Project Title: Currency Converter

Team Members: 1. Sai Kartheesha Muthyala
2. Harshitha Gudapati

Table of Contents

Introduction

The Currency Converter project aims to develop a Java-based application that allows users to convert currencies from one unit to another. The application will provide an intuitive interface for users to input the amount and currency they want to convert, and display the converted amount in the desired currency.

The project will benefit individuals and businesses that need to perform currency conversions, such as travelers, importers, and exporters. The application will use up-to-date exchange rates to ensure accurate conversions.

Glossary

- Currency: a system of money in general use in a particular country
- Exchange Rate: the rate at which one currency can be exchanged for another
- Conversion: the process of changing one currency to another

User Requirements

1. The system shall allow users to input the amount and currency they want to convert.
2. The system shall display the converted amount in the desired currency.
3. The system shall provide an option to select the source and target currencies.
4. The system shall update exchange rates in real-time.
5. The system shall provide a history of previous conversions.
6. The system shall allow users to save their favorite conversions.
7. The system shall provide a user-friendly interface.

8. The system shall retrieve the exchange rate from a trusted API.
9. The system shall provide error handling for invalid input.
10. The system shall store recent conversions locally.
11. The system shall allow users to switch between different languages.
12. The system shall be user-friendly with a simple interface.

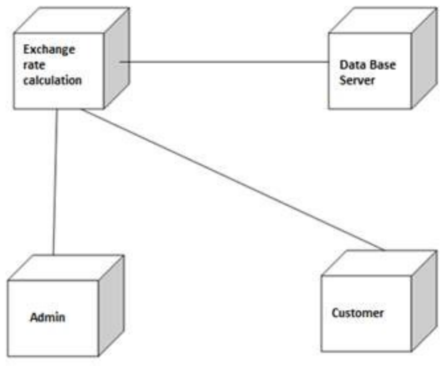
System Requirements

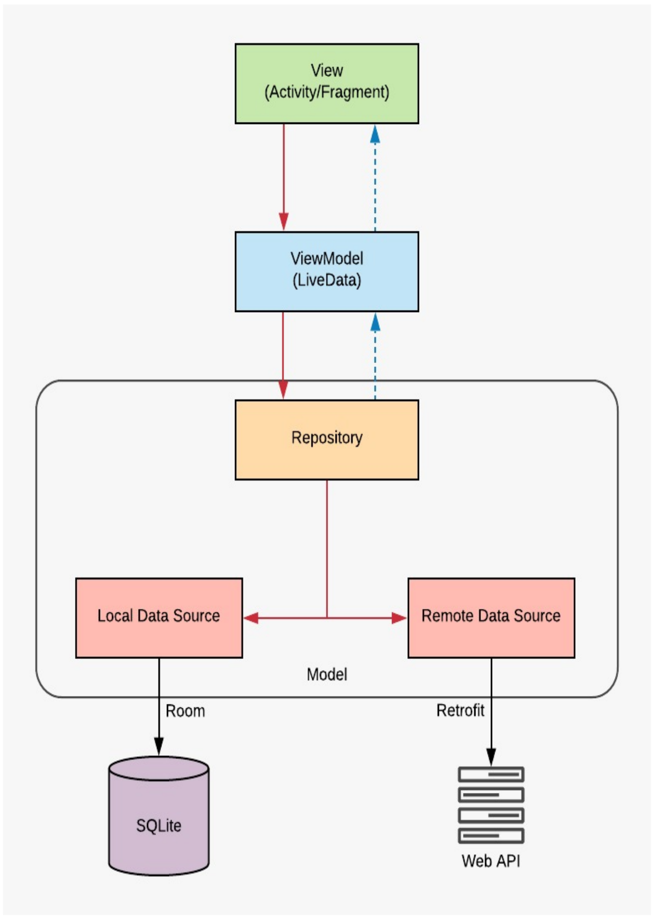
1. The system shall be developed using Java programming language.
2. The system shall use a relational database to store exchange rates and user data.
3. The system shall use a web-based interface to interact with users.
4. The system shall be able to handle a minimum of 100 concurrent users.

