Colleen Minor

Take-Home Final Exam

Mobile and Cloud Development

Summer 2016

08/14/2016

Airbnb Mobile: A Reflective Evaluation

Usability

Pros:

Because I used Boostrap buttons by Bearded-Hen (https://github.com/Bearded-Hen/Android-Bootstrap) I was able to provide the user with feedback about what sorting option they had chosen. For example, on the ListOfListings page, if a user has sorted by "listings in your city," the listings in your city button will be the only one colored in. The same is true when they hit "listings in your state" or "View All Listings."

Another good thing is that I made it so that when a user is adding a comment, they will see "You are commenting as [username]" right above the comment box. This informs the user that their comment will not be anonymous and reassures them that they are logged into the correct account.

I also prevented a user from accidentally deleting a listing that they do not mean to by forcing them to type the name of the listing that they wish to remove in order to delete it. I think most users would be grateful for this.

Cons:

I wanted to make it so that you have the option to log in and navigate to different pages through the menu, but I had trouble getting the menu options to work. I ended up having to use a lot of buttons and it does not make for the most intuitive navigation. For example, a user might not guess that to get from the ListOfListings page back to the log-in page, they should click 'your profile' and there will be a 'register/log-in' button on that page. They also wouldn't know that just because they are at the register/log-in page, that does not mean they have been logged out: The buttons on the register/log-in page seem to indicate that the only thing a user can do to move forward is to register or log-in. To make this app more intuitive, I would make it so that if a user is logged in (if there is a value in the 'username' key in their preferences file), the log-in/register buttons are hidden and instead there is a 'log out' button that will remove that value from the user's preferences, or buttons that the user can use to navigate around the app.

Also, from the "ListingWithComments" page, there the buttons only allow for the user to navigate to the 'ShowTheMap' page (or the update page if they own the listing). I would make the buttons on this page consist with the ListingWithMap page.

An inconsistency that a user might not expect is that on the 'UserActivity' page, where the user views their activity, clicking on listings that they have posted will take them to that particular listing, but clicking on comments that they have posted will not take them to the listing on which they have posted that comment. I would improve this by setting on onClickListener on each comment in the list that goes to page where the comment is.

Portability

Well for one, my project is strictly for Android devices. But the main portability flaw is to do with the way that I designed the user location feature.

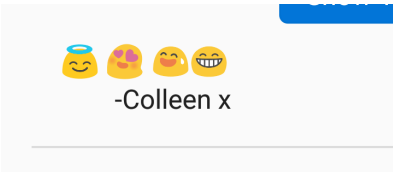
My app first retrieves the user's latitude and longitude coordinates with the help of Location Manager by yayandriod: https://github.com/yayaa/LocationManager. Then it takes those latitude/longitude coordinates and puts them into getListingsByLocation (in ListOfListingsActivity), where they are used in a GET request to googleapis.com/maps/api/geocode/json?latlng= [user's latitudinal and longitudinal coordinates]. While trying to find a way to get the user's city and state, I looked at the JSON results for 2 or 3 sample coordinates so that I could look for a pattern as to how to parse the JSON to get the city and state. I decided that, based on these few samples, I could get the city by taking the String "formatted_address" and grabbing whatever value was between the first and second commas, and I could get the state by taking the JSON array "address component" and getting the String in the object at 'administrative level 1.' This seemed to work during testing. What I didn't anticipate: Today I tried my app after turning my location mode back to normal (it had been on "GPS only" mode before, so you could see my location change in the demo), and I was very surprised to see my exact address, to a letter, displayed when I pressed "only show listings in my city!" The results of this is that the listings displayed were not narrowed by city at all, only by state. This was a major design flaw. I realize now that the Android library has a class built into it called "Geocoder" that would have allowed me to get the address objects more easily, instead of doing an HTTP request (top answer here:) https://stackoverflow.com/questions/2296377/how-to-get-city-name-from-latitude-and-longitude-coordinates-in-google-maps. Or even if I were to not use Geocoder, I should have done what the

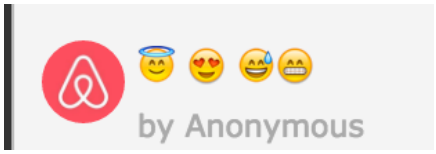
second-top answer in that stackoverflow thread did and parse the city in more specific terms instead of assuming that it would always be between the first and second comma in the “formatted_address” String.

