Final Report: Scavenger Hunt

Carol Huang and Yu Hu

1. **Application Description**

Problem Statement

We were inspired by the Giorgio App and so we wanted to create a project using image recognition with the Android studio. Since there was a lot of free code online, we wanted to create something that was more interesting and included more work and at the same time we wanted to create a game that was user-friendly and fun like the Giorgio App.

Solution: Our solution came from our professor, Peter, and we decided to create a scavenger hunt app that would allow the user to search for objects with their android phones and then take pictures of these objects and the app would use the Google Cloud Vision Api to recognize the objects and cross them off from the list. We also uses Google Map API for the user to see where they are and and to record where they found their objects.

2. **Block diagram**

The Java classes that bridge front end and back end are:

PhotoRecognitionActivity.java: extends Activities. Contains photo recognition

MapsActivity.java: extends Activity. Contains location tracking.

startPage.java: extends Activity. Connect the welcome page to the main page.

listWriter.java: extends Activity. Serve as an XML writer that dynamically add Scavenger Object from the list to the xml main page.

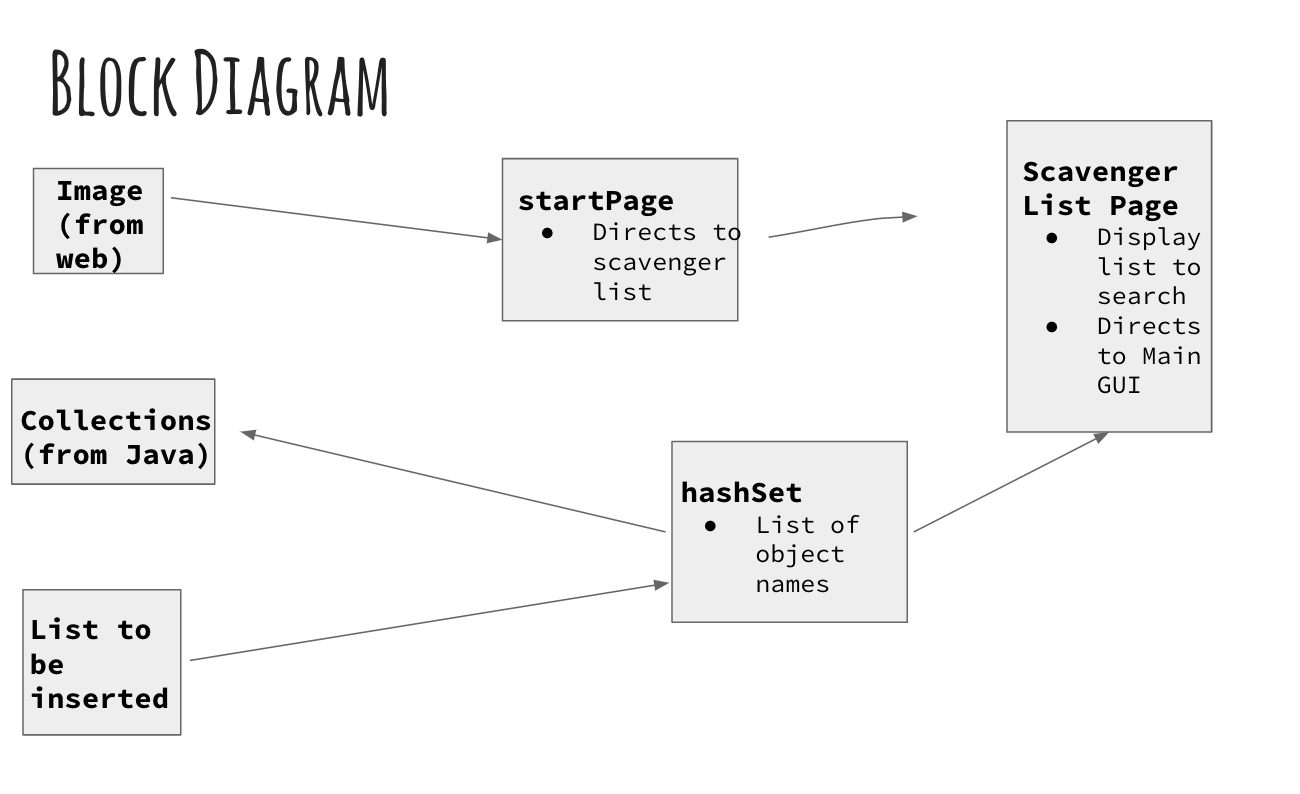
PermissionUtils.java

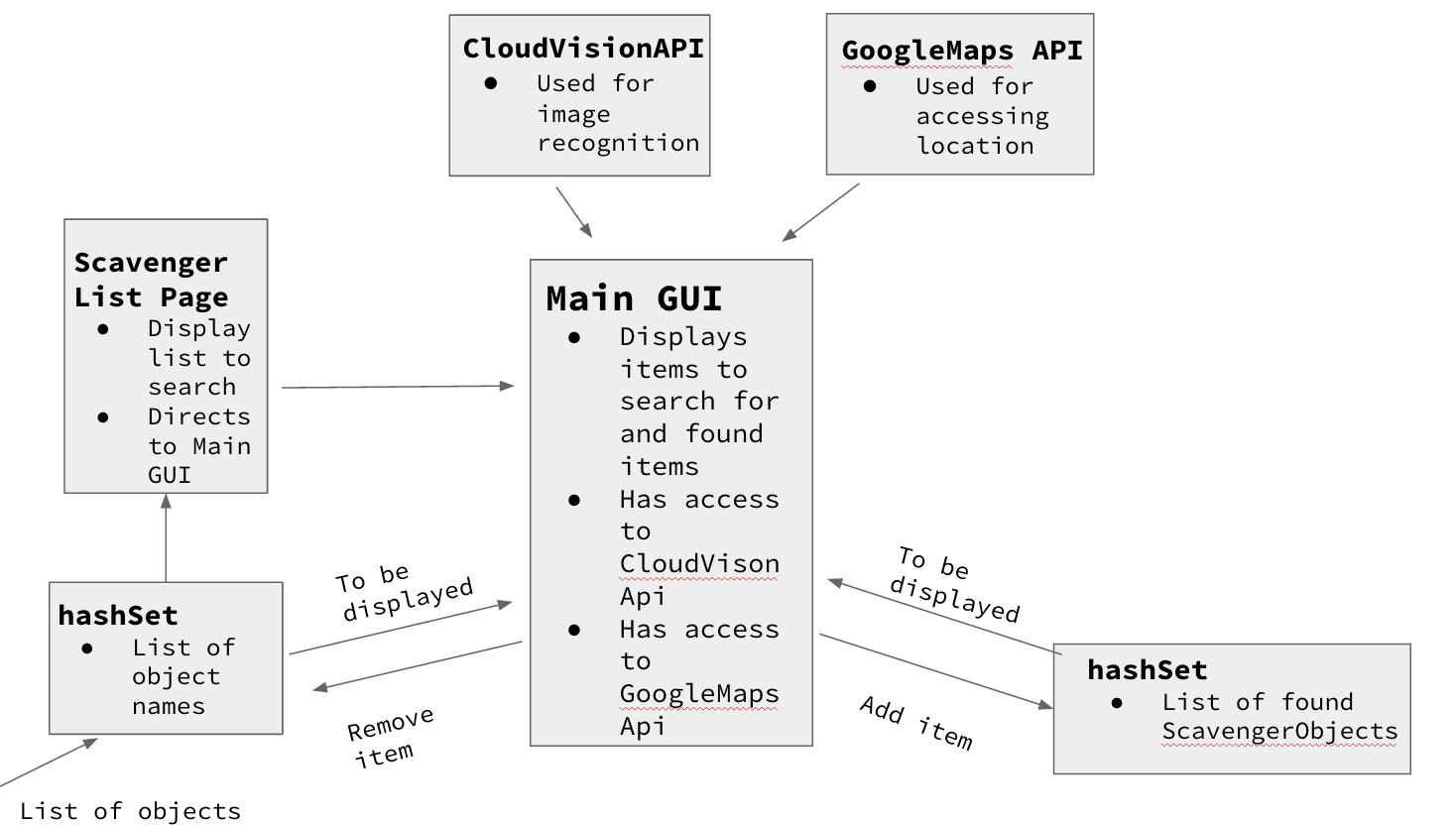
The back end java classes for data storage are:

ScavengeList.java: Use HashSet to store data: HashSet<ScavengerObject>. It contains two hashSet, one for searching objects, one for found objects.

ScavengeObject.java: a ScavengerObject has two variables: name and location.

ScavengeObjects are stored in HashSet. They are displayed on the front end by listWriter.java which is an XML writer class. Users press buttons to starts either the map activity or photo recognition activity. If the map activity starts, the current user location is stored as a String. If the photo recognition activity starts, the object identity is stored as a String and a search method of the ScavengeList class is called to check if the user found the correct objects. If search() returns true, the object is removed from the searching object HashSet and is added into the found objects HashSet. Every time the hashSet is renewed, the XML file on the front end is rewritten by listWriter.java.





Resources (Tutorials and documentations)

Basic Android

<https://github.com/YuHu0621/Dining2Go>

<https://developer.android.com/training/basics/firstapp/starting-activity.html>

<http://stackoverflow.com/questions/9704898/how-to-set-my-activity-as-main-activity-in-android>

<https://developer.android.com/training/basics/activity-lifecycle/starting.html>

<https://developer.android.com/training/permissions/requesting.html>

Cloud vision API source code:

<https://github.com/GoogleCloudPlatform/cloud-vision/blob/master/android/README.md>

Google Map API source code:

<https://developer.android.com/training/location/display-address.html>

<http://blog.teamtreehouse.com/beginners-guide-location-android>

Google Map API resources:

<https://developer.android.com/training/location/retrieve-current.html>

Camera access:

<https://developer.android.com/guide/topics/media/camera.html>

XML Parsing:

<https://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html>

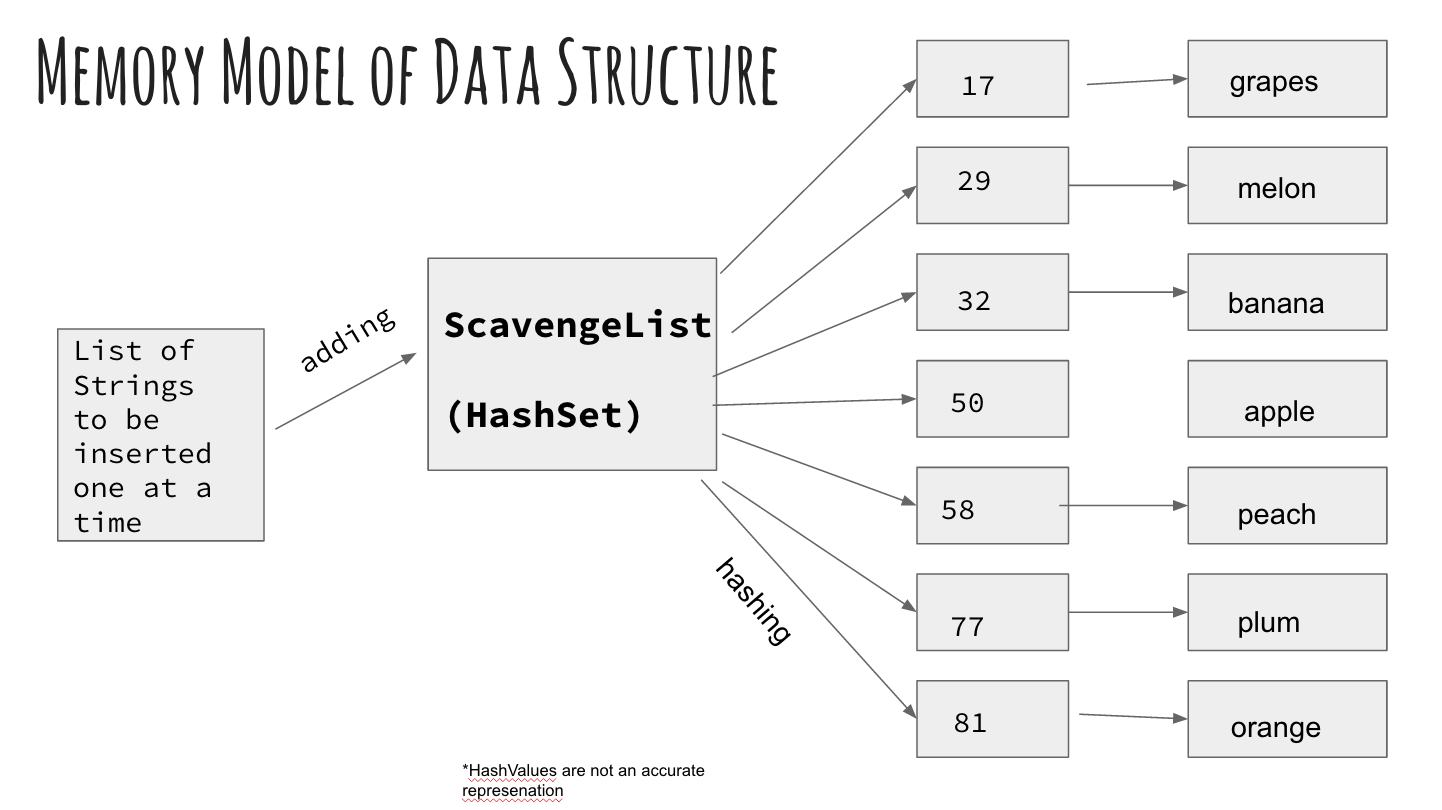
<https://github.com/foamguy/stacksites>

JSoup:

<https://jsoup.org/apidocs/>

2. **Data structure**

HashSet is used to store the data in this app. Because the search method will be called more frequently than the add method in this app, a data structure that allows fast search was more needed than to add it on. HashSet stores the object by hashing it and it spreads the data more evenly across the set. So the run times for search, add and remove are all constant.



3. **Application Demonstration**

Start page: users click Play



Scavenge object list page: display all the objects. Users view the object list and clicks GO.

MainActivity page: User clicks SNAP IT to snap the photo and clicks CHECK MAP to check current location. User can also clicks CHECK LIST to see the renewed object list (whether the found object is removed from the searching list).

Map page: Users can view their current location.

