95-702 Distributed Systems

Project 4

Project Topics: Mobile to Cloud application

This project has 3 tasks.

Task 0 involves researching and selecting a 3rd party API that you will use in Task 1&2.

Task 1 will build on the Deploying to the Cloud Lab and the Android Lab. You will design and build a simple mobile application that will communicate with a RESTful web service in the cloud.

Task 2 will add an operations logging and analysis function to your Task 1 web service.

When completing these tasks, the student should reflect on synchronous and asynchronous calls, event handling, remote interfaces, mobile and cloud computing.

Task 0 Choose a 3rd Party API

Research and choose a 3rd party API to use in Task 1 and 2. Be sure to read Task 1 carefully to see how you will use the API. Experiment with the API you choose enough to make sure that you want to commit to using it for your project.

A Canvas quiz (Project 4 API) will be used to submit your choice. You will need to provide:

a. The name of the API (e.g. Flickr.com)

b. The URL of the API documentation (e.g. https://www.flickr.com/services/api/)

c. A short description (1-3 sentences) of what your mobile application will do with the information from the API (e.g. My mobile app will prompt the user for a string and then search Flickr using their API to display an interesting picture tagged with that string.)

d. Does the API require authentication? Yes or No. If yes, what authentication (e.g. OAuth, OAuth2)?

Therefore be sure to do some research, experimentation, and planning before you submit your answer to Task 0.

Task 1 Mobile to Cloud Application

Design and build a distributed application that works between a mobile phone and the cloud. Specifically, develop a native Android application that communicates with a web service that you deployed to Heroku.

The application must be of your own creative design. It can be simple, but should fetch information from a 3rd party source and do something of at least marginal value. For example, we have assigned projects that generate hash values, implement clickers, and securely transmit sensor information. Your application should do something similarly simple but useful (but you should not reuse our ideas or the ideas of your peers!).

Your web service deployed to Heroku should be a simple REST*ful* API as you have developed in prior projects. You do NOT have to implement all HTTP methods, only those that make sense for your application. Your web service must fetch information from some 3rd party API. In Project 1 we experimented with screen scraping, therefore that is **not** allowed in this project. Rather, you must find an API that provides data via XML or JSON. One good place to look for such APIs is ProgrammableWeb. (http://www.programmableweb.com/apis/directory)

Use APIs that require authentication with caution. Many APIs will require you get a key (e.g. Flickr, which you used in the Android lab, required an API key). This is ok. But APIs that require authentication via OAuth or other schemes add a lot of work. Experiment ahead of time, but if you are brave, go ahead…

Banned APIs: There are a number of APIs that have been used too often and are no longer interesting in this class. Therefore you **cannot** use any of the following:

- Any weather API
- Flickr (for we have already done that)
- Spotify
- Google Maps (unless you also use a 2nd API to get info to put on the map)
- OpenMovieDatabase
- NYTimes top stories, news wires, popular, and books APIs. If you want to do something creative with one of the other NYTimes APIs post a message to Piazza.

Users will access your application via a native Android application. **You do not need to have a browser-based interface for your application** (only for the Task 2 dashboard). The Android application should communicate with your web service deployed to Heroku. Your web service is where the business logic for your application should be implemented (including fetching information from the 3rd party API).

In detail, your application should satisfy the following requirements:

1. Implement a native Android application

1.1. Has at least three different kinds of views in your Layout (TextView, EditText, ImageView, etc.)

1.2. Requires input from the user

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

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

1.5. Displays new information to the user

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

2. Implement a web service, deployed to Heroku

2.1. Using an HttpServlet to implement a simple (can be a single path) API. (It is recommended that you do **not** try to use JAX-RS / Jersey.)

2.2. Receives an HTTP request from the native Android application

2.3. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response.

- -10 if you use a banned API
- -10 if screen scrape instead of fetching XML or JSON via a published API

2.4. Replies to the Android application with an XML or JSON formatted response. The schema of the response can be of your own design.
Alternatively, you can adopt a standard schema that is appropriate to your application. (E.g. Common Alerting Protocol if your application deals with emergency alerts.)

- -5 if information beyond what is needed is passed on to the Android app, forcing the mobile app to do more computing than is necessary.

Writeup

Because each student's mobile/cloud application will be different, you are responsible for making it clear to the TAs how you have met these requirements, and it is in your best interest to do so. You will lose points if you don't make it clear how you have met the requirements. Therefore, you must create a document describing how you have met each of the requirements (1.1 – 2.4) above. Your writeup will guide the TAs in grading your application. See the provided example (Project4Task1Writeup.pdf) for the content and style of this document.

Task 2 Web Service Logging and Analysis Dashboard

For Task 2, you are to embellish your web service to add logging, analysis, and reporting capabilities. In other words, you are to create a web-based dashboard to your web service that will display information about how your service is being used. This will be web-page interface designed for laptop or desktop browser, not for mobile. In order to display logging and analytical data, you will have to first store it somewhere. For this task, you are required to store your data in a noSQL database, or more specifically a MongoDB, database hosted in the cloud.

