Cal Poly Application Parking Feature Integration

Ryan Lin
Advisor: Bridget Benson
Cal Poly State University San Luis Obispo
Electrical Engineering Program
6/10/2015

Table of Contents

List of Figures

List of Tables

A. Abstract

The Cal Poly mobile app provides students, faculty, staff, and visitors of Cal Poly with information pertaining to the campus (such as information about admissions, campus life, campus news, etc.) This project extends the functionality of the Cal Poly app to provide information regarding campus transportation. In particular, this project provides 1. a live twitter feed with announcements about events that may affect the roads and parking spaces on campus 2. an ‘alternative transportation’ points of interest map illustrating the locations of zipcars and electric vehicles charging stations on campus and 3. an easy to use interface to reserve an available zipcar 4. parking lot locations specified in the existing map section of the app, along with the number of each parking space type. These additions to the Cal Poly app may help users save fuel, money, and time.

B. Introduction

When driving to the Cal Poly campus, people may find themselves in situations that require unnecessary effort when it comes to parking. Drivers may find themselves spending half an hour trying to find parking due to events such as open house or Oktoberfest. Big events on campus require roadblocks and reserved parking for special guests. Parking can also be hard to find, especially in the mornings, on a daily basis as students, faculty, and staff need to park on campus in order to get to their classes or to their jobs. As a result, it has become necessary to provide better communication to the people who commute on campus.

The primary focus of this project is to make transportation on campus easier for students, faculty, staff, and other guests by saving them time, fuel, and money. This has been done by integrating a set of new features requested by the Cal Poly Police Department into the existing Cal Poly App.

The feature set that was proposed was inspired by the transportation features of other campuses' mobile applications. Other campuses have applications that may cover a similar feature set. For example, Florida State University has a wide variety of applications, one specifically for transportation. Although Cal Poly currently does not have the technology to specify live data on the number of currently *available* parking spots, this project specifies the capacity of each parking lot in terms of total number of spaces, and specifies the number of spaces per permit type.

The new features include:

- Points of Interest allowing the user to display parking lots on the Cal Poly map, as well as each parking lots' capacity, categorized by type (staff, general, sponsored guest, etc.)
- Points of Interest allowing the user to display Zipcar locations and charging points for electric vehicles
- An easy-to-use interface for registering for Zipcar and for reserving a Zipcar
- A live feed from the Cal Poly police department with status updates such as parking lot closures, road closures, increased areas of traffic, and on-campus-events which may affect parking or navigation

C. Requirements

The specifications for this project were given by the Cal Poly police with some advising given by the mobile development team from Cal Poly ITS.

a. Specifications

1. Developed for Android devices only (eventually want iOS version of features, but outside the scope of this senior project)
2. Must have Announcements, Points of Interest, and Zipcar page.
3. Live data (parking lot and traffic status) must be updatable by the Cal Poly police via methods that are accessible by them.
4. Data must be hosted on a server, not in the app
5. Point of Interest must be integrated into the existing map in the Cal Poly application.
6. Zipcar page will either have a link to the Zipcar website or will allow users to directly register and reserve a car directly from the application.
7. Parking lot capacity may be displayed in a static manner (number of spots of each type), although dynamically displaying number of open spots would be the eventual goal (outside scope of senior project)
8. Development cost: $0 (not including the cost of effort)

D. Design

The following figures show the overall system of the features integrated into the application. Figure 1 shows a block diagram which displays that the POIs feature is housed in the application, while the announcements get the data from the UPD Twitter feed and the Zipcar feature utilizes the mobile website. Figure 2 is a signal flow diagram of the features of the application which shows where the user will be taken after pressing specific buttons. Figure 3 shows where the features could be accessed in the menu of the official Cal Poly application. The Points of Interest will be located in the Map under the Information section. The Announcements will be located in the News under the Campus Life section. The Zipcar will be located under the Services section.