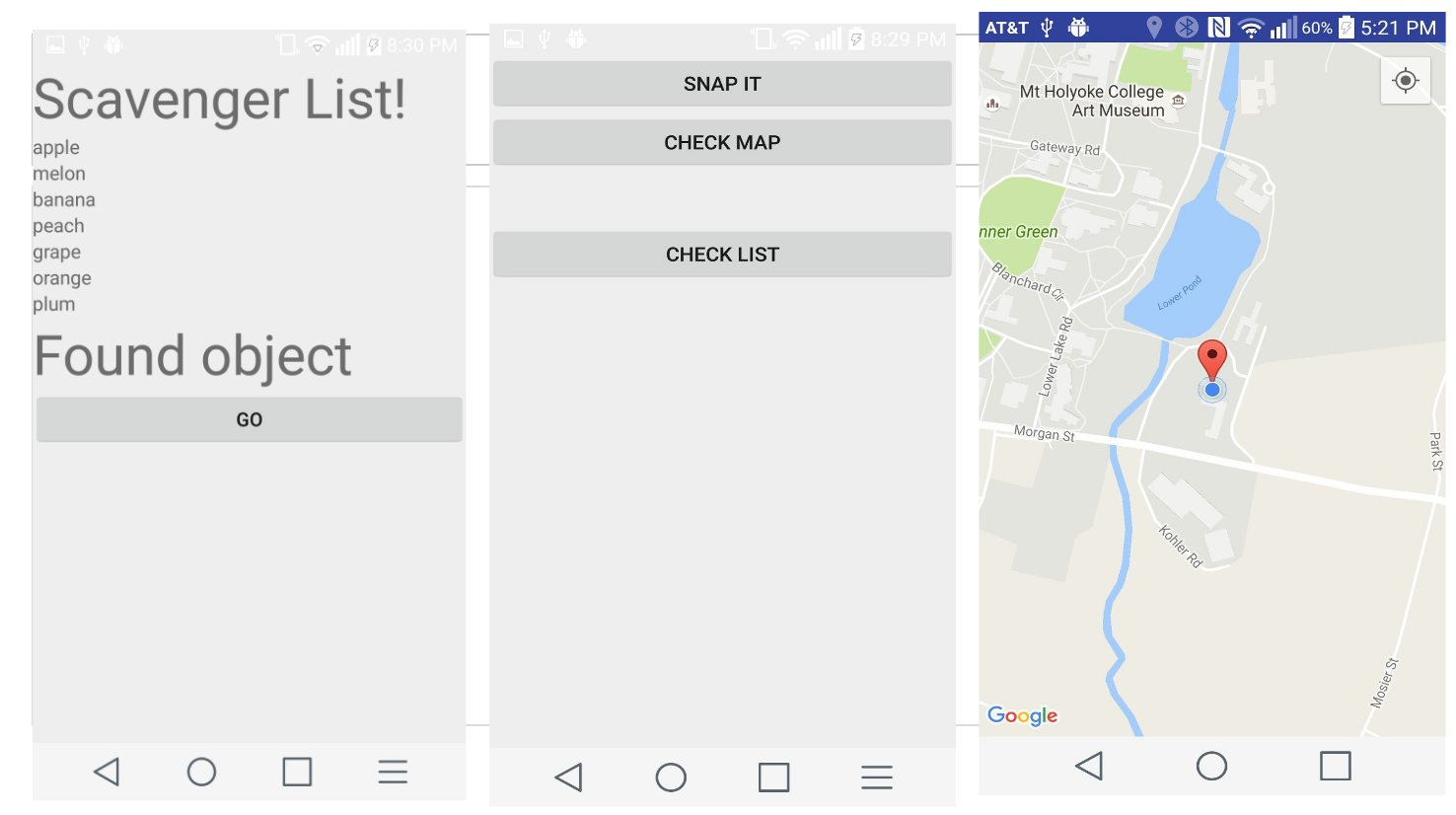
