Spring 2016

CMPE235\_Lab1

**Android Application - Smart tree**

Submitted by Khanh Dao

Table of Contents

Design & Use Case 2

Implementations 3

File structures 3

Configurations 4

OOP concept 5

Notable features 7

User flows & Error Handling 7

Publish to apps store 10

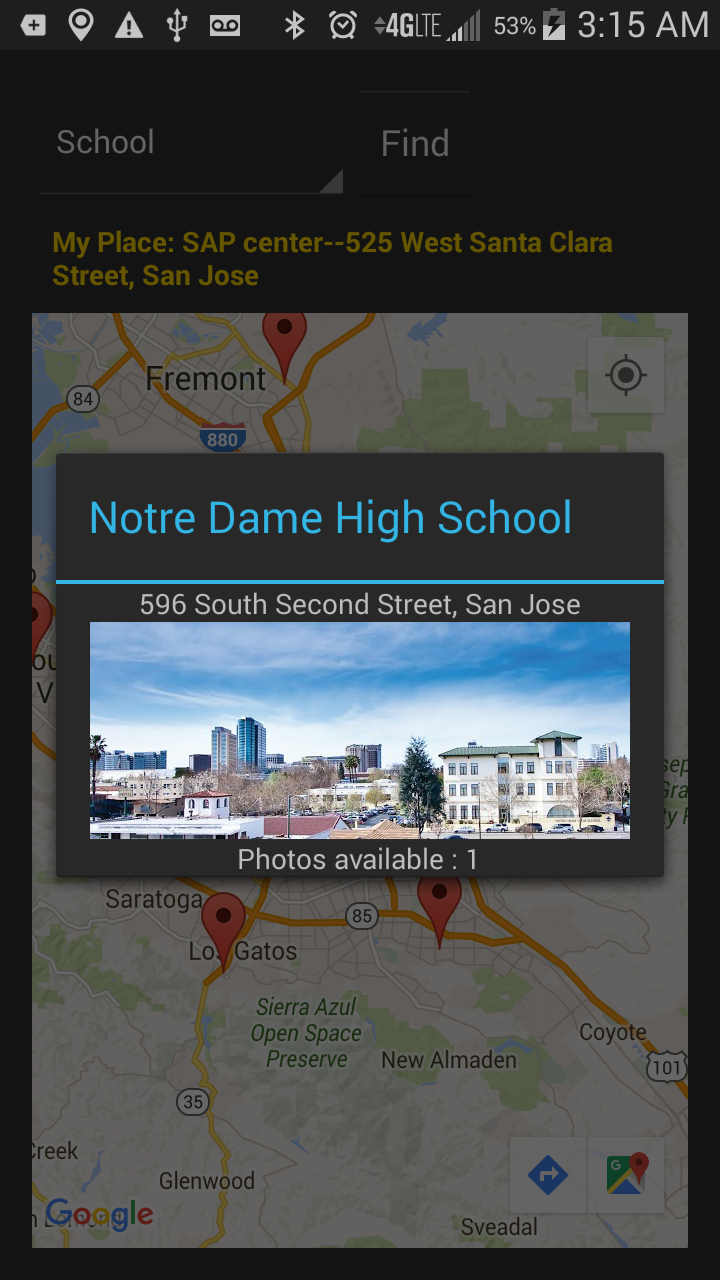
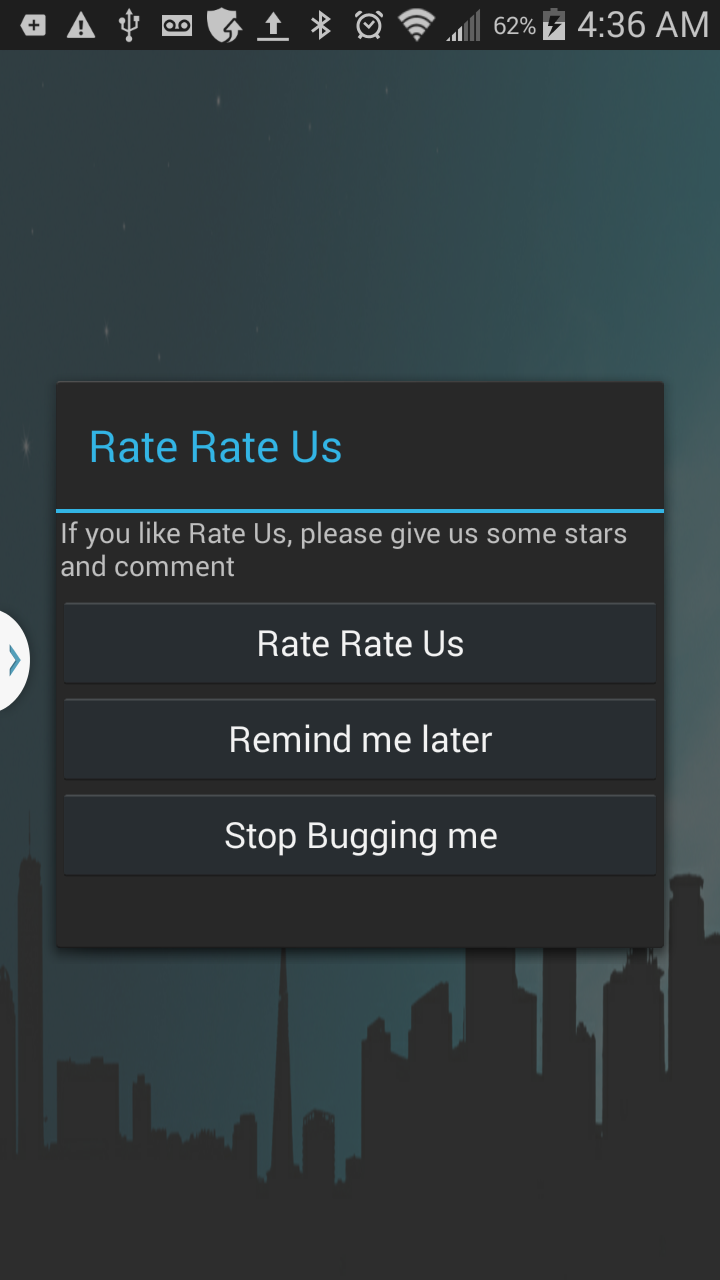
Lessons Learned 12

