CS478: Software Development for Mobile Platforms

Project #5

Due time: 11:59 pm on 12/7/2018
Submit using Blackboard web site
Total points: 100
Instructor: Ugo Buy

HoangMinh HuynhNguyen and Dhananjay Gupta

This project consists of two Android apps, a client app called *FederalMoneyClient* and a server app called *FederalMoneyServer*. *FederalMoneyClient* takes as input a date from the device user. This app packs the date in a request to the *FederalMoneyServer* app, which queries a database maintained by the US Department of the Treasury. *FederalMoneyClient* uses a worker thread to make the request, as the service app may take a long time to return the requested data.

The database is at URL http://treasury.io/ and supports most networking protocols, such as URL connections and HTTP Clients. In order not to block its user interface during network operations, *FederalMoneyServer* uses a service running in a worker thread when querying the government database.

The service in the *FederalMoneyServer* app defines a simple API, described below, for use by *FederalMoneyClient* and other apps. Upon receiving an API call from a client, *FederalMoneyServer* appropriately creates and formats a query to be forwarded to the *treasury.io* site. After it receives the site's response to the query, *FederalMoneyServer* will return the requested information back to the client as the value returned from the API call.

The API exposed by *TreasuryServ* consists of the following two remote methods:

1. *monthlyAvgCash(int aYear) : List(Integer)* — This method takes as input an integer denoting a year between 2006 and 2016 inclusive. It returns a list of 12 integer values denoting the average amount of cash the US Government had on hand in each month of the input year.

2. *dailyCash(int aYear, int aMonth, int aDay, int aNumber) : List(Integer)* — This method takes as input 4 integers: a day, a month, a year, and a number of working days. The first 3 integers denote a date in the years 2006–2016. The last integer denotes a number of working days between 5 and 25. This method returns a list of integers denoting the cash the government had on hand at the opening of the input date and subsequent working days, as specified by the fourth input parameter.

In addition to the service, *FederalMoneyServer* defines an activity whose sole purpose is to display the status of the service. The service must be both *started* and *bound*. The service could either be (1) not yet started, (2) started, but not bound to a client, (3) started and bound to one or more clients, (4) bound but not started and (5) destroyed. The reason to *start* the service is to let the service continue running even when it is not bound to any clients. It is responsibility of *FederalMoneyServer*'s activity to start the service, as soon as the activity is displayed, if the service is not running

App *FederalMoneyClient* has two activities. The first activity allows the interactive user to specify a request to be forwarded to *FederalMoneyServer*. Define appropriate fields (widgets) allowing a user to select one of the two APIs supported by the service in *FederalMoneyServer* and the corresponding input parameters. An additional widget forwards a request to the service. If *FederalMoneyClient* was not bound

to the service, it will bind to the service at this point. Finally, this activity contains a widget that will unbind *FederalMoneyClient* from the service. The second activity consists of a list of data returned by the service. This activity is opened when the service request returns from the *FederalMoneyServer* app.

Implementation notes. You must use an AIDL spec to expose the service's functionality to the clients. Make sure that the service's code is thread-safe; multiple clients could be bound to the service at the same time. As your implementation platform, use a Pixel 2 AVD running the usual Android version (API 27—Oreo 8.1). Design your client app layout in such a way that it will display best in landscape mode. You are not required to provide backward compatibility with previous Android versions or to support device reconfigurations.

Hints. Use class *HttpUrlConnection* to forward a request from your service to the *treasury.io* site. Use class *URLEncoder* to create the query portion of the URL sent to the HTTP client. Use an appropriately formatted URL, containing a suitable SELECT statement, to query the database. The URL should start with the following string http://api.treasury.io/cc7znvq/47d80ae900e04f2/sql/?q= followed by the SELECT query.

You must work alone on this project. Submit a zip archive containing two root directories; each directory contains the full Android Studio repository of the corresponding app. No late submissions will be accepted.