Name – Yash Jahagirdar **Andrew id - ybj**

Name - Leo Lin **Andrew id - hungfanl**

Project 4 task 2 documentation-

Description:

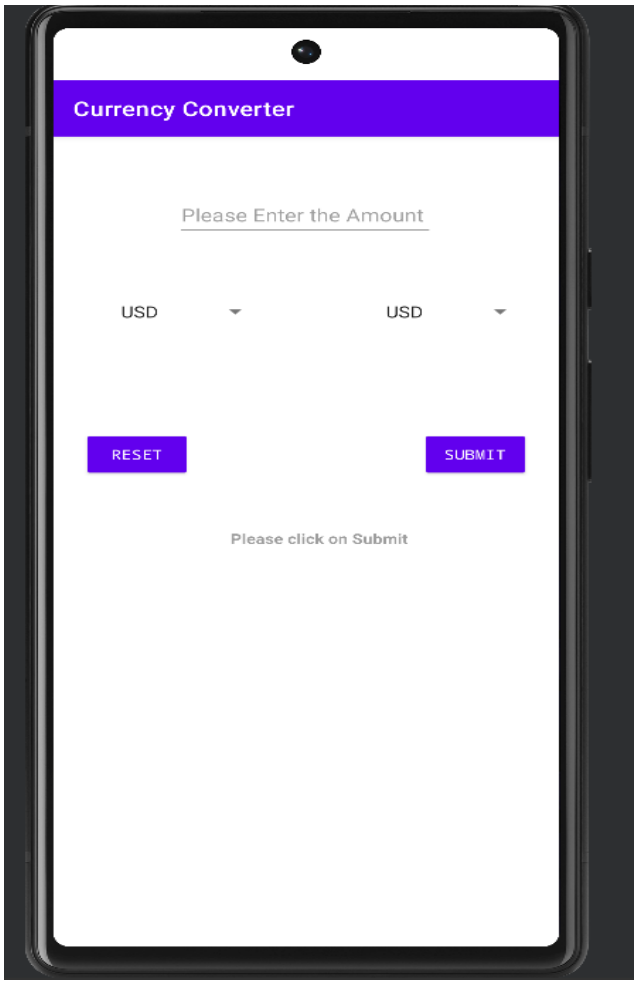
My mobile application is a currency converter application for certain currencies. My mobile application converts the amount entered by the user to the desired currency selected by the user.

1. Implement a native Android application-
   The name of my native Android Application project in Android Studio is Currency Converter.

a. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View)-

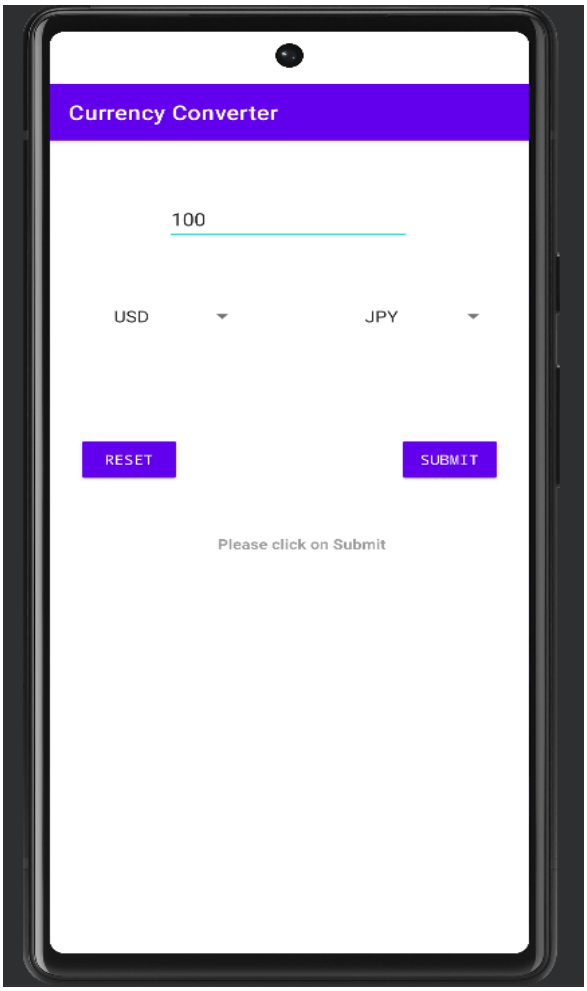
My application uses EditText, TextView, Spinner and Button. See content_main.xml for details on how they are incorporated for the ConstraintLayout.