TODO 12

Conclusion 12

References 13

## Design & Use Case

Designing a smart tree application, where scanning a given barcode will return a specific location’s details (including name, details description, address, nearby place). We also provide user the capability to take pictures and video and share it via “pre-installed” sharing apps. To make it simple, yet descriptive, we also implement error messaging to notify user if an invalid barcode is scan.

For this lab exercise, I choose to implement the app as “Android application”, the homepage UI as following (image attached). The idea of this design is using “popular icon” centric, with suburb style color to reflect the common use of this app (eg: search for location’s information in the city). I personally choose “blueish” and “gold” color for this app, re-present SJSU/San Jose downtown color. When user click on each icon, they will get directed to corresponding pages for scanning barcode, take pictures or video or leave their comments about the apps.

The followings are some screen shots for the app (how it’s displayed among other apps, home page, map, rating page…)

## Implementations

### File structures

I follow the standard Android apps structure and create “activity class”, which corresponds to each page in the app as following.

|  |
| --- |
| ├── BarCodeActivity.java  ├── CameraActivity.java  ├── CommentActivity.java  ├── InteractActivity.java  ├── MainActivity.java  ├── MapsActivity.java  ├── SharingActivity.java  ├── SplashActivity.java  └── data  ├── Location.java  └── Tree.java |

As could be seen in the structure above, we have 6 main different activity class, corresponding to 6 different pages supported by this app. We also implement SplashActivity, which is initial page load on app start before navigating to MainActivity (Note: this is optional, we implement SplashActivity as common best practice for a good android app)

We also follow best practice of Java OOP program, by maintain Location.java and Tree.java class, which contains getter/setter for information of a tree item (see source code for detail implementation)

