Offline Coding Challenge

Introduction

Welcome to the Mama Money offline coding challenge!

We have chosen to center the problem around a real world scenario rooted in one of the features we have developed. Hopefully this can give you some insight into some of the work that we do at Mama Money. It's also a good way for us to assess your design and implementation of a solution to a given problem.

This problem is aimed at mid to senior developers and has been left intentionally vague in implementation details to see how you think about the problem and approach the solution. However, if there is something that has been omitted or is unclear that is blocking you please feel free to reach out and we will assist.

Problem Statement

USSD (Unstructured Supplementary Service Data) is one of the channels that our customers use to send money home with us. It is particularly relevant in lower income communities as there is no requirement to have a fancy phone with access to our mobile app and there are no data costs.

The high level USSD flow is to dial a code on a mobile phone which will send a request to a WASP (Wireless Application Service Providers) who will convert it into a standardised message that is sent to a web application for processing. The response from the web application will contain content that is converted by the WASP to be displayed in a menu format on your cell phone. This interaction is repeated multiple times as you move through a numbered menu until you have completed a transaction of some sort.

Unfortunately if you are not registered with Mama Money, you won't have access to our USSD menu - found at ***134*542#**. However, you should be able to see the interaction using your cellphone provider code:

Provider	Code
Cell C	*147#
MTN	*136#
Vodacom	*111#

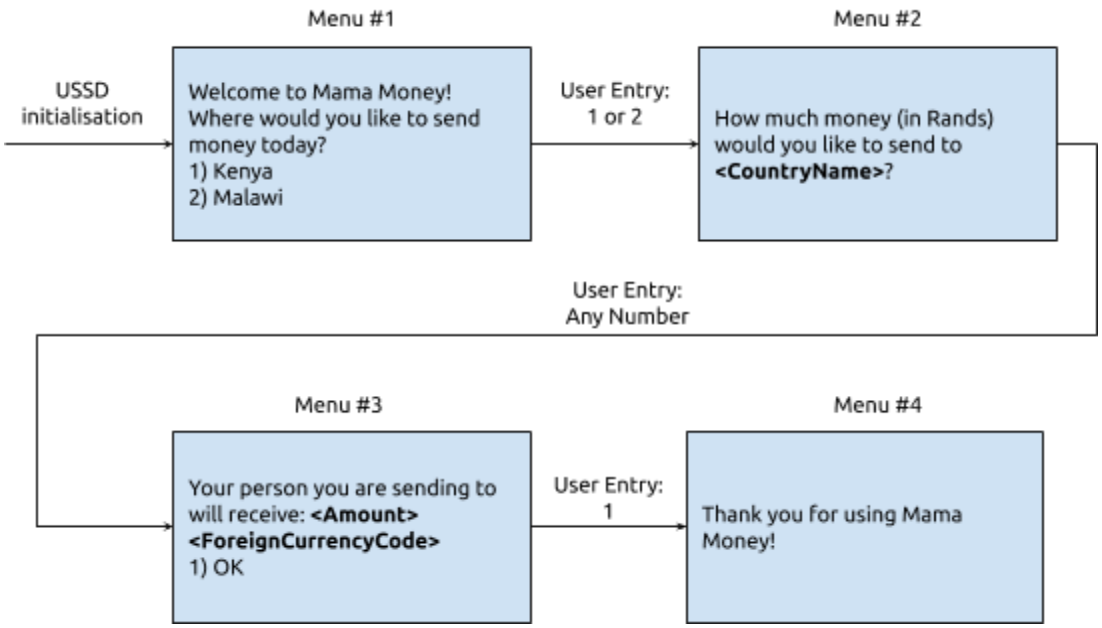
We are looking for a means to work around the two main restrictions we face when using USSD as a messaging system:

1. The structure of the requests and responses is consistent for every USSD call into the application. Therefore we are not able to construct complex objects in a single call, instead we build objects over a series of interactions.

2. All requests from the WASP are routed to the same URL. This implies that there is no way to use endpoint or request data to figure out what menu item we are moving into, it must be determined by the application.

This challenge will require you to build a web application that will simulate someone using USSD to get a quote for sending money to either Kenya or Malawi. This will be done through a series of REST calls to a single endpoint. This endpoint will be responsible for figuring out which previous interactions have already been performed for a user in a session and respond with the text representing the next prompt.

The menu traversal you are looking to simulate is as follows:



Each arrow represents a REST request that will be made to your application, where *User Entry:* is the text that will be sent in the request. Each menu represents the text that is returned in the REST response and is determined by where a user is on the menu. Values in **<angle braces>** are variables that are informed by previous choices:

- **<CountryName>** in *Menu #2* is taken directly from *Menu #1*
- **<Amount>** in *Menu #3* is the amount of foreign currency to be received, this will be calculated by your application
- **<ForeignCurrencyCode>** is determined by the county selection in *Menu #1* and will be one of:
    - **KES** - Kenyan Shillings
    - **MWK** - Malawian Kwacha

Requirements

You are required to build a web application that will expose a /ussd endpoint (this can be served locally or cloud hosted) that receives POST requests with the body containing a JSON representation of a UssdRequest object:

Parameter	Required	Data Description
sessionId	Mandatory	String A session identifier assigned by the WASP, it will be consistent for a user across multiple interactions.
msisdn	Mandatory	String This represents a cell phone number and can be considered unique per user.
userEntry	Optional	String This represents the user's text/number entry from their cellphone. It can be left absent or blank for the initial interaction.

Example:

```
{
  "sessionId": "#AA1234",
  "msisdn": "27821231234",
  "userEntry": "1"
}
```

As explained above, your application will then be required to determine where the user is in the menu flow so that the appropriate text can be returned in a JSON representation of a UssdResponse object:

Parameter	Required	Data Description
sessionId	Mandatory	String Echoed from the request
message	Mandatory	String This represents the User Entry. It can be left absent or blank for the initial interaction.

Example:

```
{
  "sessionId": "#AA1234",
  "message": "Welcome to Mama Money! Where would you like to send money
to?\n1) Kenya\n2) Malawi"
}
```

Through a series of these requests and responses you will be building up a more complex object that is composed of the choices a user made, i.e. What country is money being sent to? How much money is being sent? How much money will be received?

On the transition between *Menu #2* and *Menu #3* you will have noticed that there is a currency conversion between Rands and either Kenyan Shillings or Malawian Kwacha. You can use the following currency conversion for each:

- ZAR - KWS: **6.10**
- ZAR - MWK: **42.50**

Upon reaching *Menu #4* the user journey should be considered completed. Any subsequent calls should restart the journey.

We will primarily be looking at how well your solution works as well as how you have designed and organised the components. We will also be looking at how well your solution caters for edge or fail cases that haven't explicitly been defined in this document.

There are no restrictions in terms of dependencies. Feel free to use any frameworks or libraries that speed up your development and you feel are most appropriate for the requirements of the application. Just ensure that you have provided clear instructions on how to build and run the application.

You can submit your solution by providing a zip file with the source code or sending us a link to a github repo.

Good luck and have fun :)