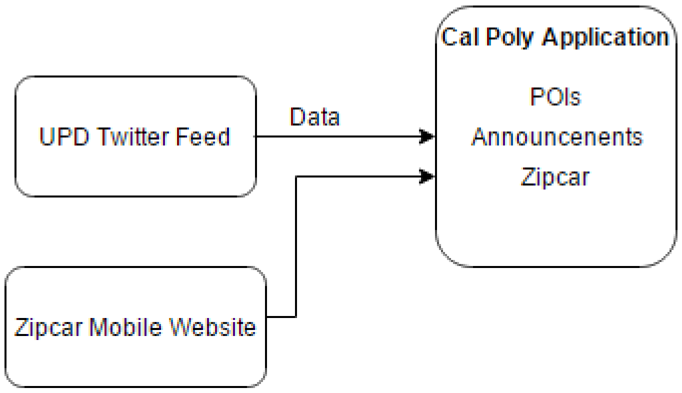


Figure 1: Block Diagram of Entire System

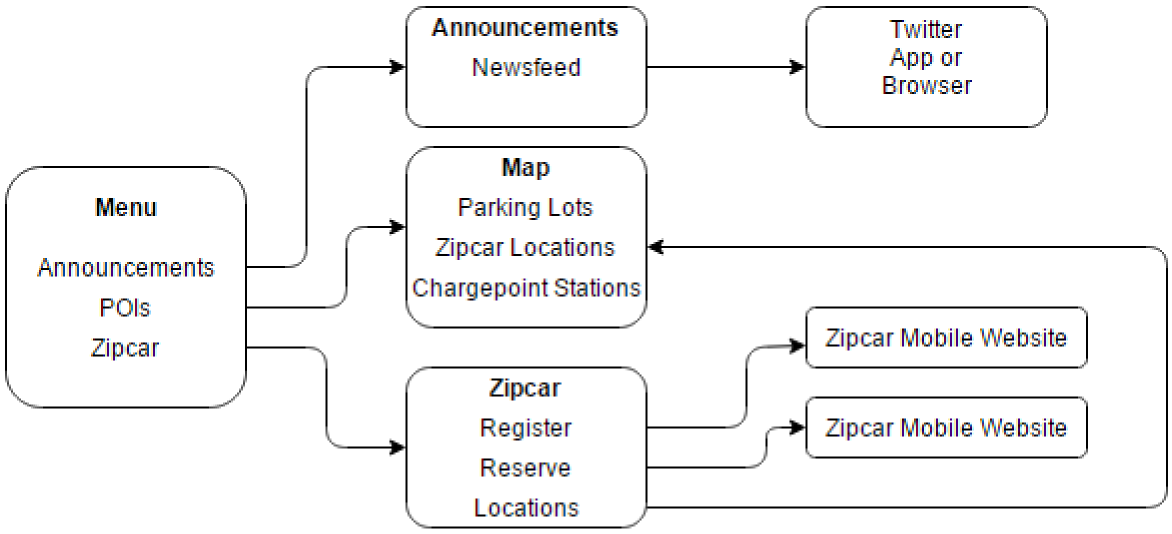


Figure 2: Signal Flow Diagram of Application

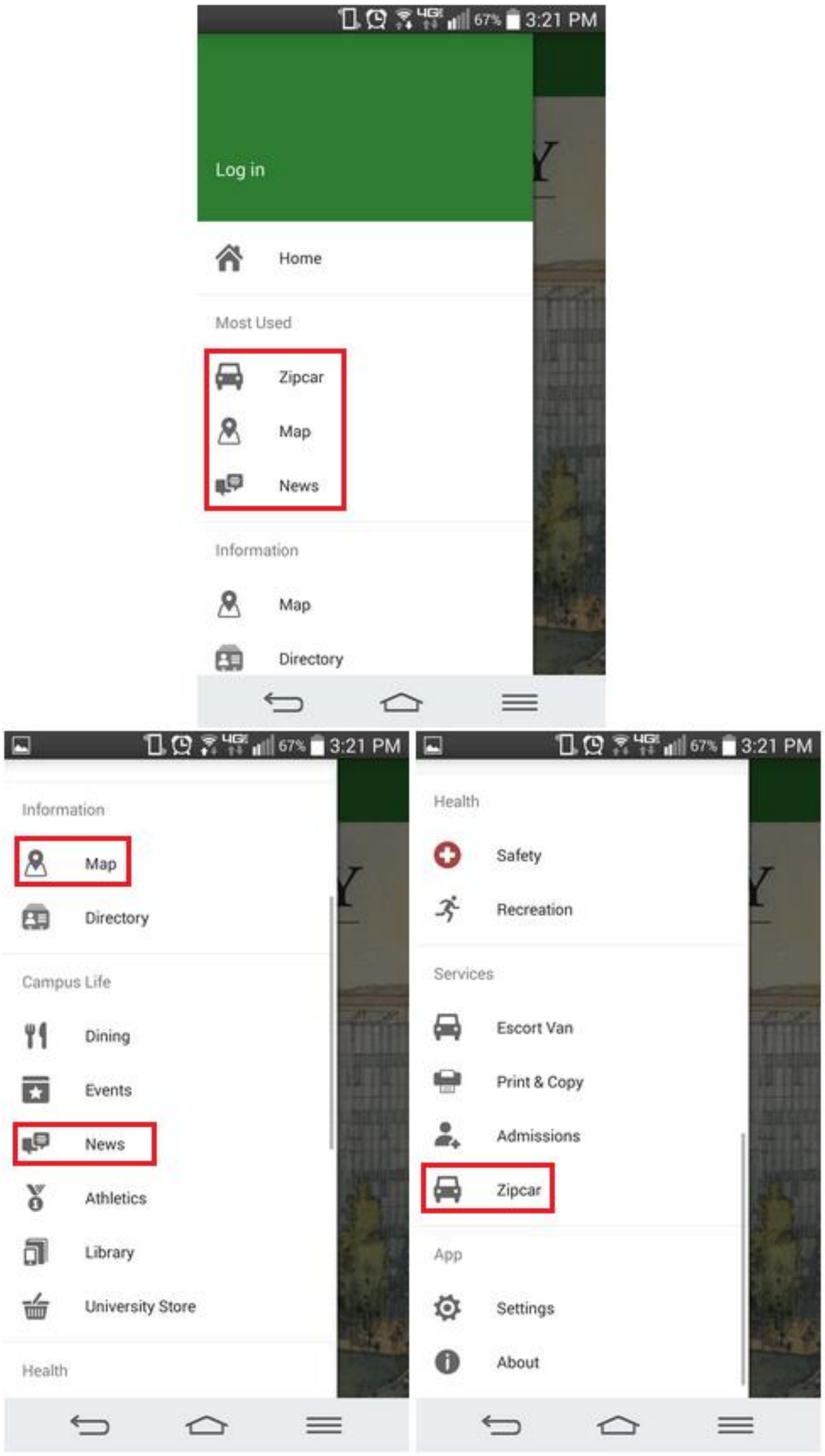


Figure 3: Location of Features in the Menu

a. Announcements

Figure 4 shows the block diagram of the announcements feature from the menu until the announcements page. After the user presses the News button, they are brought to the page with all of the feeds relevant to Cal Poly (Figure 6). Pressing the Cal Poly Police button will bring them to the announcements page of the UPD shown in Figure 7. Pressing on any one of the announcements will bring the user directly to the tweet either using the official Twitter mobile application or the default browser set to the user's phone.