### Configurations

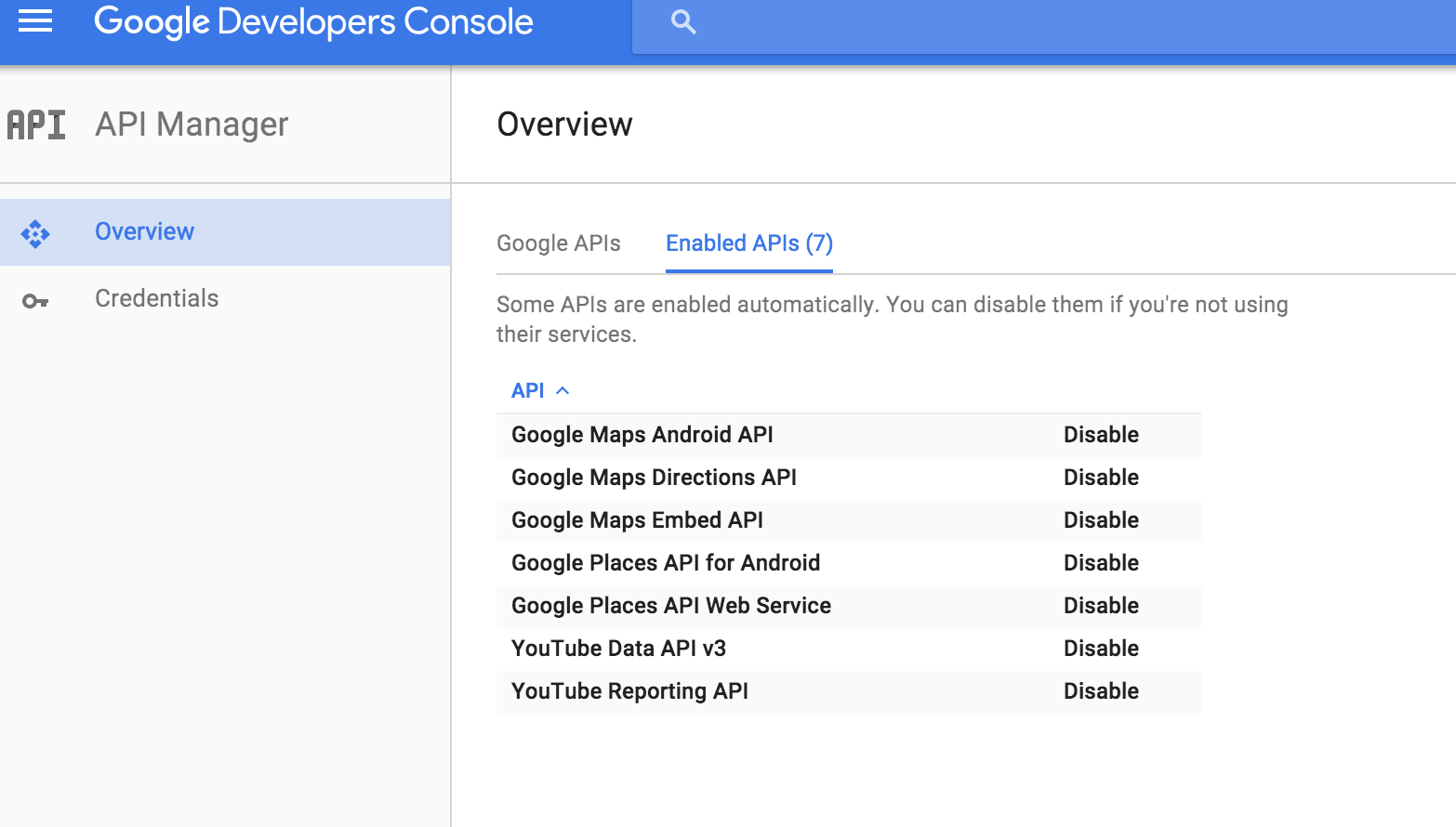
* To support camera, video and map feature, we will need to configure user-permission in AndroidManifest.xml as following

|  |
| --- |
| <**uses-permission android:name="android.permission.INTERNET"** />  <**uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"** />  <**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />  <**uses-permission android:name="com.google.android.providers.gsf.permission.READ\_GSERVICES"** />  *<!--*  *The ACCESS\_COARSE/FINE\_LOCATION permissions are not required to use*  *Google Maps Android API v2, but you must specify either coarse or fine*  *location permissions for the 'MyLocation' functionality.*  *-->*  <**uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"** />  <**uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"** />  *<!-- Allow access to android camera -->*  <**uses-permission android:name="android.permission.CAMERA"** />  <**uses-feature android:name="android.hardware.camera"** />  <**uses-feature android:name="android.hardware.camera.autofocus"** />  <**uses-permission android:name="android.permission.RECORD\_AUDIO"** /> |

* We also need to provide google\_api in res/values/google\_maps\_api.xml for map feature enabling.
* We also need to use google “**zxing**” library for **QRCode** scanner support. That library will be put in build.gradle as following
* “**YouTube**” jar file also be bundled to support Youtube video embedded in the app

|  |
| --- |
| dependencies {  compile fileTree(include: [**'\*.jar'**], dir: **'libs'**)  testCompile **'junit:junit:4.12'**  compile **'com.android.support:appcompat-v7:23.1.1'**  compile **'com.android.support:design:23.1.1'**  compile **'com.google.android.gms:play-services:8.4.0'**  compile **'com.google.android.gms:play-services-maps:8.4.0'**  compile **'com.google.android.gms:play-services-location:8.4.0'**  compile **'com.journeyapps:zxing-android-embedded:3.0.2@aar'**  compile **'com.google.zxing:core:3.2.0'**  compile **'com.github.jd-alexander:library:1.1.0'**  compile files(**'libs/YouTubeAndroidPlayerApi.jar'**)  } |