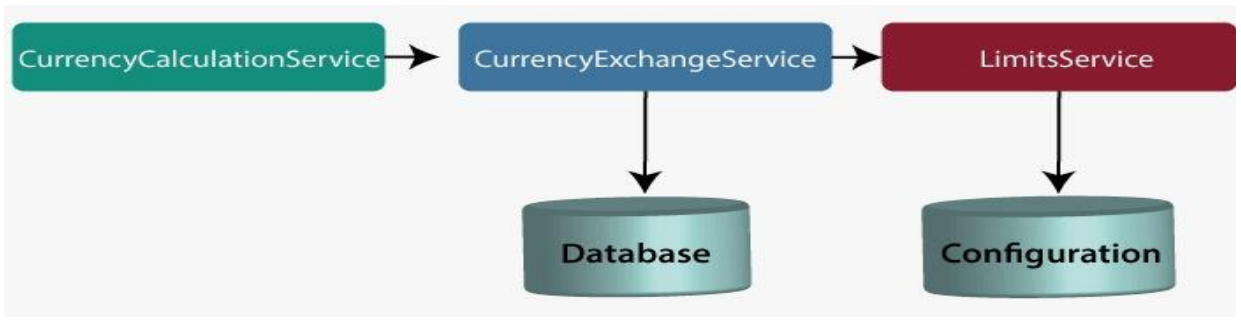
System Architecture

The system architecture will consist of three key components:

1. **User Interface**: A simple interface that allows users to input data and view results.
2. **API Integration**: An API client that fetches exchange rates from the external API.
3. **Local Storage**: A system to store recent conversions and handle offline data.



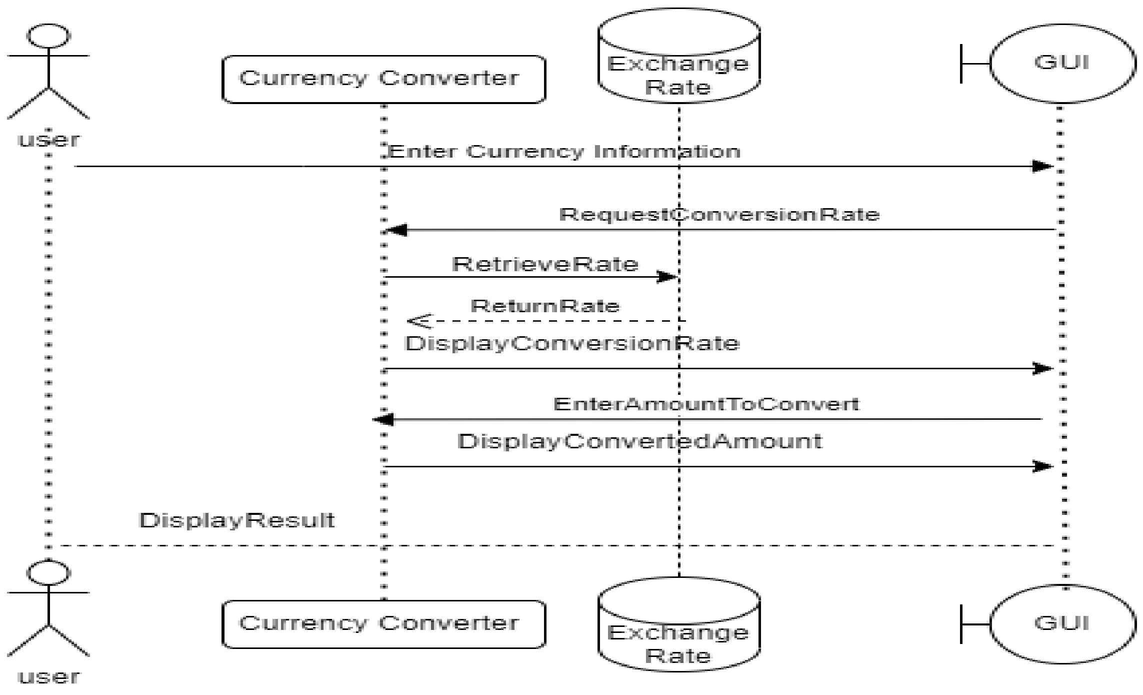




System Models:

1. <u>**Sequence Diagram**</u>

- **User Input (Enter Currency Information)**:
- The user initiates the process by entering the currency information into the **Currency Converter** through the GUI. This includes selecting the base currency, target currency, and entering the amount they want to convert.
- **Request Conversion Rate**:
- The **Currency Converter** system sends a request to the Exchange Rate API to retrieve the current exchange rate between the selected base and target currencies.
- **Retrieve Rate (API Interaction)**:
- The **Exchange Rate API** processes the request and retrieves the current exchange rate for the specified currencies. It then sends the rate back to the **Currency Converter** system.
- **Display Conversion Rate**:
- Once the **Currency Converter** system receives the rate from the API, it calculates the conversion based on the entered amount and exchange rate. The converted amount is then displayed back on the GUI for the user to view.
- **Display Converted Amount**:
- After the conversion is completed, the Currency Converter system displays the final result (converted amount) on the GUI interface. The user can see the amount in the target currency.
- **Display Result**:
- The user can view the converted amount through the GUI once the system has performed the necessary calculations and returned the result.



