**Currency Converter**

**University of Maryland Global Campus**

**CMSC 495 7981 Current Trends and Projects**

Group 4: Justin Miller, Mike Yacht, Ayao Adanto, Brandon Tennyson

Project Design

Revision 7

April 21, 2020

# 

**Revision Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Description | Revisor | Date |
| 1 | Initial Document | Michael Yacht | April 6, 2020 |
| 2 | Group conference on document and revisions | Entire group through online meeting | April 7, 2020 |
| 3 | Added PseudoCode for GUI | Brandon Tennyson | April 8, 2020 |
| 4 | Cleaned up some layout | Michael Yacht | April 9, 2020 |
| 5 | Clean up some grammar and formatting | Justin Miller | April 9, 2020 |
| 6 | Fixed minor grammar errors | Brandon Tennyson | April 9, 2020 |
| 7 | Updated Pseudocode  After Sprint 1 | Michael Yacht | April 21, 2020 |
| 8 | Updated Pseudocode | Michael Yacht | April 28, 2020 |

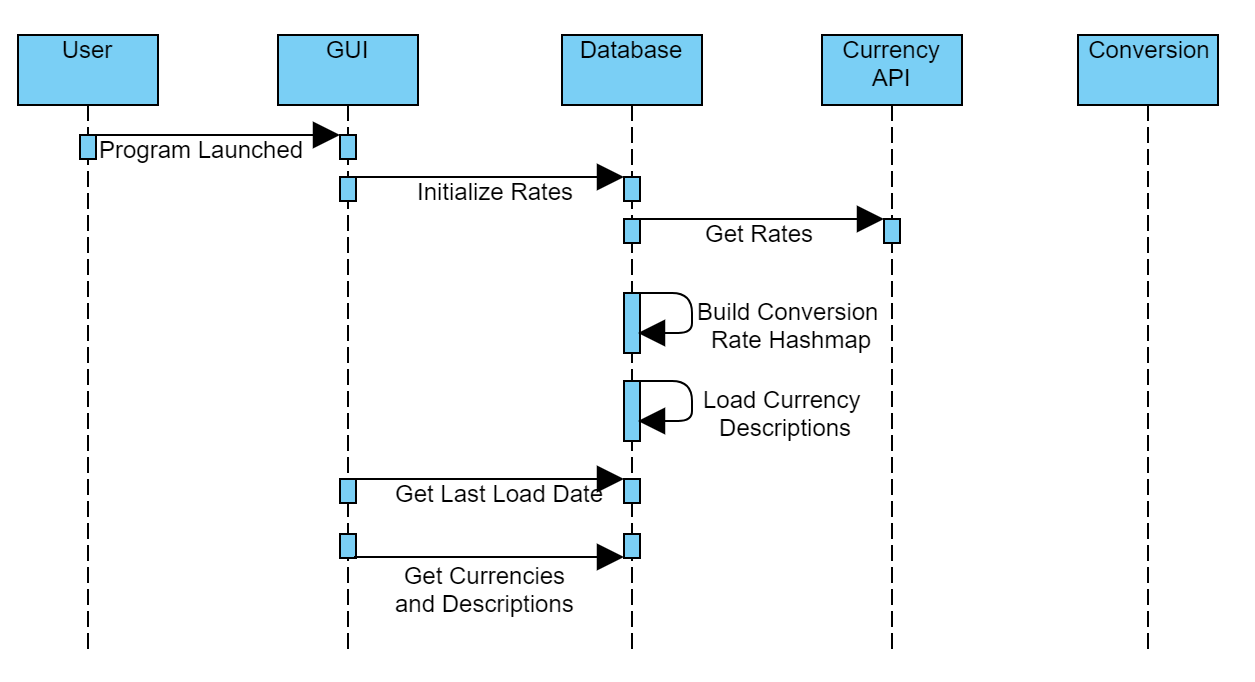
# Event Trace Diagrams

**Scenario 1: Startup**

Description: The program is loaded and all subsystems initialize

Precondition: Program is not yet loaded

Post-condition: Program is ready to convert currencies and accept user input

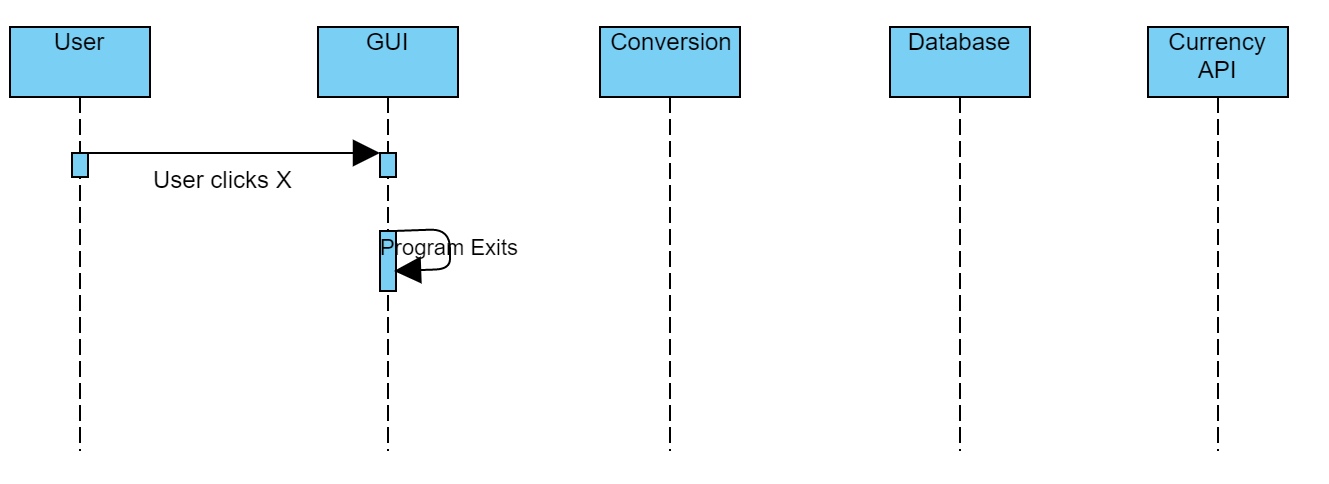


**Scenario 2: Shut-down**

Description: The program is exited

Precondition: The user clicks the X button in the top right

Post-condition: The program is no longer running

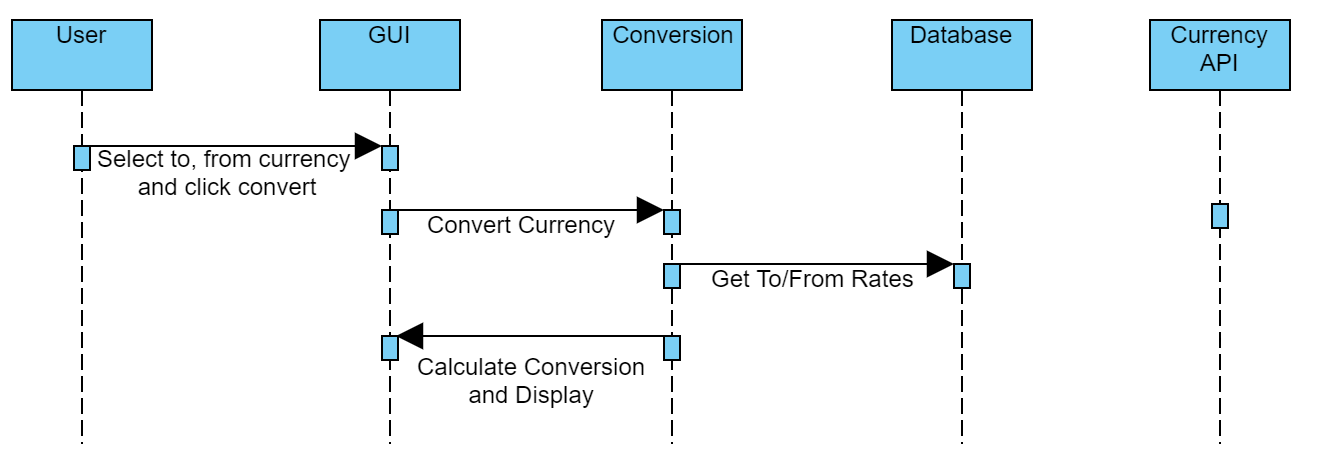


**Scenario 3: Normal Operation**

Description: The user selects a “to” and “from” currency, enters the amount to convert and clicks convert button

Precondition: The program is running and all rates and descriptions are loaded.