* To ensure google API features (map, youtube) works, we will need to make API access is enabled in google dev account



* For debugging and monitoring purpose during development process, we include “android.util.Log” in every java class to log steps/infos that might be helpful to track down possible issue. The common patterns across class files are

|  |
| --- |
| Log.*i*(**">>>>URL<<<"**, url[0]);  Log.*d*(**"Background Task"**, e.toString());  ... |

### OOP concept

I follow Java standard OOP coding to maintain all the data object, with getter()/setter() used. The object include Tree & Location (for storing and displaying place resulted through barcode), and Place, Photo, Attribute(for maintain result of specific place with photo resulted from google map). I also set-up util class such as Http() for GET request, image upload, JSONParser(), Config() to store constant value. These features might be re-used again for Lab2 and Project (so we will keep it decoupled for better reusable purpose). For example, Location and Tree are defined as below:

|  |
| --- |
| **/\*\* Define Tree Object \*\*/**  **public class Tree {**  **String name;**  **String description;**  **String videoId;**  **int icon;**  **Location location;**  **public Tree(String description, int icon, String videoId, Location location) {**  **this.description = description;**  **this.icon = icon;**  **this.videoId = videoId;**  **this.location = location;**  **}**  **…**  **}**  **/\*\* Define Location Object \*\*/**  **public class** Location {  **double latitude**;  **double longitude**;  String **address**;  String **name**;  **public** Location(**double** latitude, **double** longitude, String address, String name) {  **this**.**latitude** = latitude;  **this**.**longitude** = longitude;  **this**.**address** = address;  **this**.**name** = name;  }  …..  } |

More detail implementations will be included in source code.

For demo purposes of this lab, I will not use DB to store data, yet store a set of static data as followings: (since we have object structured in standard Java convention, therefore, the switch to connect to DB in the future will be transparent with android-activity classes)

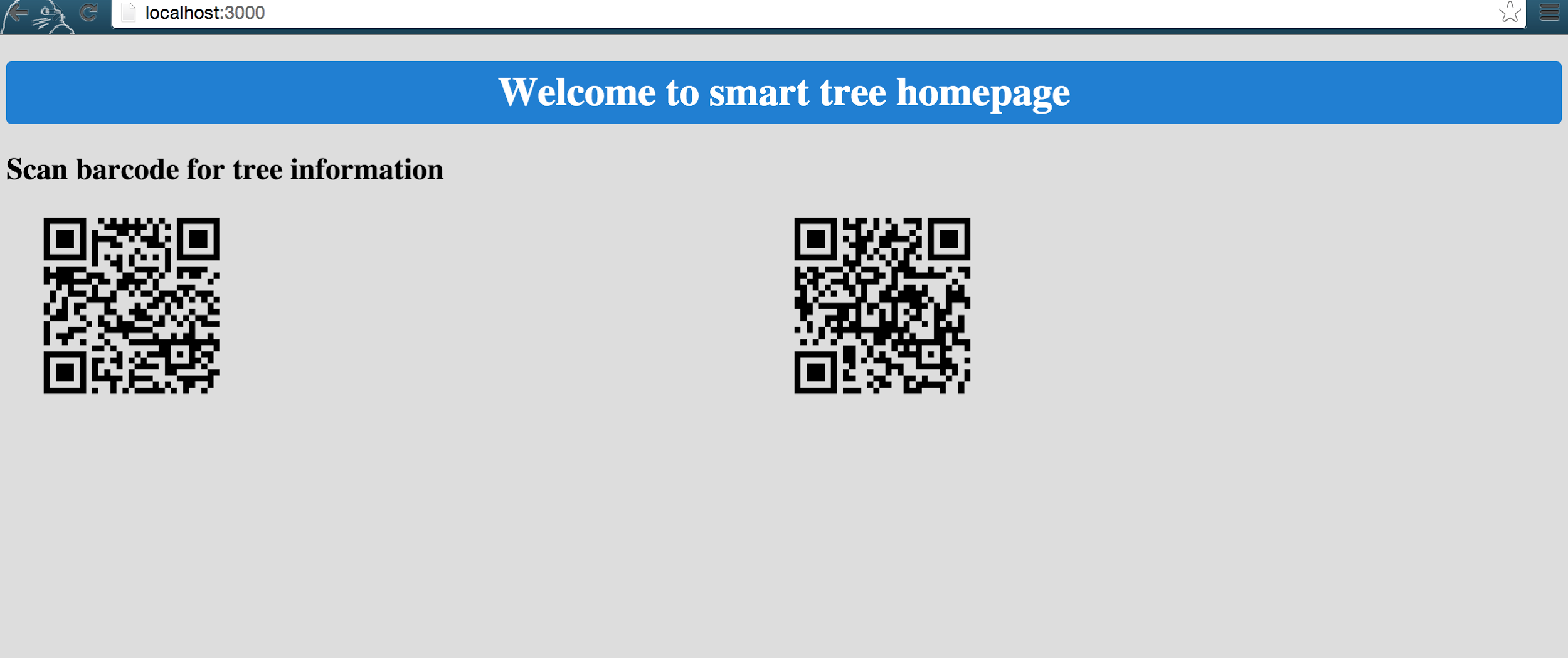
|  |
| --- |
| **private HashMap<String, Tree> hmTrees= new HashMap<String, Tree>();**  **…**  **hmTrees**.put(**"45304c60-9eac-48bf-9d0b-c02dda6c6cb3"**, **new** Tree(**"Landmark art deco-style "** +  **"theater presenting Broadway musicals"** +  **" & "** +  **"ballet & dance performances."**,  R.drawable.***barcode\_icon***, **"0fAgFUiBmQE"**, **new** Location(37.32, -121.89, **"255 S Almaden Blvd, "** +  **"San "** +  **"Jose, CA 95113"**, **"San Jose Center for the Performing Arts"**)));  **hmTrees**.put(**"8f14886c-d267-44b8-8518-8cf363634929"**, **new** Tree(**"Home of Shark"**,  R.drawable.***barcode\_icon***, **"HZS3cWlr4AI"**, **new** Location(37.33, -121.90, **"525 West Santa Clara "** +  **"Street, San Jose"**, **"SAP center"**))); |