Here is a screenshot of the layout before the amount to be converted is entered-



b. Requires input from the user

Here is a screenshot of the user entering the amount to be converted and selecting the currency from which the user wants to convert from and the currency to which the user wants the currency to be converted to-



c. Makes an HTTP request (using an appropriate HTTP method) to your web service-

My application does an HTTP GET request in MainActivity.java. The HTTP request is:

https://sleepy-sierra-12630.herokuapp.com/CurrencyExchange?currencyFrom=" + fromCurrency + "&currencyTo=" + toCurrency + "&amountString=" + params[0]

The fromCurrency is the string such as “USD”, which represents the currency from which the user wants to convert the currency from.

The toCurrency is the string such as “JPY”, which represents the currency to which the user wants to convert the currency to.

The params[0] is the amount such as “100” which needs to be converted.

On clicking on submit button, the setUpListeners method makes the request to my web application which is hosted in Heroku, parses the returned JSON to get the exchange amount and displays it in TextView .

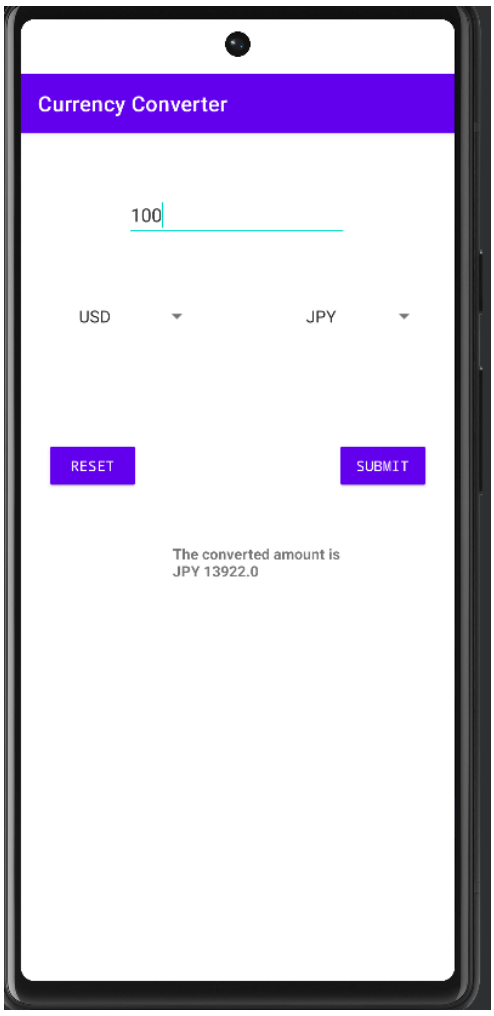
d. Receives and parses an XML or JSON formatted reply from your web service-

An example of the JSON reply is -

{"Amount":100,"ToCurrency":"JPY","Exchange Amount":13922,"Transaction Time":"Nov, 16, 2022","Base":"USD"}

