**Group 2: Project Analysis**

**Current Document Revision: 5**

**Currency Exchange Program ( Exchange! )**

**Reginald Carey, Jason Dudash, Robert Kibler, Brandon Trexler**

**University of Maryland: University College**

**CMSC 495 (7980)**

**05 May, 2016**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 1 APR 16 | Revision 1.0 | Robert Kibler | Initial document |
| 2 APR 16 | Revision 2.0 | Reginald Carey | Formatting and content updates. |
| 3 APR 16 | Revision 3.0 | Jason Dudash | Add requirement to section 3.2. Note API as sub component. Paragraph in 3.1.1 removed. Removed Android from terms. |
| 9 APR 16 | Revision 4.0 | Jason Dudash | Alphabetize section 1.3 |
| 5 MAY 16 | Revision 5.0 | Robert Kibler | Added Risks and Enhancements Section |
| 5 May 16 | Final Revision | Robert Kibler | Finalized formatting |

# Document Approval

The following Software Design Document has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

2.0 System Overview 1

2.1 Sub-System Component Decomposition 2

2.1.1 Exchange! GUI 2

2.1.2 Exchange! Exchange Rate DB 2

2.1.3 Exchange! Exchange Rate Web Service 2

2.1.4 Exchange! Currency Conversion Logic 3

Procedural Design 4

3.1 Use Case Breakdown 4

3.1.1 Exchange Rate Only Use Case 4

3.1.2 Conversion Use Case 5

3.2 Requirement Analysis Tracing 7

4.1 External Interface 8

4.2 Risks and Future Enhancements 8

# Introduction

## Purpose

This Project Analysis Document will provide the notional data flow and subsystems analysis for a currency exchange program.

## Scope

This document is to provide an overview only and will not cover any implementation design details.

## Definitions, Acronyms, and Abbreviations

* **DB –**database. This refers to a simplified storage solution that reduces storage space and simplifies search and retrieval of data.
* **GUI** – graphical user interface. This is a control mechanism that provides a user with program control using a graphical instead of text based interface
* **Windows –** A desktop operation system that supports many types of applications and web interface

## References

* IEEE Std 1016‐1998: IEEE Recommended Practice for Software Design Descriptions

# System Overview

The currency exchange program (Exchange!) is a minimalist graphical interface currency exchange program that will allow a user to calculate the value of any amount of currency given a base currency and target currency to exchange to.

The program will use the most recent possible exchange rates retrieved from the Internet. If the exchange rate service is not available, a locally stored DB of rates updated previously will be used instead.

The overall system architecture shown in Figure 2.1 depicts the high-level interactions between user, software, and remote service.

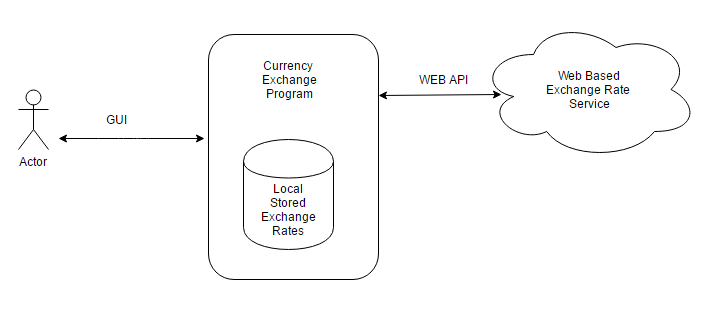


Figure 2.1 Architecture Design

# Sub-System Component Decomposition

## Exchange! GUI

Exchange! uses a minimalist interface to present the user with several options. The user will be able to select the currencies for exchange and enter a currency amount to convert. The purpose of this program is to provide an extremely simplified currency exchange experience on the fly. The GUI is a subcomponent of the Exchange! program and run on the same hardware as the rest of the software.

## Exchange! Exchange Rate DB

Exchange! will store all available exchange rates in a local DB to simplify retrieval and provide for exchange rates if the online service is not available. This DB will be updated when the program starts and when any new exchange rate is detected during use. The DB will be utilized as an exchange rate service if the internet service is unavailable.

## Exchange! Exchange Rate Web Service

Exchange! will retrieve currency exchange rates from finance.yahoo.com services. The exchange rates will be used to update the DB upon system startup. The exchange rate service shall be queried for exchange rates when the user makes a currency conversion request. The new rates retrieved shall be loaded into the DB. The exchange rate service shall provide a list of valid currency codes which will be utilized to populate the user interface. If the internet service is unavailable, the DB shall be utilized as a fallback.

## Exchange! Currency Conversion Logic

Exchange! will use a separate subsystem to calculate the total value currency when requested by the GUI. The conversion subsystem will receive data from the GUI, Web Service and the DB subsystems and serve the data back to the GUI and the database.

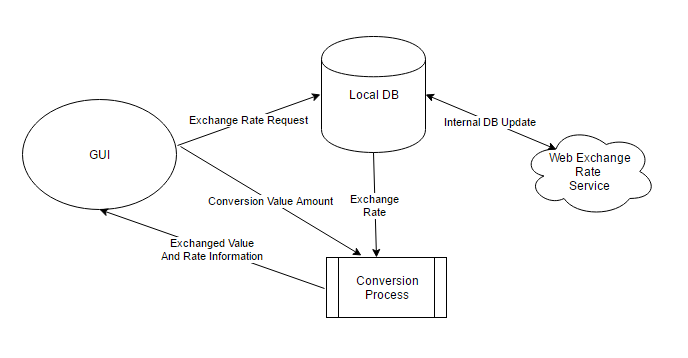


Figure 2: Sub System Data Flow

# Procedural Design

# Use Case Breakdown

The Exchange! system has two use cases. See figure 3.1

* + Request Exchange Rate
  + Convert Currency

All other use cases are simply modifications of these basic use case designs where the user is inserted into the sequence at a different step. These use cases will be discussed below.