There are several classes that achieve this feature which can be seen in Figure 5: TwitterFeedView, NewsFeedView, and NewsReader. The NewsReader and NewsFeedView classes were already in place at the beginning of implementation. The NewsReader class parses through an XML file hosted on a server, grabbing other XML files hosted from other servers. The NewsFeedView parses through these XML files and displays the announcements on the application. The TwitterFeedView class was treated differently using a twitter account to handle the feed. The NewsReader class was changed to handle the data coming in from the twitter account with the use of the TwitterFeedView. The TwitterFeedView class communicates with the Twitter API and authenticates the API keys, which allows the user to access the twitter feed.

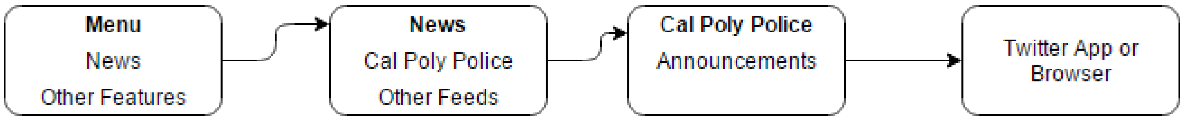


Figure 4: Block Diagram of the Announcements Feature

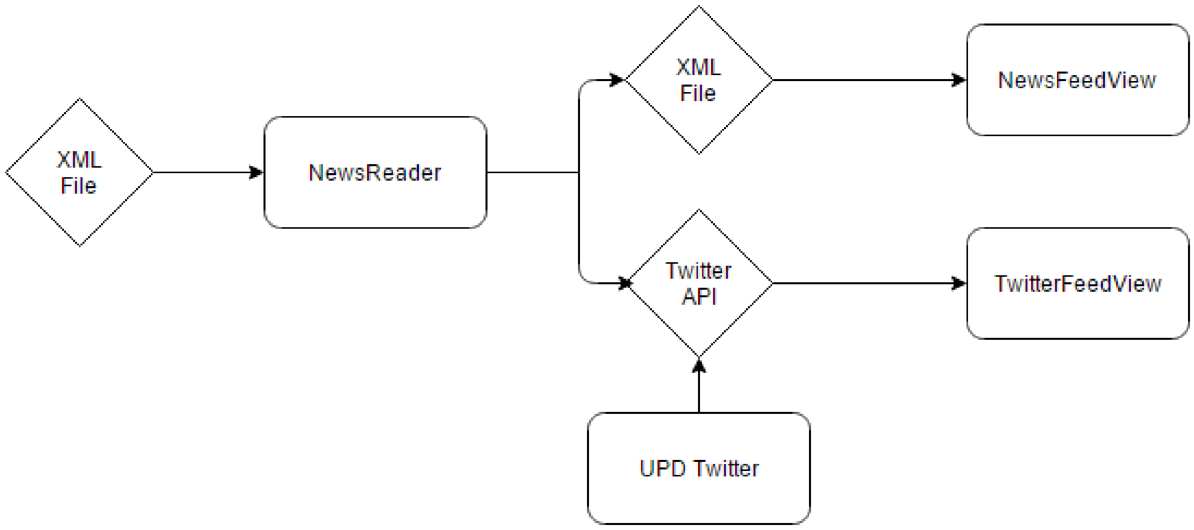


Figure 5: Software Flow Diagram of the Announcements Feature

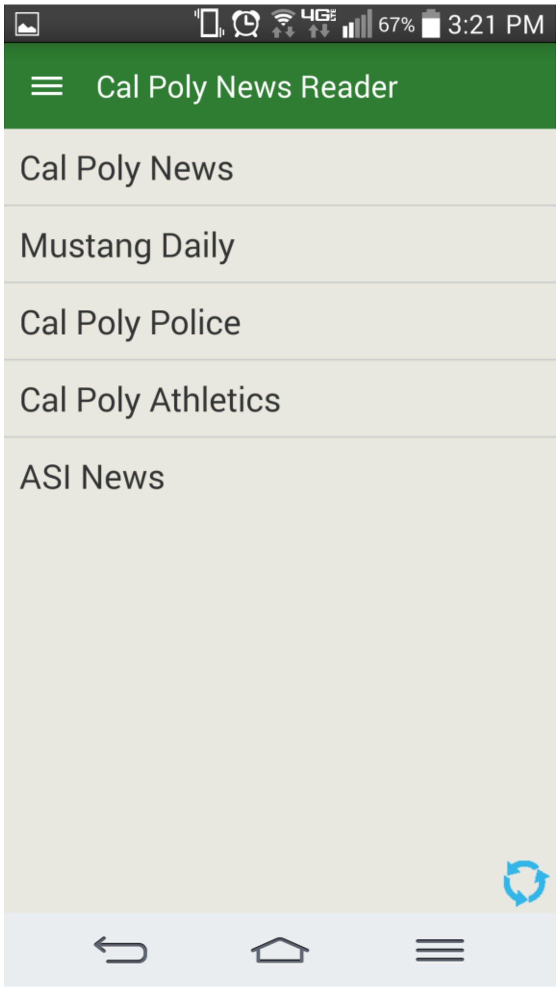
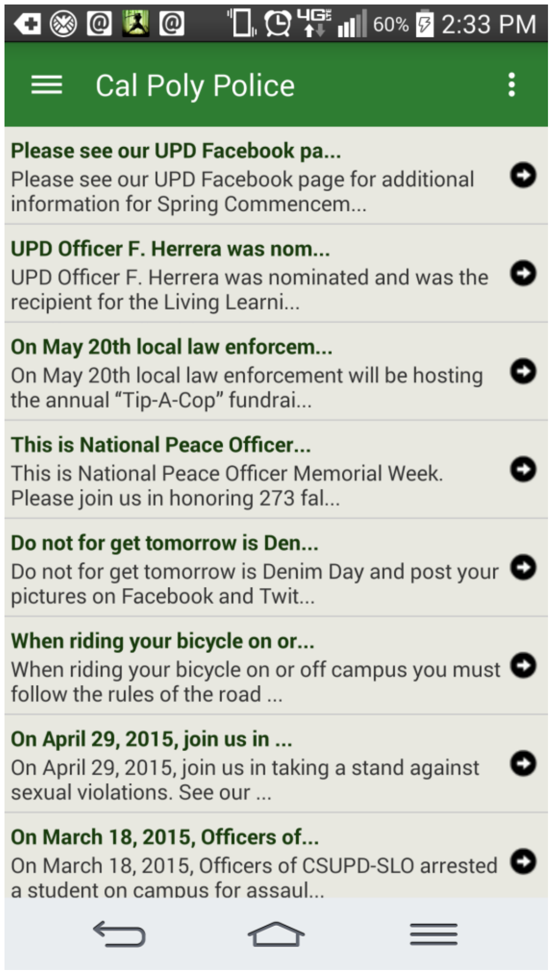