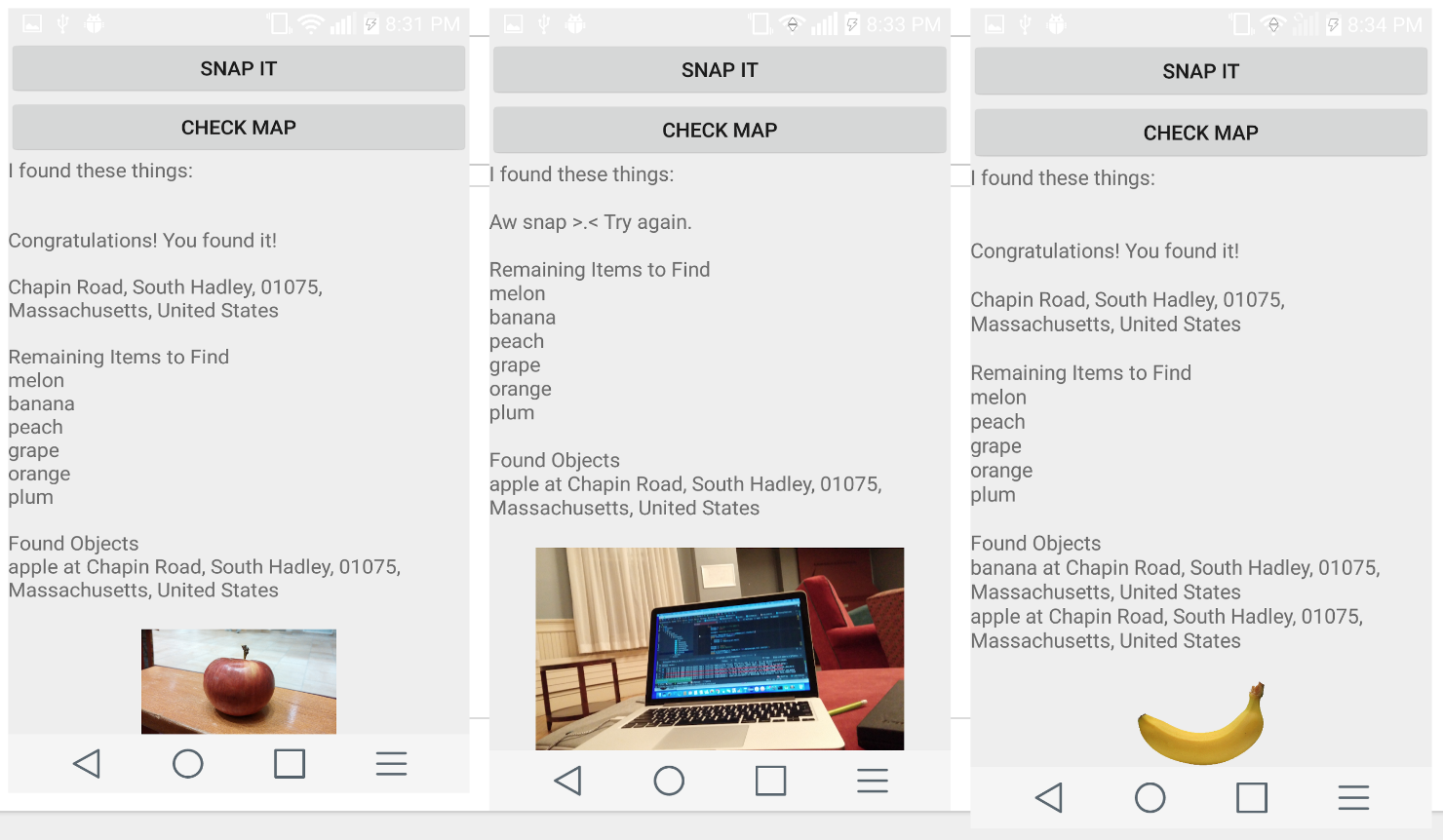


