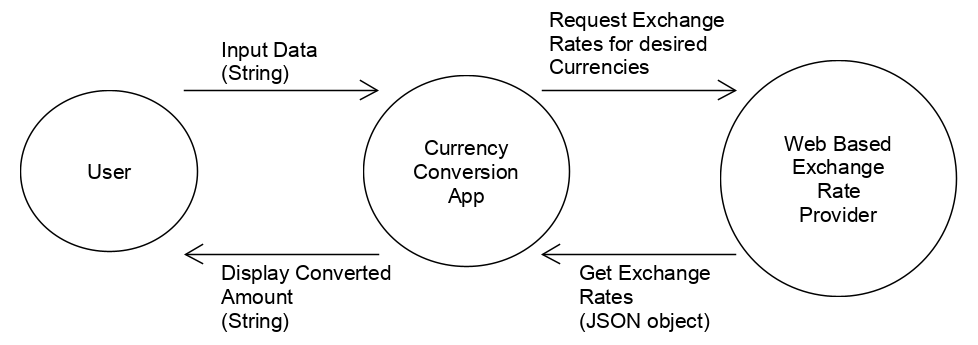
This project aims to create a simple GUI-based application that takes in user input and displays an output. The user input will be the amount of money to convert, the currency the input amount is in, and the currency the output amount will be in. After usage, the output will be the accurately converted amount. This app will source the most up-to-date currency conversion rates from https://www.exchangerate-api.com/docs/java-currency-api

**Outside System**: User, Web Based Exchange Rate Provider

**Input Data**: (1) Amount of money the user wants to convert, (2) the currency the input amount is in, (3) the currency that the user wants the output to be in.

**Output Data**: The amount of money converted from input data 1 into the currency the user specified the output to be in.

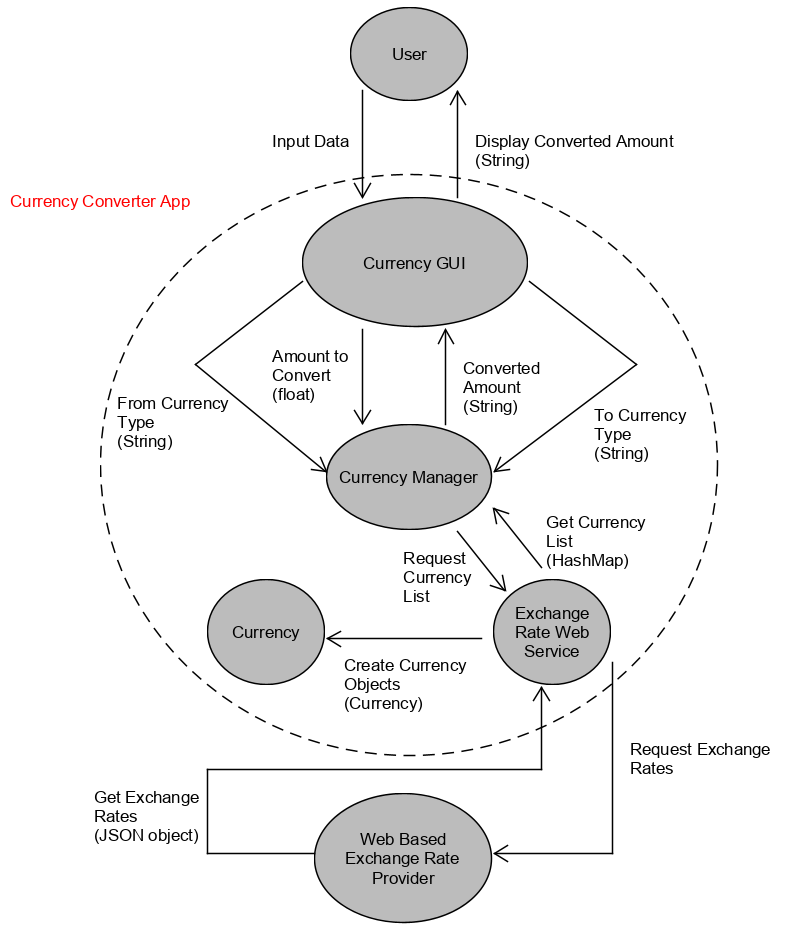
**Data Processing**: Take the input data from the GUI, use the web service to calculate the appropriate conversion, produce the converted amount, and then send that converted amount to the GUI for display.