Figure 6: Page with All Feeds Relevant to Cal Poly



Please see our UPD Facebook pa...
Please see our UPD Facebook page for additional information for Spring Commencem...

UPD Officer F. Herrera was nom...
UPD Officer F. Herrera was nominated and was the recipient for the Living Learni...

On May 20th local law enforcem...
On May 20th local law enforcement will be hosting the annual "Tip-A-Cop" fundrai...

This is National Peace Officer...
This is National Peace Officer Memorial Week. Please join us in honoring 273 fal...

Do not for get tomorrow is Den...
Do not for get tomorrow is Denim Day and post your pictures on Facebook and Twit...

When riding your bicycle on or...
When riding your bicycle on or off campus you must follow the rules of the road ...

On April 29, 2015, join us in ...
On April 29, 2015, join us in taking a stand against sexual violations. See our ...

On March 18, 2015, Officers of...
On March 18, 2015, Officers of CSUPD-SLO arrested a student on campus for assaul...

Figure 7: Announcements page of the UPD's twitter feed

b. Points of Interest

Figure 8 is a block diagram which shows the user's route to the points of interest. The software flow diagram shown in Figure 9 shows the different classes and methods involved. The points of interest feature also included several classes: CalPolyMap and MarkPOIOnMap. The information for each point was entered into an XML file and read into the two classes. By using the google maps API, the map could be accessed. The parking lots were treated so that they could be searched for and as points of interests. There are several main functions that allow the points of interest to be displayed on the map. The configureZipcar, configureParkingLots, and configureChargePoints methods, found in the MarkPOIOnMap class, read in arrays of strings and integers with the name, subtext, and coordinates of each point of interest. If the user chooses to display one of the types of POI, the onCreate method in CalPolyMap class determines that. The reprintMap method in the MarkPOIOnMap class puts the actual marker on the map.

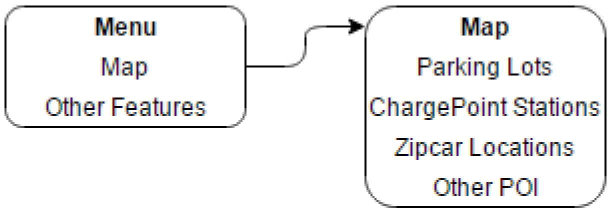


Figure 8: Block Diagram of the Points of Interest Feature

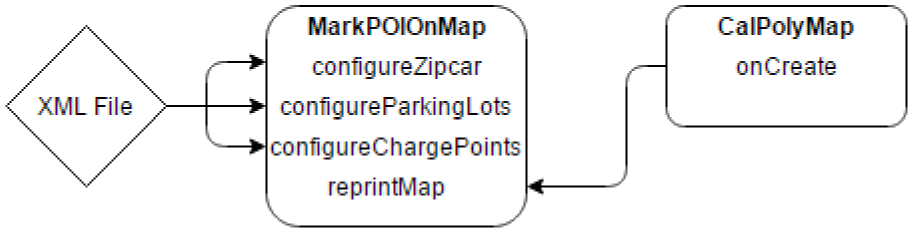
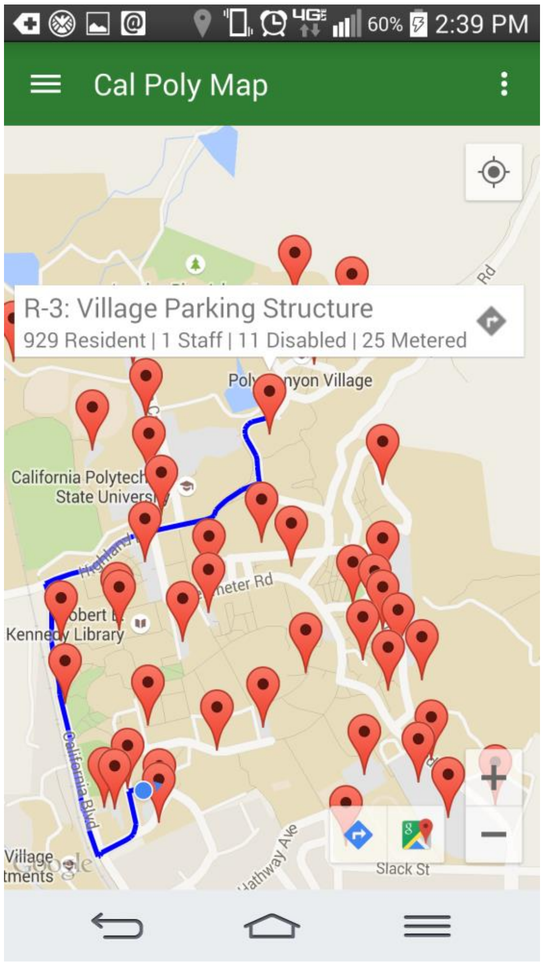