Post-condition: Currency is converted and displayed, program is ready for more conversions

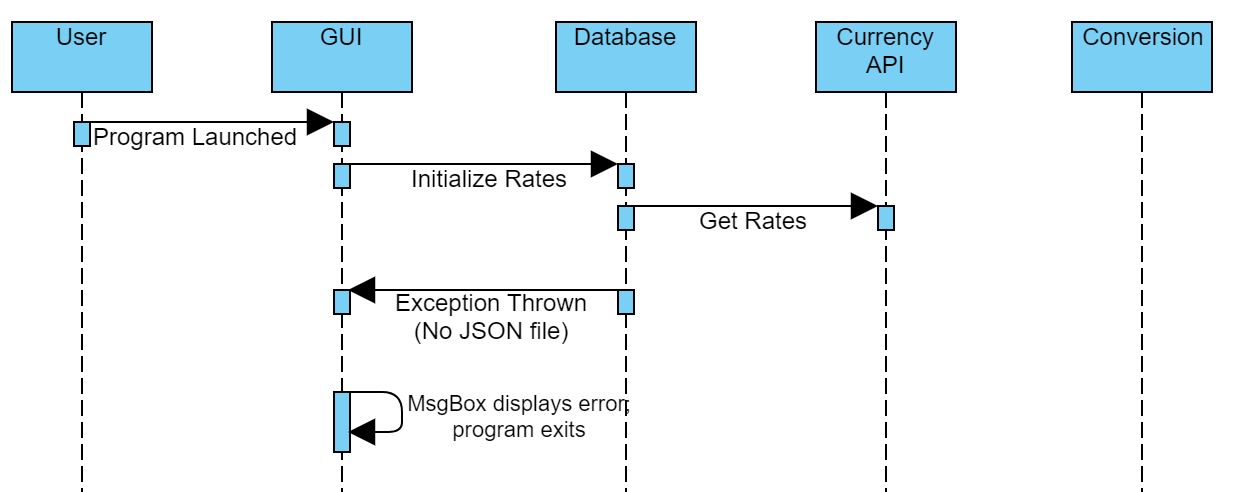


**Scenario 4: No rates loaded**

Description: Both the API and the file-load process to load rates fails. Exception is thrown, the program must exit since there’s no good data.

Pre-condition: This would happen at start-up.

Post-condition: Program exits

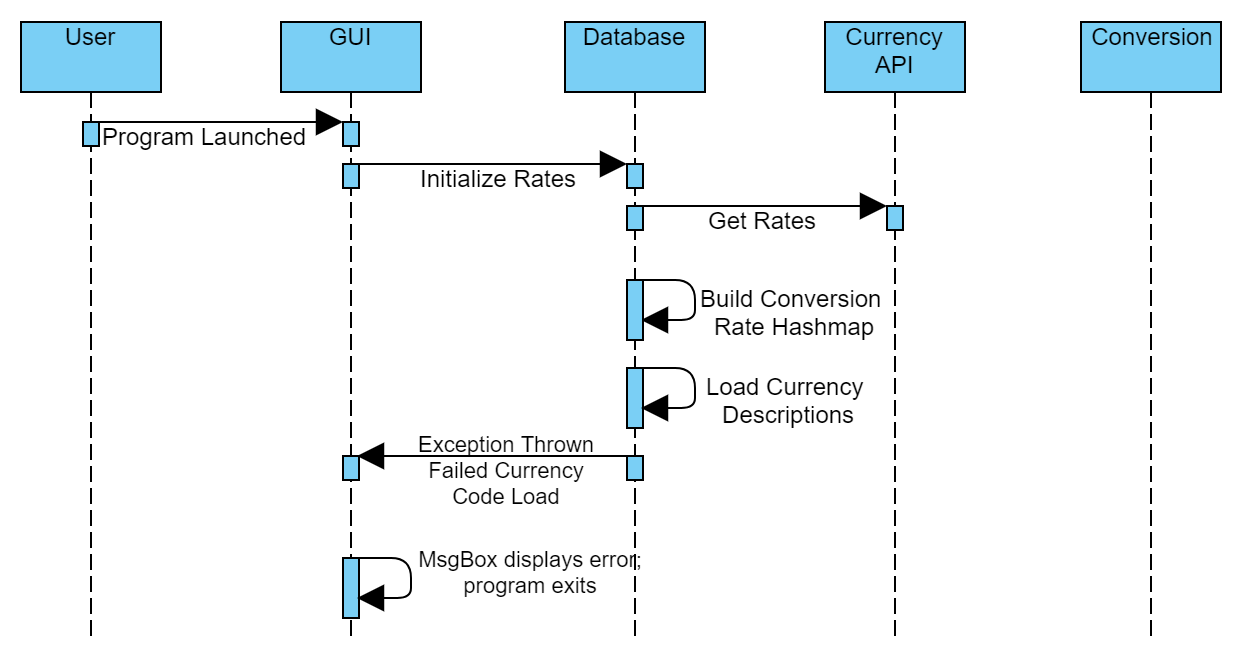


**Scenario 5: No code descriptions loaded**

Description: Rates load but the mapping of code to code descriptions fails. An exception is thrown, the program must exit since there’s no good data.

Pre-condition: This would happen at start-up.