Photo recognition page: The snapped photo is displayed. If the snapped object matches with one of the items in the scavenge object list, the screen displays “Congratulation, you found it”. Otherwise, it says “Aw snap!”. The remaining list of items and found objects are also displayed.



This application allows the user to use the Cloud Vision API to play scavenger hunt. Our initial goal of photo recognition and location tracking was reached.

4. **Reflection on the Final Project**

We learned a lot while we were doing this project. We already had some experience with parsing data from a website and we knew a little about Android Studio so doing this project allowed us to expand on our knowledge on these things. At the same time, we also had to learn how to adapt code we found elsewhere and fit it into our project in a way that would work with our conception of the idea and figure out how these outside sources worked. While there are still bugs in our program and we couldn’t figure out everything we wanted to do, we think that we did end up creating something that solved our problem statement above and completed the basic outline of what we wanted.

The parts that were especially difficult was parsing the data and it was mainly because we had to figure out how Android Studio worked when it came to parsing and we looked at resources for how to solve our parsing issue and we couldn’t figure out what it was that we were doing wrong. The other thing was using the external APis because sometimes we had to manipulate our build gradle and our manifest file because of all the errors that could pop up when using them. In that way, it was also interesting to get into Android Studio and figure out it’s quirks and despite how frustrating it became to get errors over things that would work fine a while ago, it was also a lot of fun when we figured out the errors. We also could have managed our time better because we both had things to do so we would rely on each other to finish parts of the project when it would have been better if we were both working on it so we could figure it out more quickly than relying on the other person. But despite all this, it was a fun project and we definitely learned a lot while doing this project and we had fun.

5. **Possible Future Directions**

1. Currently, the scavenger object list items are entered manually. In the future, we want to be able to use xmlPullParser to obtain data from the web put that into our hashSet of ScavengerObjects.
2. The google cloud vision API is not very accurate in recognizing images. We want to allow the users to correct inaccurate options for the image recognition.
3. We want the app to allow the user to connect with others to create a team to complete scavenger hunt.
4. We want to improve the UI design so it could be more aesthetically pleasing and more user-friendly.
5. We want to allow the user to save their progress in the app so that when they close out of it, the game would not restart but instead store their current information for the user to go back to.

**Appendix**

**Updated schedule and milestones**

Milestone

* Obtain final project approval
* Learn how to use Google Cloud Vision API
* Get one picture working
* Create skeleton file
* Create GUI for the app
* Learn Android Studio
* Figure out how to get the camera to work for the Android Studio
* Create a hashTable to store the scavenger objects
* Put the front-end and back-end together
* Prepare the final project presentation
* Give final project presentation
* Complete the final report

Reach goal

* Communication among devices

Calendar

* Week 1: (Nov 20-26) Thanksgiving break
  + Learn the Google Cloud Vision API
  + Learn how to use Android Studio
  + Make skeleton files
* Week 2: (Nov 27-Dec 3)
  + Learn how to access camera in Android Studio and begin working on front end
  + Create back end of assignment and connect to Cloud Vision API
    - This includes creating a hash table of items to search for
  + Start connecting the front end and back end of assignment
  + Test with one image
* Week 3: (Dec 4-10)
  + Work out bugs in program
  + Test with multiple images
  + Between Dec 6-10
    - Develop project presentation
    - Give project presentation
    - Begin working on final report
* Week 4: (Dec 11-13)
  + Finish final report and turn it in

**PoC Update**

PoC Goals & Progress

* Use CloudVision API for image recognition
* Create a hashSet for scavengerObjectsList
* Identify the user’s location when taking the picture
* Went through the CloudVisionAPI for Android Google tutorial
* Got the image recognition app to work on our android studio
* Found a html webpage on list of food
* Made a JAVA class HTMLReader using JSoup library
* Read a webpage on list of food and add it to a HashSet in the Java class ScavengeList
* Made a Java class XMLWriter and put all the objects into the layout xml. In other words, users can see the list of food now.
* Made the other layout xml file and java class that connect them
* Went through the tutorial for Google play service location.
* Got the location identification app on our android studio

PoC Weekly Time Accounting

* Went through the CloudVision API tutorial and got the image recognition app to work: 1 hour, **completed**
* Read HTML file using JSoup: 2 hours, **completed**
* Went through Android studio tutorial: 1 hour, **completed**
* ScavengeObject and ScavengeList class with HashSet: ½ hour, **completed**
* Made an XMLWriter class: 2 hours, **completed**
* Going through tutorial for Google play service for location and creating the program: 6-7 hours, **completed**
* Made layout xml file: ½ hour, **completed**
* Use xml parser: 3 hours, **in-progress**

