**Project Documentation**

Project: Currency Converter

Group: Da Boiz - Patrick Agri, Jack Belmont, Sebastian Nevarez, Thuc Nguyen, Thomas Shea

Front End Code Description:

Front end code for this application includes the buttons to switch between activities, and the views we have on each activity screen. The first activity simply has a dropdown menu with all of the possible currencies you can select. It also has a textbox that allows user to input what they have of that currency. There is then an enter button that takes the user to the second activity. On this second activity there is a scrollview so the user can see all of the converted values from the main currency to target currencies. In order to fill all of this information, there is a load button which must be clicked on the second activity. In addition each labeled currency on this page is also a button which will bring the user to the third activity, which is an image screen. This will have an image of the flag of the target currency’s country, as well as an image of the actual currency. This third activity is shows a separate activity for each currency as each target currency has it’s own information that will need to get displayed. The first page drop down menu contains strings which are held in the strings.xml file on Android Studio. These strings are linked to the editText box on the first page via their identification name. On the second screen where the currencies are displayed with their respective conversions, these buttons were identified by an id name, and created one on top of another in the xml file. The textViews were placed via the android:nextTo function which placed all of the boxes next to whatever currency they were compared to. Finally on the last page, all of the pictures were originally stored in the res > drawable folder. All of these had their own identification, which allowed them to be called in the third page to be displayed. The currencies worked the same way, containing an identifier which allowed the picture to be shown in the last page.

Back End Code Description:

Back end code for this application includes functions for conversions as well as for passing values between activities. When you click the “Go!” button on the main activity, you will be brought to the second activity via a new intent in android studio that is attached to that second activity. The amount of money and the selected base currency are stored in String variable from the main activity, and are then passed to the second activity where they are utilized to calculate and return the appropriate conversions. Separately from the main and second activities, ExchangeRate.java defines two methods, get\_Index() and getSpecificRate(). get\_Index() iterates through a 2D array of currencies and corresponding exchange rates, and returns the index of the currency that is passed as a string to the method. getSpecificRate() creates a 2D array that stores 11 different currencies and the corresponding exchange rates of those currencies to the Euro. The method is passed 2 strings, one that corresponds to the base currency and the other which corresponds to the target currency. The rate is calculated by calling the get\_Index() method for each string that is passed in order to find the index of those currencies in the 2D array. It then divides the exchange rate to Euros of the base currency and divides it by the exchange rate to Euros of the target currency. The result of this division is stored in a double and returned. When the “Load” button is clicked in the second activity, each textView that appears next to each button calculates and shows the converted value by taking the passed value from the main activity and multiplying it by the rate that was returned from getSpecificRate() method (which will be different for each textView). This converted value is rounded to two decimal places and then cast to a string and then displayed by each textView. Next, once a button from the second activity is clicked, the app will jump to the final activity, also via a new intent. This activity will use image processing to display an image of the target currency nation’s flag, and a picture of the currency itself. Each currency its own intent which brings the user to the respective xml layout and java class. This was done because it allowed problems to be easily caught since each currency had its own page and class. While it might not be the most efficient way to code, this was the most effective way for us to create a successful user interface within our time frame.

What the App Does:

This app allows the user to input what they have of a certain currency, and clicking a button which will bring you to a screen displaying how much that value is worth in many other currencies. Each target currency will be available as a button on this second page, and the button will bring the user to a third page. This third page will display a flag of the nation this currency is used in, as well as an image of some of the actual currency.

App Use / Necessity:

This app would be useful for someone who is planning on going on a trip that involves going to countries with different currencies. Instead of having to look up all of the conversion rates from various currencies to various other currencies, this app gives the user the ability to find all of these conversions in one place. They will easily be able to change your current currency and your target currency, so it will be a quick and simple process to obtain the exchange information desired.

Target Users:

We would be targeting an audience of frequent international travellers with this app, as it would benefit this audience the most. These people would have to do much less work looking up conversion rates on the internet if they could just use this simple app instead. This would save time during the trip planning process, as well as during the actual trip.

API Struggles and Ideas:

After condensing the app idea from a full fledged travel application to a currency conversion application, the initial Idea was to call an API in order to get the most recent exchange rates. The team spent the majority of 4 days and and 3 team meetings attempting to work with this API. After talking with one of the TA’s (Steven), and with Professor Densmore over E-Mail, it became clear that the the team would be unable to solve the API problem in the time allotted. After consulting briefly with Professor Densmore on this issue, the Project Lead made the call to hard code the conversions. The issues faced with the API were as follows.

The team began by searching for an appropriate API to call, and settled on Fixer.io. This API is free to use and does not require an API Key, making it an ideal and theoretically easy to use candidate. With this in mind the team began researching how to call an API in java and in Android Studio. This is where the team ran into some major problems. The syntax for calling and API in java was long and overly advanced for the team which had minimal advanced java understanding. The team worked on piecing together a call to the API multiple times inside of Android Studio, and was able to craft multiple different variations of the API call that would compiled. However, everytime the app was used, the API call caused an immediate crash of the app. With more research it was found that the API call in Android Studio needs to be done in a background thread. This is where the team finally got stuck. The team was unable to find a method of doing this that would work. Some attempts would compile and crash on use, but most attempts simply would not compile. Most online sources suggested using AsyncTask to do the API process but the Team was unable to determine a successful way to do this, despite the online tutorials utilized. Other suggested methods for using the API was to use external downloadable libraries for use in the background API calls. These suggestions were attempted as well, and continue to result in application crashes.

The group did remember to request internet access in the app, so we know that this was not the issue. After almost 5 days of struggling with the API call the group ran out of time to continue work on it.