Another portability issue is that I limited the location feature to the United States on the Android app (by specifying the country as USA in the call to maps.googleapis.com) and to countries that have ‘states’ in the API (by making city and state the location fields ‘city’ and ‘state’). This basically limits both listings and users (who want to use location features) to within the United States.

Interoperability

I would say that my API (talking about the back-end, not the mobile app) is pretty interoperable by effect of using the JSON format. Many of its URIs return only JSON, so those can be used in lots of different applications. The mobile app and API interoperate with each other: Right now there is the website up, which has a whole online UI, and everything that gets posted on there goes into the Android app, and vice-versa. As for the mobile end, that is not really meant to communicate with anything other than the back-end and itself. I think native Android apps in general work well with the Android system though, and one cool thing is that, because Android uses the Unicode 6.0 standard for its emoticons, if a user adds emotions to a comment with the Android keyboard, emoticons (slightly different looking ones) will also show up in the comment that gets posted to the website. For example, here is the same comment on the mobile app (left) and web interface (right):





As for the interoperability of individual parts of my app, I wish that I had separated the code out more so that more functions could be reused. I have a lot of code that is copy/pasted from other code and little changes. I wish that I had made more universal functions outside of the activities that I could use in other apps.

Security

This was a feature that I could have done much better on. At the time that I made the project, I thought to encrypt the user's passwords on the server side, so that the passwords would get stored on mlab.com encrypted, but somehow it didn't occur to me to encrypt the passwords on their way from the Android device to my AWS server. To be secure, I should have encrypted on both sides. I'm not sure of how the implementation would go exactly, but apparently the Android Library has a class called Cipher for encryption and this person (2nd-top answer https://stackoverflow.com/questions/7787773/encrypt-with-node-js-crypto-module-and-decrypt-with-java-in-android-app) describes how they used that and node's crypto model to encrypt and decrypt between Android and node.js (my server side runs on node.js). Also I did not encrypt the username before storing it on the user's device, which I should have done.

You could also make the case that storing the user's username on their device is not great security, and it would be better to just use OAuth or something. Personally

though I don't like having to log into apps every time, and my ideal situation with my app would be one where the log-in options only showed up if the user had chosen to log out (see usability).

Reliability

I set my minimum SDK to 15, which is 4.0.3/4 Ice Cream Sandwhich from December 2011, so it will not run on devices with an older OS than that. I only tested my app in a few devices because for my application, when you want to run it on an emulator, you have to add install Google Play services on it just to view the google maps, and then you have to configure the emulator's GPS and location settings every time you want to use location services. I did test it in one genymotion emulator meant to imitate a nexus 4 with a recent OS (5.1) and everything works fine on that, but it would take a while to test the full range of devices. On both my device and the emulator though, the buttons on the map page get all messed up when the app is in landscape mode (orientation when the phone is on its side). On the pros side, I've used the app pretty extensively on my device and it doesn't crash on there, so there is that.

Performance

I noticed while I tweaking the GPS settings on my emulator that it listed my app as having "high battery" usage. I googled "high battery location settings" and I found out that "'High battery use' gets information from sources that may use more power"

(https://support.google.com/accounts/answer/6179507?hl=en). I had been thinking it was really good that my app could get the user's location from multiple sources (both GPS and network), since a lot of apps don't seem to do that, but I guess that's a battery drain.

I'm impressed with how quickly my lists of listings load, since the app does a new GET call to the API each time the user sorts in a different way, but I know that this could get monetarily expensive on the server side. Plus I don't know if it would run out of RAM or something if I had a big userbase and many calls were being made at once. I should probably cache some of the data and sort it in the app, instead of doing a new call every time a user wants to do a new sort. Currently a GET request is made pretty much every time a user navigates to any page in the app.

Welp, that just about wraps up my project. Thank you for the great course!!! 😀