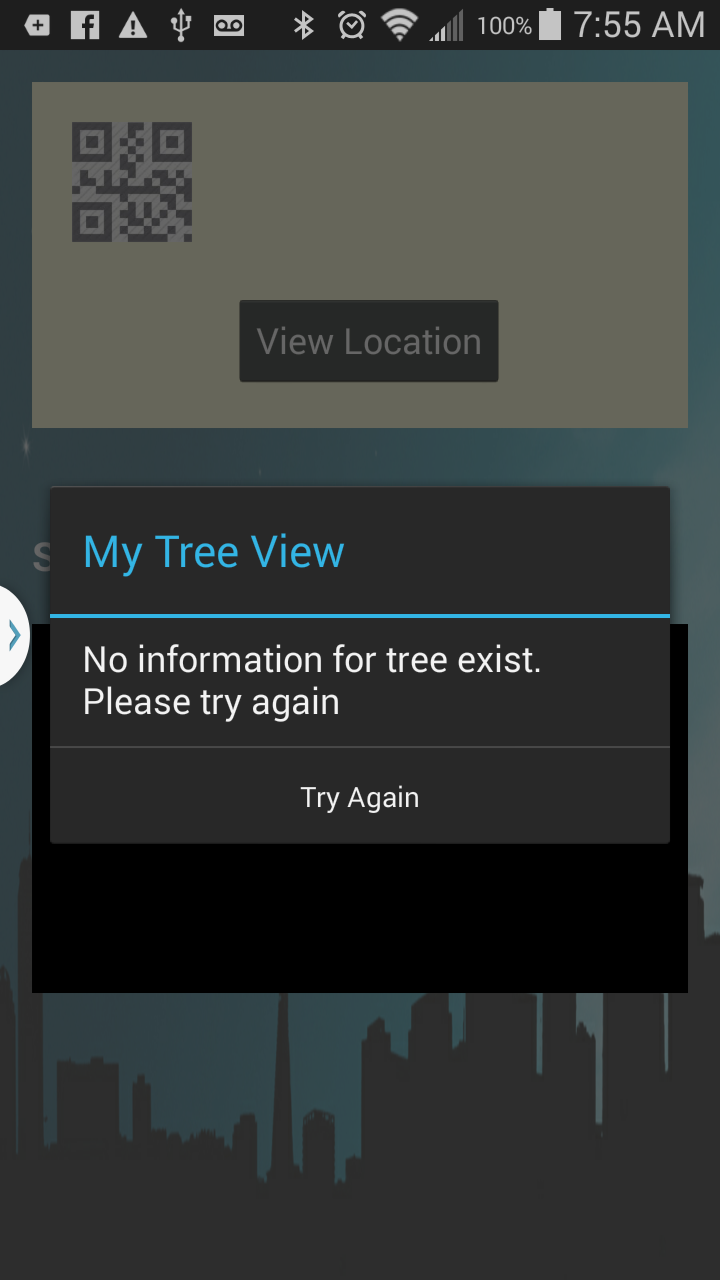
### Notable features

Some of notable features used for this app include Camera, map features, youtube embedded video, sharing feature. The code, in details, will be included in source code .zip file.