<u>2.Class Diagram</u>

1. CurrencyConverter Class:

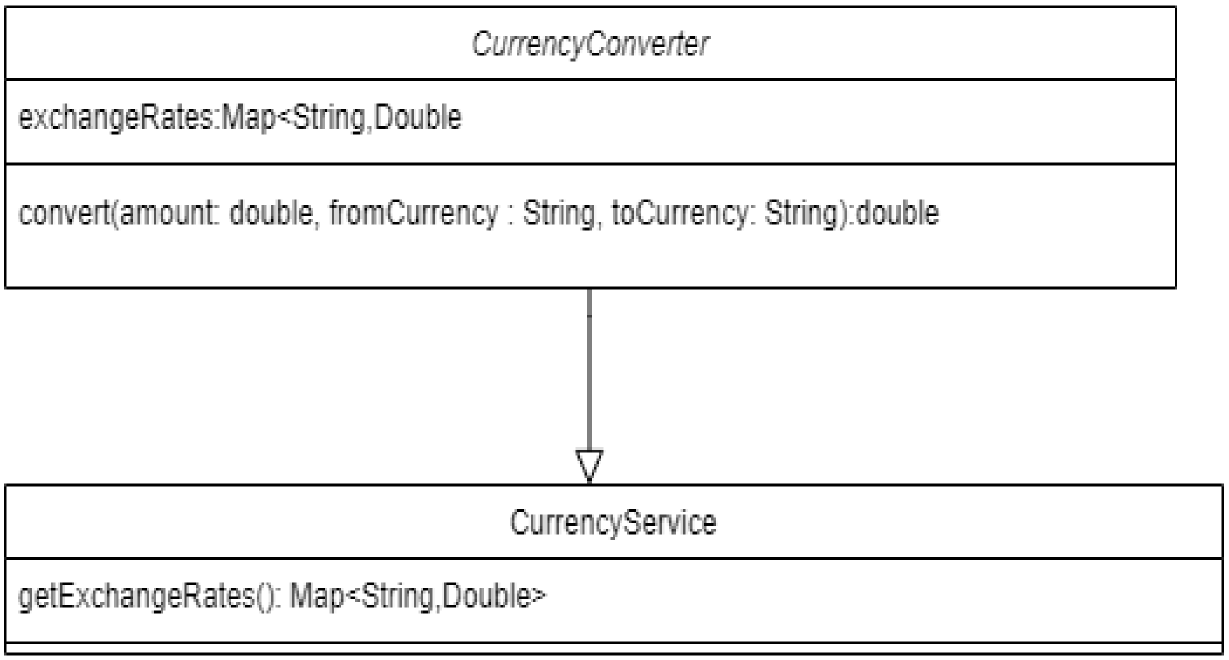
- **Attributes**:
  - `exchangeRates`: A map that stores exchange rates where the key is a `String` representing a currency pair (e.g., "USD/EUR") and the value is a `Double` representing the exchange rate.
- **Methods**:
  - `convert (amount: double, fromCurrency: String, toCurrency: String): double`: This method takes in an amount and two currency strings (from and to). It uses the exchange rate data to convert the amount from one currency to another, returning the converted amount as a `double`.

2. CurrencyService Class:

- **Methods**:
  - `getExchangeRates(): Map<String, Double>`: This method retrieves a map of exchange rates (currency pairs as keys and rates as values). It provides the data necessary for conversion in the `CurrencyConverter` class.

Interaction Between the Classes:

- **CurrencyConverter** depends on Currency Service to get the exchange rates needed for conversion. In the diagram, the arrow from `Currency Converter` to `Currency Service` represents a dependency or association.
    - The `Currency Converter` uses the `getExchangeRates()` method from the `CurrencyService` class to populate its internal `exchangeRates` map.
    - Once it has the exchange rates, the `convert ()` method in `CurrencyConverter` can perform the conversion using the provided `fromCurrency` and `toCurrency` inputs.


CurrencyConverter
exchangeRates:Map<String,Double
convert(amount: double, fromCurrency : String, toCurrency: String):double
CurrencyService
getExchangeRates(): Map<String,Double>

3.Active Diagram

1. User Input:

- The process begins with the user initiating input. The user enters the amount they want to convert and selects the source currency (the currency they are converting from) and the target currency (the currency they want to convert to).

2. Enter Amount:

- The user provides the amount of money they wish to convert.

3. Select Source Currency:

- The user chooses the currency they currently hold (e.g., USD, EUR, etc.).

4. Select Target Currency:

- The user selects the currency they want to convert the entered amount into (e.g., JPY, GBP, etc.).

5. Validate Input:

- The system validates the user inputs, checking if the entered amount and selected currencies are valid (e.g., non-empty values, positive numbers, etc.).
- **If the input is invalid**:
  - The system displays an error message to the user, signaling that the entered data needs correction (e.g., negative amount, missing currency).
- **If the input is valid**:
  - The process continues to the next steps for currency conversion.