PoC Hard & Interesting

* When we were importing jSoup into the android studio we kept running into this error “Compile error: Java.io.FileNotFoundException” which was really frustrating because we were following tutorials on how to import it and we saw that the jSoup was part of the application but the IDE was not recognizing it.
* Then we realize that in order for android to access internet, the xml parser need to extend asykTask. Then we found out that there’s a XmlParser for Java. Though we followed the tutorial, we still have trouble with the XmlParser. 
* The other thing was learning how to use the Android Studio and get it to work because we would run into sync errors which would cost us time when we were testing things and we would have to restart the application or look around the web to see why it kept happening so it became pretty frustrating to work with Android studio.
* It was interesting and cool to see how the CloudVision API works and put it with the application and we had a lot of fun testing it out.
* Figuring out location for the app was really hard to get through because we followed [this tutorial](http://blog.teamtreehouse.com/beginners-guide-location-android) which was informative and explained what they were doing but there were steps that were confusing and the writer didn’t elaborate on them in the tutorial so we had to look around the web to see what the inconsistencies were. The other thing was because the location required permissions, we also had to look through the Android Studio developer’s guides to figure out how to input it in the program which took a while but ultimately it was really fun to work through it. One other thing that was mostly odd was that even when we got the program to run on the emulator, the coordinates were always in California and we didn’t understand why (but they showed the correct coordinates on the phone) and we looked around and couldn’t find an answer so that was pretty frustrating.
* The other difficult part was exporting the app because we could run them fine on the emulators but we wanted to see how they would work on an actual phone. It turns out that exporting the app was also difficult because there were extra steps we had to take to export successfully and even when we did export it, sometimes the app wouldn’t run and we weren’t sure why it was so we kept searching for reasons and watching videos of people exporting their apps and trying that out. Ultimately we figured out to use USB debugging and use the phone like we would use the emulator and that way we could install the app in the phone.

PoC Next Week’s Plan

* Finish parsing
* Put together the location with the image recognition
* Prepare and complete presentation

**Proposal**

*Project Idea:* We would like to create an scavenger hunt app that would present a list of objects for the user to search for whether individually or in groups and the user would be able to take pictures using the camera on the phone. The program will then recognize what the object is and then check off the list if the image matches with one of the objects in the list. When all the objects are checked off, the user (and their group) “wins” the game and the app restarts.

*Targeted user*: Our target audience is anyone who owns a smart device though mainly for people who own android phones. Scavenger hunt is a very useful team-bonding activity. Our app is designed to be accessible to everyone and for the enjoyment of everyone because the interface should be straightforward so the users will understand what to do.

*Tools*: There are a couple of things that we need to prepare for the app. We’re planning on using the Cloud Vision API for accessing a wide arrange of images and for the image recognition aspect of the app. Since we want to create an app, we’re planning on using Android studio for the front end part of the app. Within the front end part of the app, we would need to create a main screen for the application as well as gain access to the device’s camera in order to take pictures and store it for later access. Our main programming language will be in Java.

*Data structure*: We would like to use a hashTable to store the targeted objects. Function include insert, delete, and search.

*Reach Goal*: The app would allow communication among devices so people can work as a team.

Milestone

* Obtain final project approval
* Learn how to use Google Cloud Vision API
* Get one picture working
* Create skeleton file
* Create GUI for the app
* Learn Android Studio
* Figure out how to get the camera to work for the Android Studio
* Create a hashTable to store the scavenger objects
* Put the front-end and back-end together
* Prepare the final project presentation
* Give final project presentation
* Complete the final report

Reach goal

* Communication among devices

Calendar

* Week 1: (Nov 20-26) Thanksgiving break
  + Learn the Google Cloud Vision API
  + Learn how to use Android Studio
  + Make skeleton files
* Week 2: (Nov 27-Dec 3)
  + Learn how to access camera in Android Studio and begin working on front end
  + Create back end of assignment and connect to Cloud Vision API
    - This includes creating a hash table of items to search for
  + Start connecting the front end and back end of assignment
  + Test with one image
* Week 3: (Dec 4-10)
  + Work out bugs in program
  + Test with multiple images
  + Between Dec 6-10
    - Develop project presentation
    - Give project presentation
    - Begin working on final report
* Week 4: (Dec 11-13)
  + Finish final report and turn it in