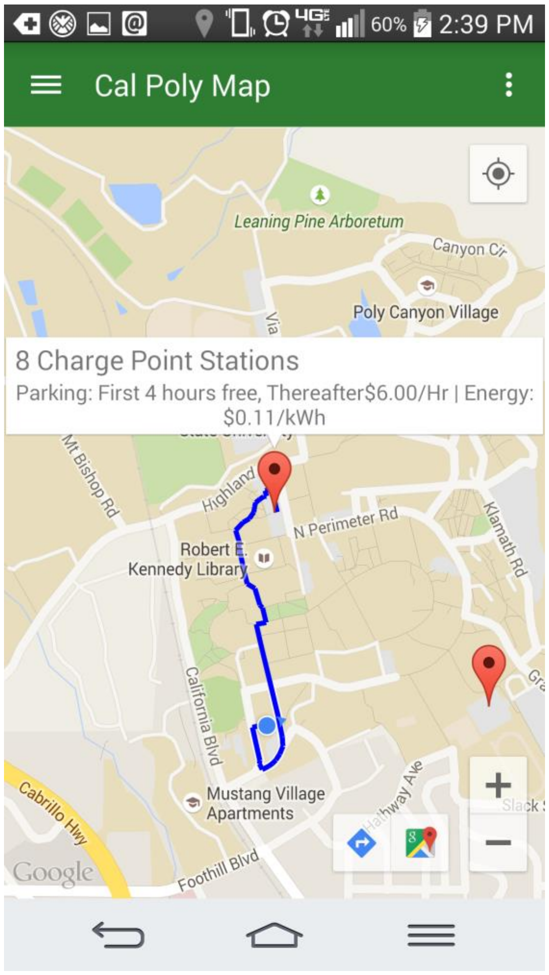
Figure 9: Software Flow Diagram of the Points of Interest Feature


2:39 PM
Cal Poly Map
Poly Canyon Village
California Polytechnic State University
Robert E. Kennedy Library
California Blvd
Slack St

R-3: Village Parking Structure
929 Resident | 1 Staff | 11 Disabled | 25 Metered

Figure 10: Parking Lot Points of Interest

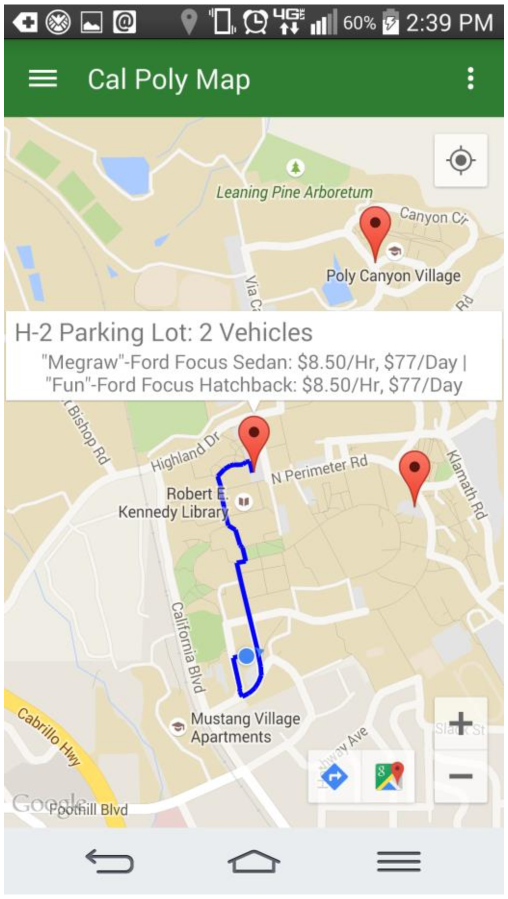
Figures 10, 11, and 12 are screenshots of some of the points of interest categories on campus. Figure 10 shows all of the parking lot points of interest. When a point is selected, the name of the parking lot appears as well as a static description of its capacity (resident parking, staff parking, etc.) Figure 11 displays the charge points that are located on campus for electric vehicles and the rate at which the user pays per hour. Finally, Figure 12 is a screenshot of the Zipcar points of interest, which shows where users can pick up zipcars and which vehicles are available.