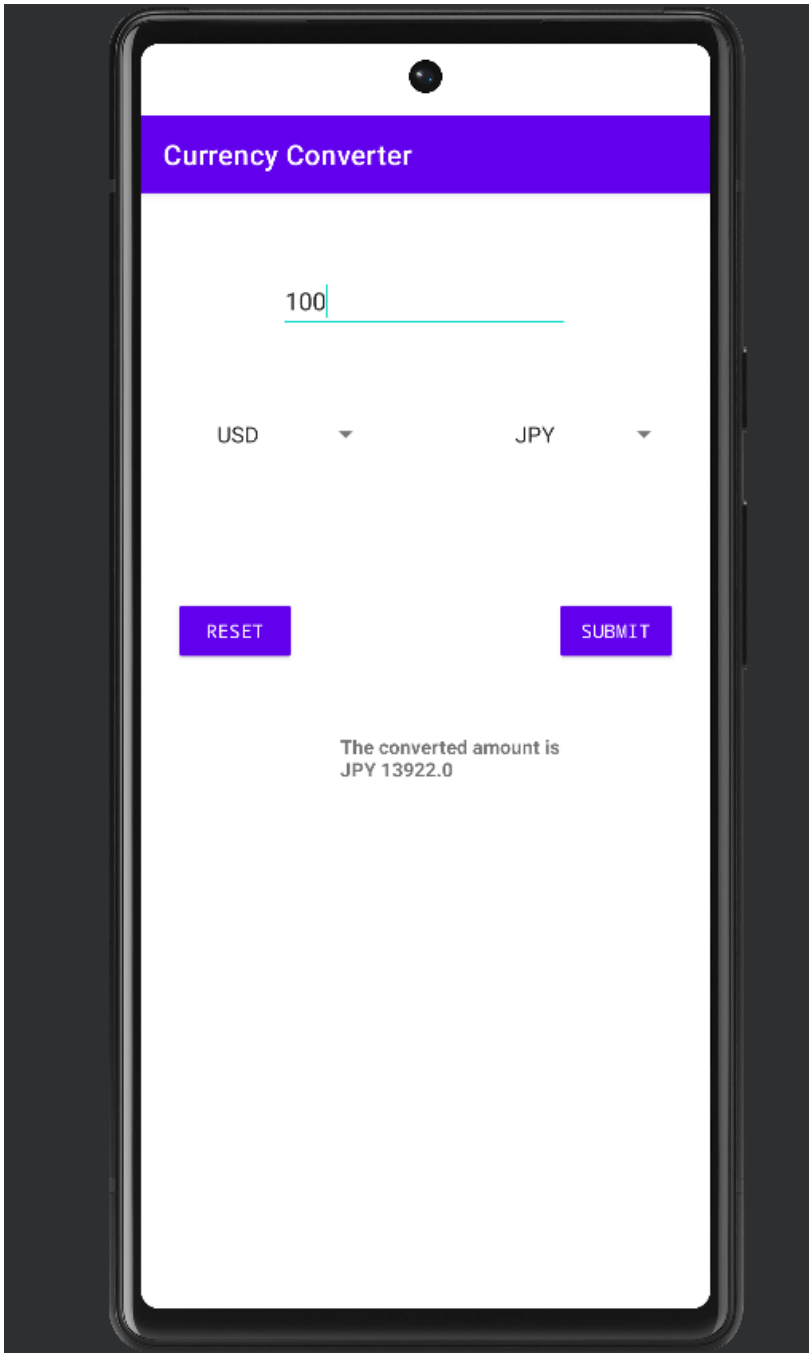
e. Displays new information to the user-

Here is the screenshot after the submit button is clicked by the user.



f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

The user can type another input amount and select the same fromCurrency and toCurrency options at the same time to get the same exchange amount(result.)



2. Implement a web service, deployed to Heroku

   The URL of my web service deployed to Heroku is –
   sleepy-sierra-12630
   The project directory name is Project4Task2

   a. Using an HttpServlet to implement a Simple (can be a single path API)

      In my web app project:
      Model: CurrencyExchangeModel.java
      View: result.jsp
      Controller: CurrencyExchangeServlet.java

   b. Receives an HTTP request from the native Android application

      CurrencyExchangeServlet.java receives an HTTP GET request with the argument "currencyFrom", "currencyTo", and "amountString".

c. Executes business logic appropriate to your application
CurrencyExchangeServlet.java makes an HTTP request to https://www.frankfurter.app/. It then store the record to MongoDB and create a GSON Object and send it back to the client.