## User flows & Error Handling

We are making sure that this app is designed with “user-friendly” experience. Which mean, not only positive case need to be handled correctly, but also error cases and limitation of the app need to be notified upfront to user.

1. The users open the app and will see 6 icons (corresponding with scanning barcode, user interaction, sharing, map, camera and rating in app store feature.
2. When users click on “barcode” icon button, he/she will be directed to barcodeActivity, where user will able to scan “QRCode” from the webpage below to get information of a specific page. (Note: the webpage is developed on NodeJs with dynamic generated QRCode for future scale purposes).
3. If **QR code is invalid**, we will notice users and will direct them back to barcode activity to re-scan (see image below). If **QRCode valid**, we will direct user to interactActivity page, where user could see the tree’s information, such as addresses, name, description. User interact with virtual switch button and also be able to watch Youtube video for that tree. 



1. From the interactActivity page, users also are able to navigate to “map” feature to see the current tree’s location. There is a drop down (placepicker) in this page to help user search for specific nearby places (which are grouped in 1 category by google api). We set the zoomLevel to 10 for acceptable viewmode. User click on nearby icon, a dialog popup will appear, notice user about the place’s address.

Note: In order to maintain one address why navigate between InteractActivity and MapsActivity page, I use Android’s session concept. Before navigate away from InteractActivity page, we will store session value (which is current address place). In MapsActivity, we will read these value to display the current place. Code example as below:

|  |
| --- |
| /\*\* In IntentActivity.java, setting session \*\*/  *//Set up handler for view location button for specific tree*  Button viewLoc = (Button) findViewById(R.id.***tree\_view\_location***);  viewLoc.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View v) {  Intent launchActivity = **new** Intent(InteractActivity.**this**, MapsActivity.**class**);  launchActivity.putExtra(**"longitude"**, myLocation.getLongitude());  launchActivity.putExtra(**"latitude"**, myLocation.getLatitude());  launchActivity.putExtra(**"name"**, myLocation.getName());  launchActivity.putExtra(**"address"**, myLocation.getAddress());  startActivity(launchActivity);  }  }); |

|  |
| --- |
| /\*\* In MapsActivity, reading session \*\*/  Bundle extras = getIntent().getExtras();  Log.*i*(*TAG*, **"current location display called"**);  **if** (extras != **null**) {  **double** longitude = extras.getDouble(**"longitude"**);  **double** latitude = extras.getDouble(**"latitude"**);  String name = extras.getString(**"name"**);  String address = extras.getString(**"address"**);  *// Add a marker in chosen location*  **mLocation** = **new** LatLng(latitude, longitude);  Log.*i*(*TAG*, **mLocation**.toString());  **mGoogleMap**.addMarker(**new** MarkerOptions().position(**mLocation**).title(name).icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_AZURE***)));  **mGoogleMap**.moveCamera(CameraUpdateFactory.*newLatLngZoom*(**mLocation**, 10));  *//set my location*  **mGoogleMap**.getUiSettings().setMyLocationButtonEnabled(**true**);  TextView address\_text = (TextView) findViewById(R.id.***map\_address***);  address\_text.setText(**"My Place: "** + name + **"--"** + address);  } |

1. User can also take and share picture/record video that has been taken via social media on CameraActivity page. In this feature, I leverage the “Intent” method to display sharing social medias that are available on user’s phone. Snippet of code as follow

|  |
| --- |
| **...**  **shareButton**.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View view) {  BitmapDrawable bitmapDrawable = (BitmapDrawable)**viewImage**.getDrawable();  Bitmap bitmap = bitmapDrawable.getBitmap();  *// Save this bitmap to a file.*  File cache = getApplicationContext().getExternalCacheDir();  File sharefile = **new** File(cache, **"toshare.png"**);  **try** {  FileOutputStream out = **new** FileOutputStream(sharefile);  bitmap.compress(Bitmap.CompressFormat.***PNG***, 100, out);  out.flush();  out.close();  } **catch** (IOException e) {  Log.*e*(*TAG*, e.toString());  }  *// Now send it out to share*  Intent share = **new** Intent(android.content.Intent.***ACTION\_SEND***);  share.setType(**"image/\*"**);  share.putExtra(Intent.***EXTRA\_STREAM***, Uri.*parse*(**"file://"** + sharefile));  **try** {  startActivity(Intent.*createChooser*(share, **"Share photo"**));  } **catch** (Exception e) {  Log.*e*(*TAG*, e.toString());  }  }  }); |