IMPORTANT NOTE: Task 2 builds on Task 1, but for your own safety, you should not overwrite Task 1. Rather, once you have Task 1 working, you should create a separate Task 2 project. In this way you will never lose the working Task 1 that you are required to submit. When deploying to Heroku, you should deploy Task 1 and Task 2 separately. Heroku allows you do have two applications. In this way, if Task 2 does not work for some reason, we still have Task 1 to grade.

Logging data

Your web service should keep track (i.e. log) data regarding its use. You can decide what information would be useful to track for your web application, but you should track at least 6 pieces of information that would be useful for including in a *dashboard* for your application. It should include information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone. Information can include such parameters as what kind of model of phone has made the request, parameters included in the request specific to your application, timestamps for when requests are received, requests sent to the 3rd party API, and the data sent in the reply back to the phone.

You should NOT log data from interaction with the operations dashboard, only from the mobile phone.

Database

You should log your data persistently so that it is available across restarts of our application. For this task you should use MongoDB to store your logging data. MongoDB is a noSQL database that is easy to use. By incorporating it into your web service you will gain experience using a noSQL database, and experience doing CRUD operations programmatically from a Java program to a database.

The main MongoDB web site is https://www.mongodb.com. The site provides documentation, a downloadable version of the database manager application (*mongod*) that you can run on your laptop, and MongoDB drivers for many languages, including Java.

Mongod is the MongoDB database server. It listens by default on port 27017. Requests and responses with the database are made via a MongoDB protocol.

Mongo (without the DB) is a command line shell application for interacting with a MongoDB database. It is useful for doing simple operations on the database such as finding all the current contents, or deleting them.

Studio 3T (https://studio3t.com) is a free MongoDB IDE that you can use to view and edit mongoDB databases visually.

Because your web service will be running in the Heroku PaaS, you can't run your database on your laptop. Rather, you should use a MongoDB-as-a-Service to host your database in the cloud. mLab (https://mlab.com) is recommended because it has a free level of service that is adequate for your project.

There are two ways to create an mLab MongoDB database:

a. Create a login on mLab and in the free *sandbox* tier create a database to use. (see: https://mlab.com)
b. Provision a free *sandbox* tier mLab MongoDB database from within Heroku (see: https://elements.heroku.com/addons/mongolab)

In either case, you can access this cloud-based MongoDB database from your laptop as well as from Heroku.

MongoDB drivers exist for many languages and provide language-specific APIs for interacting with a MongoDB database, whether on the same host (i.e. your laptop) or a remote host (e.g. MongoDB in the cloud). You don't have to learn the MongoDB protocol because the MongoDB driver for Java does that for you. You do, however, have to understand how to use the MongoDB Java Driver. It can be found here: http://mongodb.github.io/mongo-java-driver/.
Alternatively, you can use the mLab's RESTful Data API:
http://docs.mlab.com/data-api/

Dashboard

The purpose of logging data to the database is to be able to create an operations dashboard for your web service. This dashboard should be web page interface for use from a desktop or laptop browser (not a mobile device).

The dashboard should display two types of data:

a) Operations analytics – display at least 3 interesting operations analytics from your web service. You should choose analytics that are relevant to your specific web service. Examples for InterestingPicture might be top 10 picture search terms, average Flickr search latency, or the top 5 Android phone models making requests.

b) Logs – display the data logs being stored for each mobile phone user interaction with your web service. The display of each log entry can be simply formatted and should be easily readable.

You will likely find HTML tables useful for formatting tabular information on a web page. And there are plenty of examples of embedding data in tables with JSP on the web. No frameworks are necessary for this, just < 20 lines of JSP (i.e. mixed HTML and Java).

Please read carefully…

This task will challenge you to do a lot of research to understand enough MongoDB to create a simple database, add a collection, and insert, update, and find documents in that collection. This is very much like you will need to do regularly in industry. Code examples are provided on the MongoDB site, and elsewhere. **As long as you include comments as to their source, you can use them in your code.** If we search for a snippet of your code find it somewhere, and you have not attributed it to where you found it, that will be cheating and reason for receiving a failing grade in the course. Of course, the bulk of your code that is unique to your application should be your own and not copied from anywhere.

In detail, your solution should satisfy the following requirements:

1. Log useful information

At least 6 pieces of information is logged for each request/reply with the mobile phone. It should include information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone. (You should NOT log data from interactions from the operations dashboard.)

2. Store the log information in a database

The web service can connect, store, and retrieve information from a MongoDB database in the cloud.

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

3.1. A unique URL addresses a web interface dashboard for the web service.

3.2. The dashboard displays at least 3 interesting operations analytics.

3.3. The dashboard displays the full logs.

4. Deploy the web service to Heroku

Deploy the web service to Heroku. This web service should have all the functionality of Task 1 but with the additional logging, database, and dashboard analytics functions.

In your Task 2 writeup be sure to include the dashboard URL!

Task 2 Writeup

In the same style as Task 1, but in a separate document, describe how you have met these 4 requirements.