8 Charge Point Stations

Parking: First 4 hours free, Thereafter$6.00/Hr | Energy: $0.11/kWh

Figure 11: Change Point Points of Interest



H-2 Parking Lot: 2 Vehicles
"Megraw"-Ford Focus Sedan: $8.50/Hr, $77/Day | "Fun"-Ford Focus Hatchback: $8.50/Hr, $77/Day

Figure 12: Zipcar Points of Interest

c. Zipcar

Figure 13 shows the block diagram of the Zipcar feature. The following figures, Figure 15, 16, 17, and 18, show different screens which a user may see when reserving a Zipcar. First (figure 15), they go to the Zipcar main menu, which shows three main options: Register, Reserve, and Locations. Figure 16 shows the different types of accounts the user can register for, depending on what group (or groups) they belong to at Cal Poly. The icons used for the menu options in figures 15 and 16 are only temporary placeholders until the portal team designers finalize the icons which they'd like to use. Once the user selects an account type, they can register for an account (Figure 17) and get benefits relating to their account. Figure 18 shows the log in screen for existing Zipcar users. Figure 14 shows the software flow diagram of the Zipcar feature. There are four classes involved: Zipcar, ZipcarSectionAdapter, ZipcarLocation, and RegisterZipcar. Zipcar waits for the user to press a button, which decides where to go. The user is taken to the mobile webpage if the reserve button is pressed. If the location button is pressed, the app goes to the ZipcarLocation which takes in the array of names, subtext, and coordinates and displays the points of interest on the map. If the Register button is pressed, the app will go to the RegisterZipcar class which takes in the discount's name and URL from an XML file.