6. Convert Currency:

- Once the input is validated, the system initiates the currency conversion process.

7. Fetch Exchange Rates:

- The system fetches the required exchange rate for the currency pair (source and target) from the available data source, possibly a database or external API.

8. Calculate Converted Amount:

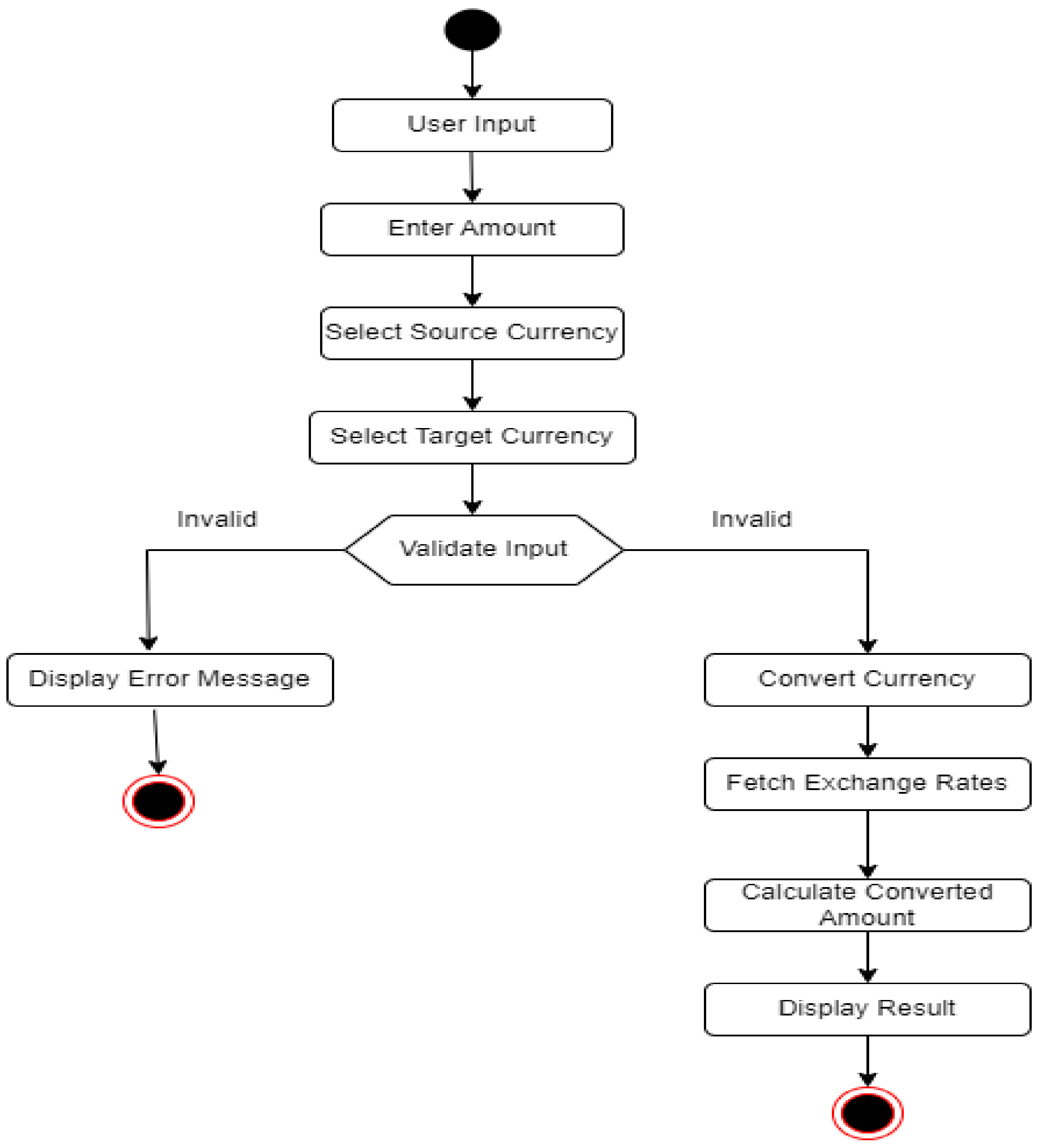
- Using the fetched exchange rate, the system calculates the converted amount. The formula typically involves multiplying the entered amount by the exchange rate.

9. Display Result:

- After calculating the conversion, the system displays the converted amount to the user, completing the process.

10. End:

- The process ends either after an error message is displayed (due to invalid input) or after the result is successfully shown to the user.


User Input
Enter Amount
Select Source Currency
Select Target Currency
Invalid
Validate Input
Invalid
Display Error Message
Convert Currency
Fetch Exchange Rates
Calculate Converted Amount
Display Result