Post-condition: Program exits



# **PseudoCode**

**GUI subsystem:**

**Class currencyGUI{**

JFrame frame;

JLabel output; //Finished conversion

JLabel date; //Last load date

JTextField originalValue; //user input value

JButton convertButton; //the convert button

JComboBox originalCurrency; //value to convert

JComboBox convertedCurrency; //value to convert to

void addElements(){

1. Add the gui elements to the frame and display

}

void actionPerformed(){

1. Checks value of each combo box
2. Checks textbox value
   1. If value is not numerical, throw exception
3. Calls conversion module
4. Displays converted value with description
   1. If the description fails to load, throw exception

}

void getVersion(){

1. Gets latest version

}

void initializeRates(){

1. initializes rates for conversion
   1. If it fails to load a JSON file, throw an exception

}

}

**Database subsystem:**

**Class currencyDB {**

long lastUpdatedDate;

HashMap rates; // Code:Value

HashMap codes; // Code:Description

String JSONRates;

String JSONPath;

String OPENEXCHANGERATEURL;

void currencyDB() {

1. Initialize all variables
2. Call update rates

}

void updateRates {

updateRatesFromAPI();

If it fails: updateRatesFromFile();

parseJSON();

loadCurrencyCodesFromFile();

cleanupCurrencyCodes();

}

void updateRatesFromAPI() {

1. Call the API
2. Store the returned JSON in the JSONRates variable
3. Call saveJSONToFile()

}

void updateRatesFromFile() {

1. Check if the file exists (latest.JSON)
   1. If not, toss an exception
2. Load the JSON directly into the JSONRates variable

}

void saveJSONToFile() {

1. Save full JSON to latest.JSON file (Streamwriter)
   1. If Fails, toss exception

}

void parseJSON(String JSON) {

1. Load timestamp field in JSON to lastUpdatedDate
2. Clear rates; // init to no rates
3. Loop through rates field in JSON (until no more rates exist)
   1. Load each “currency: rate” pair into rates

}

void loadCurrencyCodesFromFile() {

1. Check if the file exists (currency.csv)
   1. If not, toss an exception
2. Loop through each line of the csv
   1. Load each pair of currency:currency description into codes HashMap

}

void cleanupCurrencyCodes() {

// This is where we remove codes from hashmaps that don’t exist in BOTH hashmaps

1. Loop through all keys in rates HashMap, ensure they each exist in codes HashMap
   1. Delete any keys from rates that exist in rates but not codes.
2. Loop through all keys in codes HashMap, ensure they each exist in rates HashMap
   1. Delete any keys from codes that exist in codes but not rates.

}

HashMap getCurrencyCodesHash() {

Returns codes HashMap;

}

double getConversionRate(String currency) {

1. If the currency does not exist in rates, toss exception
2. Return the rate stored in rates HashMap

}

String[] getAllCurrencyCodes() {

1. Create String array = # elements in rates HashMap
2. Dump all keys from rates HashMap into String array
3. Return string array

}

String getCurrencyCodeDescription(String currencyCode) {

1. Check if currencyCode exists in keys for codes HashMap
   1. If not, toss exception
2. Return value from codes HashMap for key = currencyCode

}

String getCurrencyCodeFromDescription(String currencyDesc) {

1. Check if currencyDesc exists in values for codes HashMap
   1. If not, return null
2. Return key from codes HashMap for value = currencyDesc

}

long getLastUpdatedDate() {

return lastUpdatedDate;

}

}

**Conversion subsystem:**

Class conversionModule {

Double convertCurrency(String fromCode, String toCode) {

1. Get the conversion rate for fromCode: Database.getConversionRate(fromCode);
2. Get the conversion rate for toCode: Database.getConversionRate(toCode);
3. ConversionRate = ToRate / FromRate
4. Return ConversionRate

}

}

# **Unresolved Risk and Risk Mitigation**

We believe we have addressed all the risks, and are unaware of any unresolved risks.

Risk 1: Cannot connect to currency API or API fails

Mitigation: Load latest rates from latest.JSON stored file

Risk 2: Cannot connect to currency API and cannot load latest.JSON stored file

Mitigation: Exception thrown, displays error, program exits.

Risk 3: Cannot save latest JSON from API to latest.JSON file

Mitigation: Exception thrown (and handled), display warning, program continues.

Risk 4: Non-numeric data entered into GUI for conversion

Mitigation: Prompt user with MsgBox, clear out bad data

Risk 5: Currency Code exists in only 1 source (either the conversion data or the description map)

Mitigation: All currency codes that do not exist in BOTH HashMaps are removed from HashMap (cleanupCurrencyCodes method).