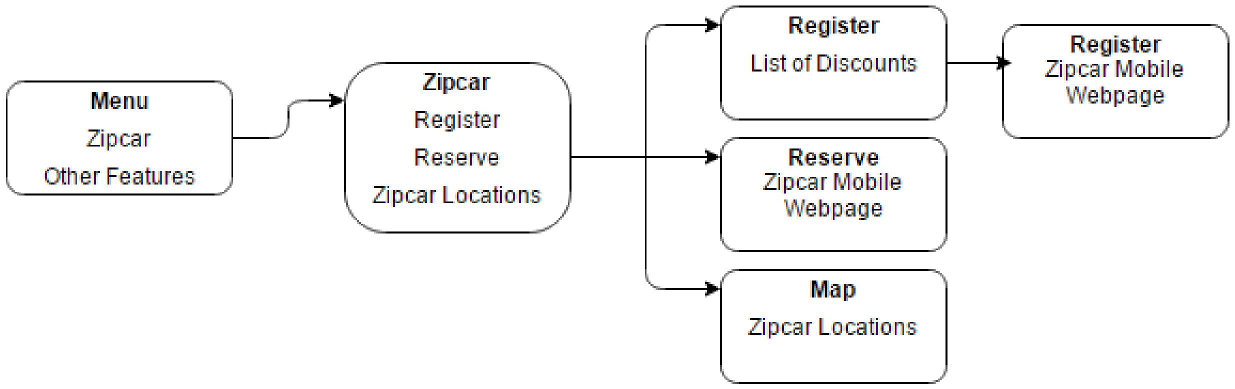


Figure 13: Block Diagram of the Zipcar Feature

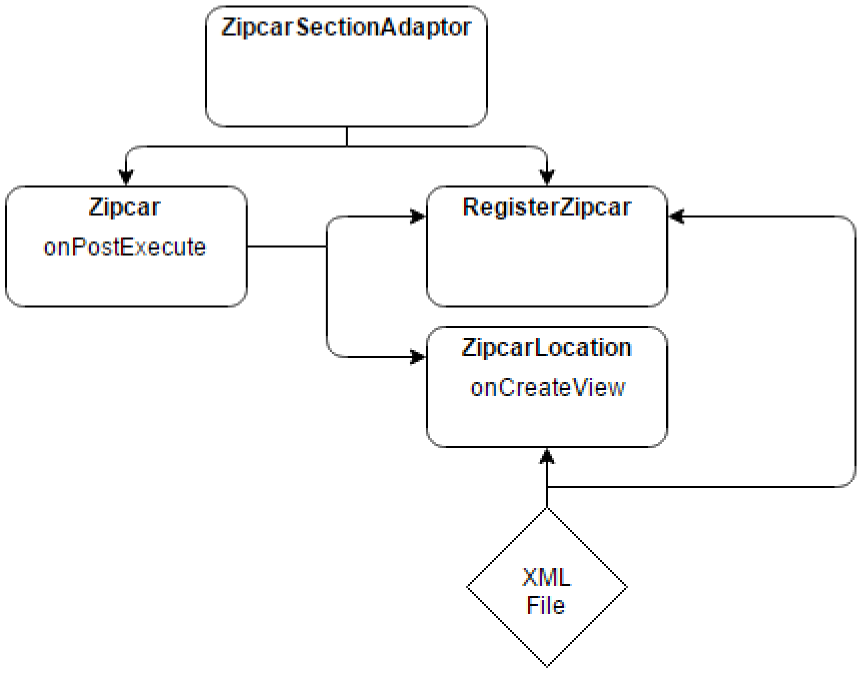


Figure 14: Software Flow Diagram of the Zipcar Feature

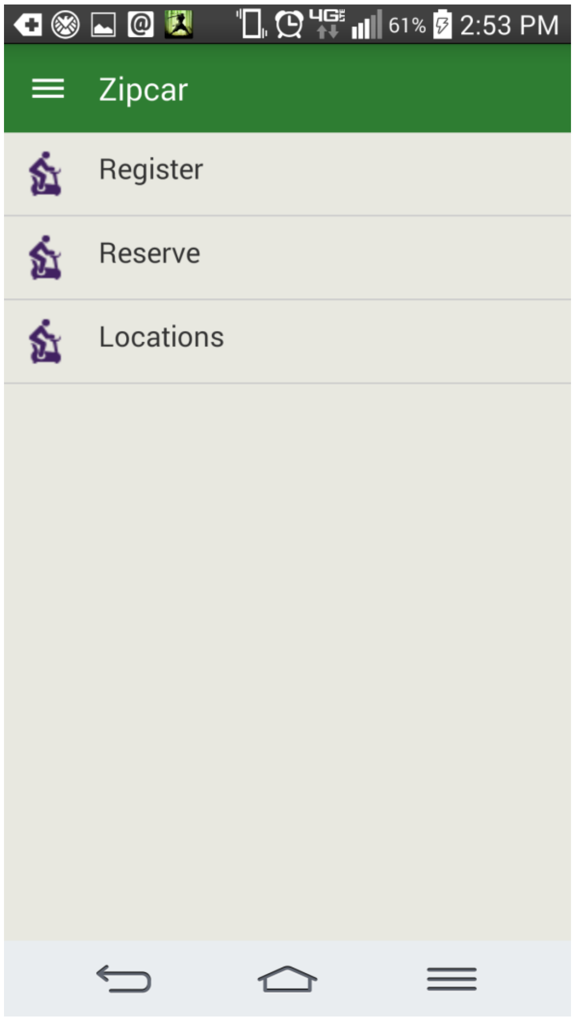
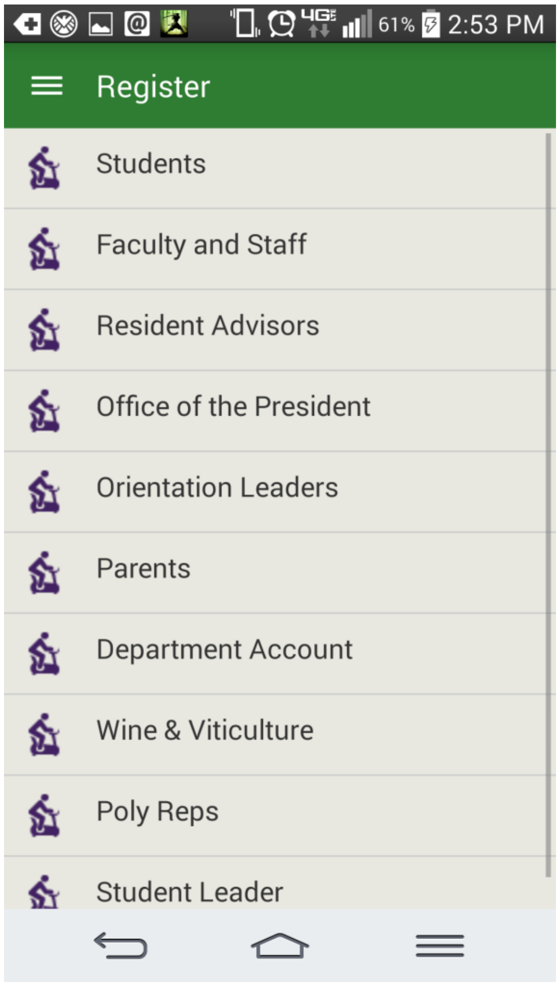


Figure 15: Zipcar main menu options (icons are placeholders for new icons that the portal team is developing)



Students

Faculty and Staff

Resident Advisors

Office of the President

Orientation Leaders

Parents

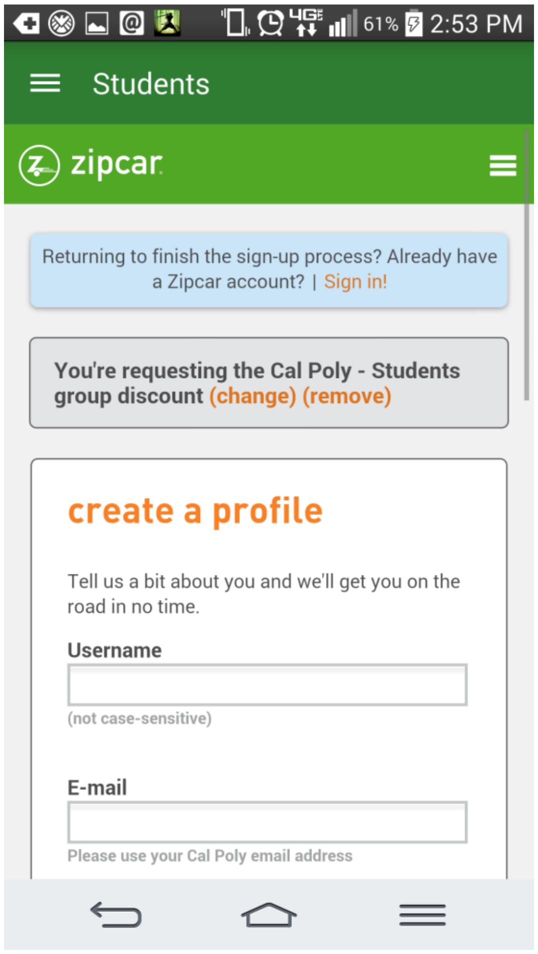
Department Account

Wine & Viticulture

Poly Reps

Student Leader

Figure 16: Zipcar registration type menu based on user type (icons are placeholders until the portal group finalizes their icons)


Students

Returning to finish the sign-up process? Already have a Zipcar account? | Sign in!

You're requesting the Cal Poly - Students group discount (change) (remove)

create a profile

Tell us a bit about you and we'll get you on the road in no time.