* Notice that with camera and video, the current code base maintained the action in 2 different activities (CameraActivity & VideoActivity)

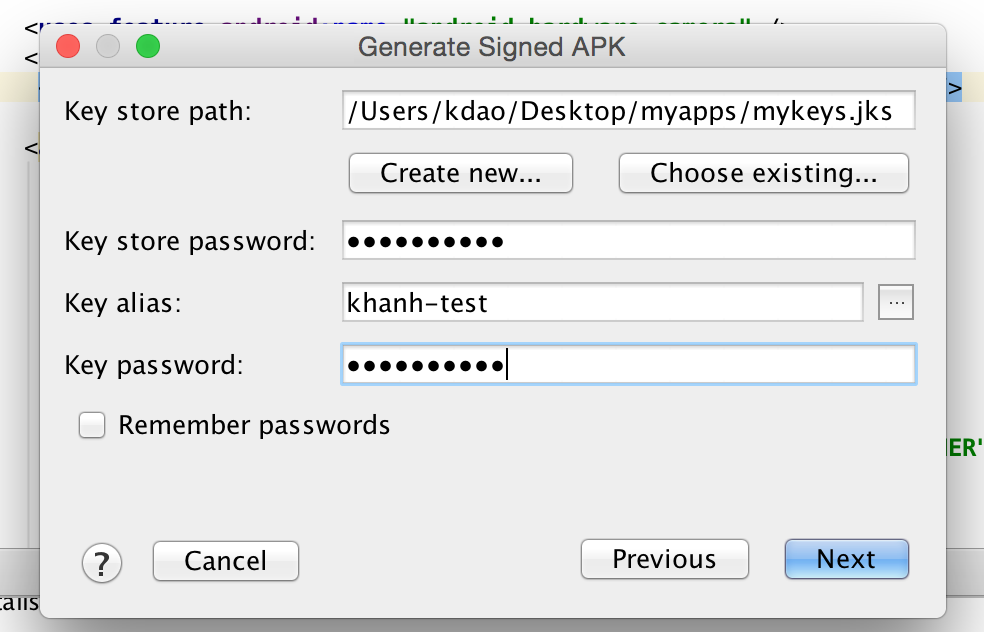


1. Finally, user able to use comment feature to comment on the app via app stores (this feature work with the fact that your app will be published to app store to make the app share URL exist) – (See above image for rating feature)