*Fig. 1: Context Diagram*

This app will take the input data (amount to be converted, pre-conversion currency type, post-conversion currency type), request and receive from the web service the applicable exchange rate, perform the conversion, and finally display it to the user.

## **II. Subsystems**



*Fig. 2: Subsystems Diagram*

**Description of Subsystems**

1. *Currency GUI:* This subsystem represents the GUI of the app, receiving currency type to convert from, the amount to be converted, and the currency type to convert to from the user. It also receives the converted amount from the Currency Manager subsystem.
2. *Currency Manager:* This subsystem is the class that handles data manipulation for the app by receiving the amount to convert, the from-currency type, and the to-currency type from the Currency GUI subsystem when the convert button action listener activates. It also requests and receives data from the Web Based Exchange Rate Provider in order to get the current exchange ratios when the app starts. Using this data, it sends the converted amount back to the Currency GUI subsystem via a textfield.

## **III. Subsystem-Requirements Mapping**

Below is the table that maps the requirements to the relevant subsystems:

|  |  |  |
| --- | --- | --- |
| **Requirement Number** | **Description** | **Subsystem** |
| 1 | This app will be able to get daily updates of the currency conversion rates from https://www.exchangerate-api.com/docs/java-currency-apifor its conversion | Currency Manager |
| 2 | This app will allow the user to input the amount of currency to convert. | Currency GUI |
| 3 | This app will allow the user to toggle what currency to convert to and from. | Currency GUI |
| 4 | This app will throw an error if the user tries to enter negative numbers, or non-numbers. | Currency GUI |
| 5 | This app will accurately convert the given amount to the requested currency and display the result in the GUI.  Formula:  Input-Amount \* Conversion-Ratio = Output-Amount | Currency Manager, Currency GUI |
| 6 | This app will use the top 11 currencies.   1. US Dollar 2. Euro 3. British Pound 4. Indian Rupee 5. Australian Dollar 6. Canadian Dollar 7. Singapore Dollar 8. Swiss Franc 9. Malaysian Ringgit 10. Japanese Yen 11. Chinese Yuan Renminbi | Currency Manager  Currency GUI |
| 7 | The app’s GUI will feature at least:   1. a textfield where the user can enter their desired amount to convert. 2. an uneditable textfield that displays the converted amount. 3. a toggle menu for the input currency amount. 4. a toggle menu for the output currency amount. 5. a “Convert” button that, when pressed by the user, converts and displays the converted amount. | Currency GUI |

**Possible Enhancements**

* *Year-to-year inflation rate conversion*: An enhancement can be adding a feature where the user can also input the year in which the input amount and the output amount apply. This conversion would be inflation conversion.
* *Customizing the toggle menu*: Another enhancement would be adding a feature that allows the user to organize the toggle menu listing the currencies types by either alphabetical order or most commonly used order.
* *More currency choices:* This enhancement would add more than the original 11 currency types for the user to choose from.

**Risks and Risk Mitigation**

* *Hardware or system failure*: If the hardware or the host operating system experiences total failure or shutdown, then the app will not be operable.
  + *Possible risk mitigation*: the program can ensure that it is not the direct cause of any system failures by preventing common system failure-causing issues, such as deadlocks.
* *Web service access failure*: The hardware or the system does not have internet access, or the online currency provider has crashed on their end.
  + *Possible risk mitigation*: the Currency Manager subsystem can retain the last accessed conversion rate or have rates we program in as backup.