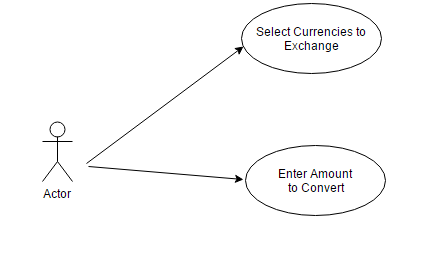
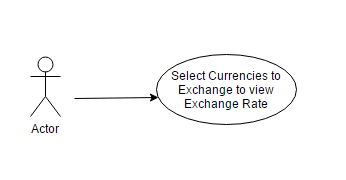


Figure 3.1 Use Cases

## Exchange Rate Only Use Case

A user may select currencies to exchange for the sole purpose of seeing what the exchange rate currently is.

* + In this case the program will send a request to the DB for exchange rates
    - The DB will attempt to query the online service the acquire the most up to date exchange rates. If the service is unavailable, the stored rate will be returned.
  + The exchange rate and validity date of the rate will be displayed on the GUI

Figure 3.2 represents the sequence of events in the exchange rate only case.

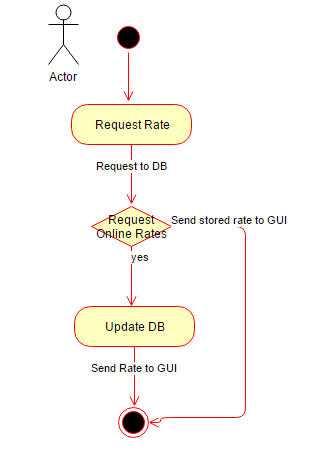


Figure 3.2 Request Rate Sequence

## Conversion Use Case

The user may request a conversion of an amount of currency. This use case will follow a similar path to the previous except the amount will be sent to the conversion subsystem and a converted amount will be returned to the GUI.

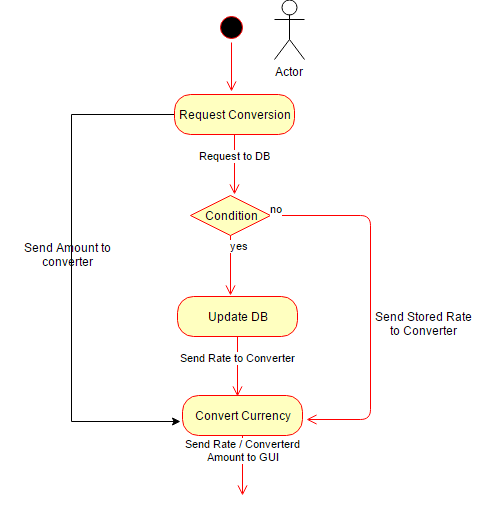


Figure 3.3 Conversion Event Sequence

# Requirement Analysis Tracing

To verify that the Requirements are met we show which subsystems combine to satisfy each of the requirements

|  |  |
| --- | --- |
| **REQUIREMENT** | **Subcomponent** |
| The system shall support a desktop based GUI interface | GUI |
| The system shall provide a converted currency output | GUI, API, DB, Converter |
| The system shall provide a mechanism to allow a user to provide a currency value to be converted | GUI – Input Field |
| The system shall provide a mechanism to allow a user to select a currency type from which to convert. | GUI – Combo Box |
| The system shall provide a mechanism to allow a user to select a currency type to which to convert. | GUI – Combo Box |
| The system shall utilize the most recent currency valuations available from a single source. | DB - Update |
| The system shall allow currency inputs to be either in whole or partial denominations. | GUI – Input Field |
| The system shall present converted currency outputs with an accuracy of two decimal points using normal rounding. | GUI – Output Field |
| The system shall present as part of the currency conversion output, the source and age of the currency valuation data used in the computation. | GUI – Output Field  DB – Rate / Timestamp |
| The system shall present as part of the currency conversion output, the currency conversion rate used in the computation as well as the reverse conversion rate. | GUI – Output Field  DB – Rate / Timestamp |
| The system shall have a mechanism to retrieve a list of “from” and “to” currency codes as a step prior to presenting a user interface. | DB - Update |
| The system shall have a mechanism for ensuring that the list of valid currency codes is less than 24 hours old. | DB - Update |
| The system shall conduct a currency conversion as soon as a new currency type or currency value is input | GUI – ActionEvent  Converter – Pass Data |
| The system shall use the United States Dollar as a pivot currency. | Converter - Logic |

## External Interface

The Exchange! system must communicate with the external exchange rate service via web access. The program will utilize the Operating System provided networks to access the internet. The program will not care if the network is wireless or wired as long as world wide web access is available. The program will use https protocol to download a JSON object from finance.yahoo.com services.

* + 1. **Interface:** finance.yahoo.com/webservice
    2. **Protocol:** https / http (if https is not available)

## Risks and Future Enhancements

* + 1. **Risks** 
       1. Hardware or Host OS Failure – Exchange cannot predict or control a hard shutdown of the host hardware. Exchange could stop working do to a hardware or host OS failure.
       2. Extended Offline Operations – Exchange is designed to have at least periodic connection to the internet for currency exchange rate updates. If the program is used in a network denied environment for an extended period of time, the exchange ranges presented will be out of date.
       3. Initial use without internet access – The application will only function to any degree if it is able to populate its database from the internet. Thus the initial run of the application must involve internet access.
       4. Database files may be corrupted on startup – all data can be purged and reloaded from the internet connection.
    2. **Enhancements**
       1. Providing the user with a favorites section to stored commonly used currencies
       2. Provide historic exchange rate data and graphical presentation of trends.