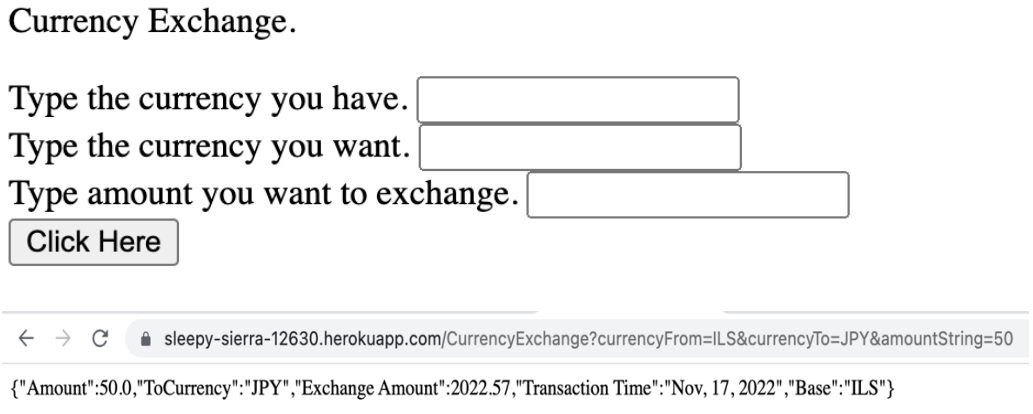
d. Replies to the Android application with an XML or JSON formatted response
The servlet directly response an JSON formatted response.
<%= request.getAttribute("result")%>

3. Handle error conditions-
We store the most wanted currencies, the most exchanged currencies, and the date that most transition happens. We believe this is valuable because these information give insight about the how people think about future currency value.

4. Log useful information-
The client only select from the menu, if the client choose the same currency “from” and “to”, which will be an error if we make such request to https://www.frankfurter.app/. CurrencyExchangeServlet.java will not make the request but simply set the exchange amount same as amount.

5. Store the log information in a database-
mongodb://leolin84200704:leolin84200704@ac-hdxdkkm-shard-00-00.grw8a3t.mongodb.net:27017,ac-hdxdkkm-shard-00-01.grw8a3t.mongodb.net:27017,ac-hdxdkkm-shard-00-02.grw8a3t.mongodb.net:27017/test?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1

6. Display operations analytics and full logs on a web-based dashboard-



Dashboard Results

The Top 3 currencies from which user wants to convert from-
SEK
USD
ZAR

The Top 3 currencies To which user wants to convert to-
PLN
TRY
USD

The Top 3 dates on which user converts currency -
Nov, 15, 2022
Nov, 16, 2022
Nov, 17, 2022

MongoDB Info

Date	User	Base Amount	To	Get
Nov, 15, 2022	AppUser	ISK 1000.0	AUD	10.2835
Nov, 15, 2022	AppUser	ISK 1000.0	AUD	10.2835
Nov, 15, 2022	AppUser	ISK 1000.0	AUD	10.2835
Nov, 15, 2022	AppUser	ISK 1000.0	AUD	10.2835
Nov, 15, 2022	AppUser	ISK 1000.0	AUD	10.2835
Nov, 15, 2022	AppUser	ISK 1000.0	AUD	10.2835
Nov, 15, 2022	AppUser	USD 100.0	ISK	14408.0
Nov, 15, 2022	AppUser	USD 100.0	ISK	14408.0
Nov, 15, 2022	AppUser	USD 100.0	ISK	14408.0
Nov, 16, 2022	AppUser	ISK 100.0	AUD	1.0284
Nov, 15, 2022	AppUser	ISK 500.0	ISK	500.0
Nov, 15, 2022	AppUser	ISK 500.0	ISK	500.0
Nov, 16, 2022	AppUser	ISK 678.0	ISK	678.0
Nov, 16, 2022	AppUser	USD 100.0	JPY	13922.0
Nov, 16, 2022	AppUser	USD 100.0	JPY	13922.0
Nov, 16, 2022	AppUser	USD 100.0	JPY	13922.0
Nov, 16, 2022	AppUser	USD 100.0	JPY	13922.0
Nov, 16, 2022	AppUser	USD 100.0	JPY	13922.0
Nov, 16, 2022	AppUser	RON 100.0	DKK	151.45
Nov, 16, 2022	AppUser	USD 100.0	JPY	13922.0
Nov, 16, 2022	AppUser	USD 100.0	DKK	714.99