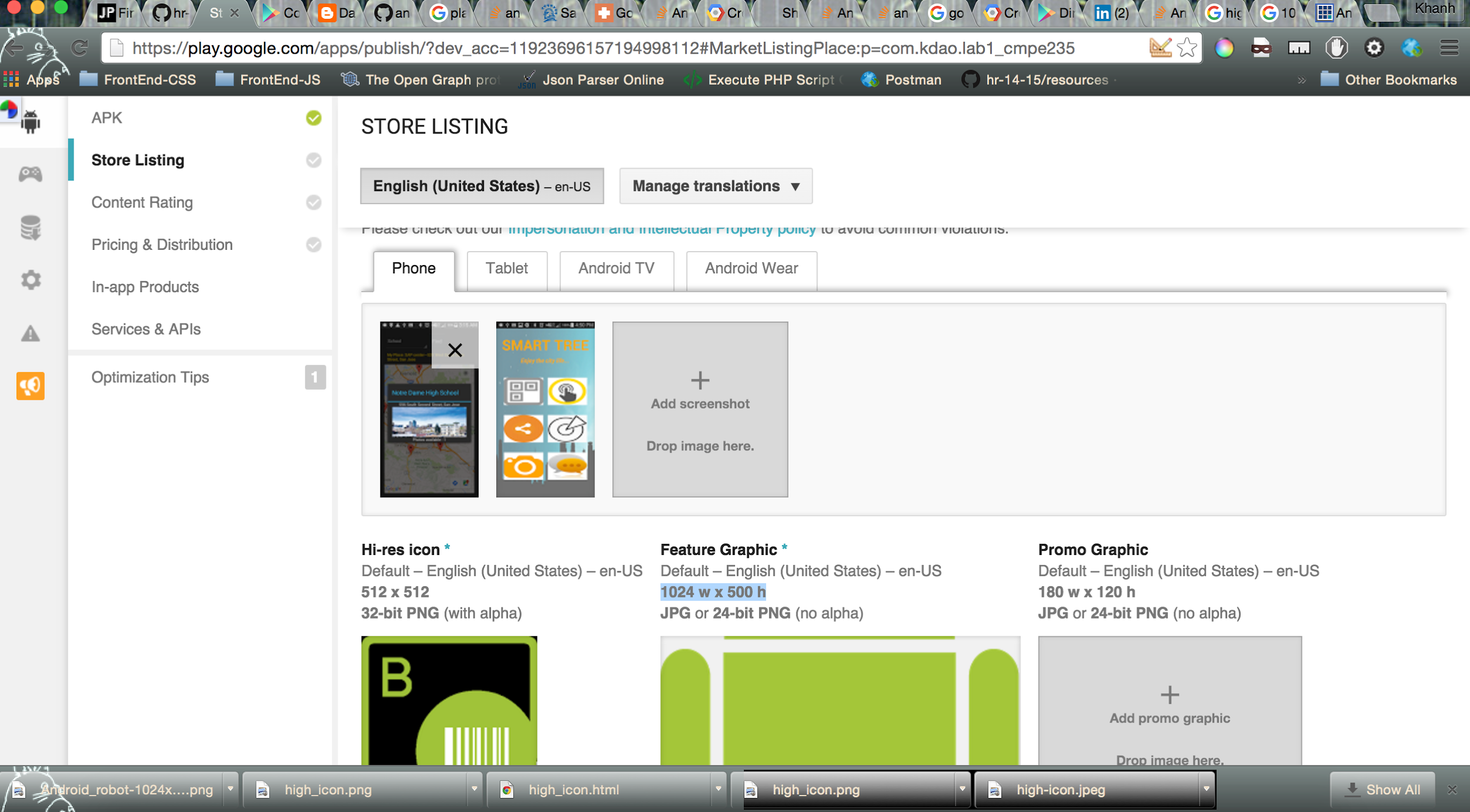
## Publish to apps store

In this lab exercise, I also learn how to publish an app to google app store (notice this process will take about 2-3 hours to get your app appear in the app stores)

1. Build .apk file for app to be published (In Android studio, navigate to /Build/Generate Signed APK). From there, we will need to input our credential, set up keystores password and upload .apk file as release version



1. Go to google app store, sign-up as developer and upload .apk file to publish. We will need to go through google’s standard app authentication process in order to publish the app (image below). For purpose of testing user comment action, I published this app to app stores as “free” and “low maturity” feature for now.



1. Finally, my app is published in google app store <https://play.google.com/store/apps/details?id=com.kdao.lab1_cmpe235>

## Lessons Learned

This lab exercise has help us to learn the whole life cycle of developing an app, from feature request, to UI design, to implement and testing, to deploy app to app stores. We’ve learned different layout feature provided by Android and how to leverage that for our design. We learn different event handler for android app (click event on button, ImageView, ListView, Toast, alert dialog). We also learn how to work with different feature of a smartphone, from camera/audio access to using google API for map or youtube display, to social sharing feature…

Not only are the logic and structures of codebase an important factors, the UI design and UX flow are also very important to keep your app consistent and coherent.

## TODO

Driving instruction: I’ve not finished this feature, however the code base contains part of it. I will use third party package to draw line between 2 chosen dots and display info -- the package is currently included in build.gradle. (compile **'com.github.jd-alexander:library:1.1.0')**

Referenced from github repo: <https://github.com/jd-alexander/Google-Directions-Android>

## Conclusion

The Android Studio platform and its development tool allows for very rapid application development. I also found the google APIs are documented pretty simple and easy to understand, which helps a new person (who has Java background, but not mobile application background, to be on board pretty quickly and effectively). The UI tools also seems very powerful for design, even though I found it’s more effective to do text editing for the layout when you already get use to the syntax.

## References

<http://wptrafficanalyzer.in/blog/showing-nearby-places-with-photos-at-any-location-in-google-maps-android-api-v2/>

<https://github.com/codepath/android_guides/wiki/Sharing-Content-with-Intents>

<http://www.tutorialspoint.com/android/android_camera.htm>

<http://www.sitepoint.com/using-the-youtube-api-to-embed-video-in-an-android-app/>

<http://stackoverflow.com/questions/14514579/how-to-implement-rate-it-feature-in-android-app>

<https://examples.javacodegeeks.com/android/core/android-video-capture-example/>

<http://www.techotopia.com/index.php/Video_Recording_and_Image_Capture_on_Android_using_Camera_Intents>