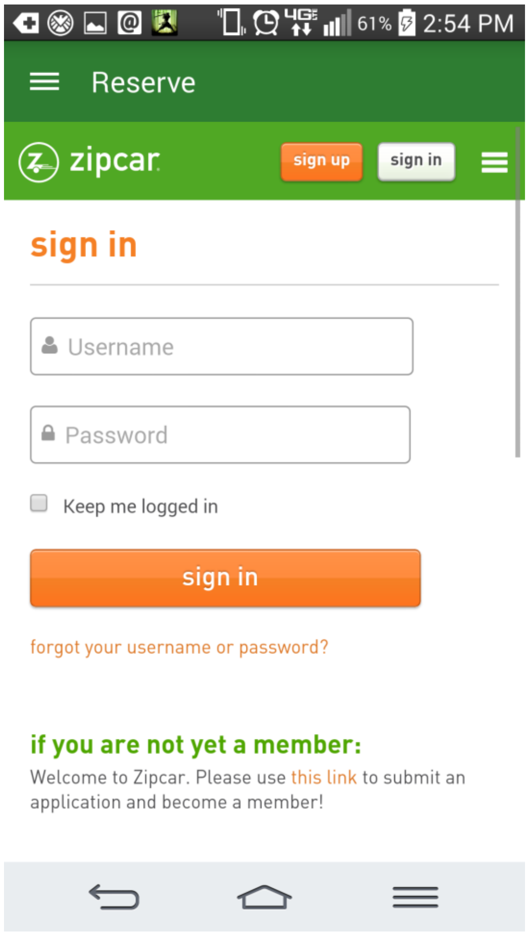
Username

(not case-sensitive)

E-mail

Please use your Cal Poly email address

Figure 17: Zipcar registration signup page for students



sign in

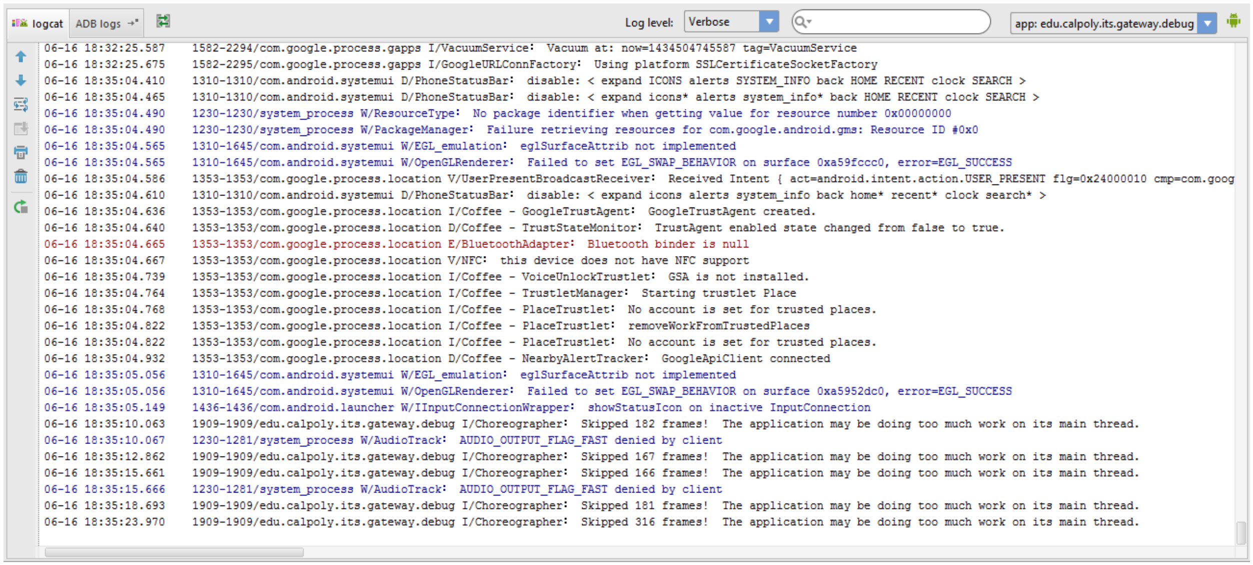
if you are not yet a member:

Welcome to Zipcar. Please use this link to submit an application and become a member!

Figure 18: Existing user login page for reserving a Zipcar

E. Testing

The features were developed on Android Studio 1.01. To test whether the features were working, both the Android Studio's emulator and a LG G2. To run the program on the emulator, the Android virtual machine chosen was the Nexus 5 API 21 x86. The LG G2 used Android version 4.4.2. In order to catch errors, the Android logging system, logcat, was used (Figure 19). This system was extremely useful in catching errors when testing the features. Most of the errors when developing caused the app to crash and using the logcat helped to find the errors quickly.



Log level: Verbose

app: edu.calpoly.its.gateway.debug

```
06-16 18:32:25.587    1582-2294/com.google.process.gapps I/VacuumService:  Vacuum at: now=1434504745587 tag=VacuumService
06-16 18:32:25.675    1582-2295/com.google.process.gapps I/GoogleURLConnFactory:  Using platform SSLCertificateSocketFactory
06-16 18:35:04.410    1310-1310/com.android.systemui D/PhoneStatusBar:  disable: < expand ICONS alerts SYSTEM_INFO back HOME RECENT clock SEARCH >
06-16 18:35:04.465    1310-1310/com.android.systemui D/PhoneStatusBar:  disable: < expand icons* alerts system_info* back HOME RECENT clock SEARCH >
06-16 18:35:04.490    1230-1230/system_process W/ResourceType:  No package identifier when getting value for resource number 0x00000000
06-16 18:35:04.490    1230-1230/system_process W/PackageManager:  Failure retrieving resources for com.google.android.gms: Resource ID #0x0
06-16 18:35:04.565    1310-1645/com.android.systemui W/EGL_emulation:  eglSurfaceAttrib not implemented
06-16 18:35:04.565    1310-1645/com.android.systemui W/OpenGLRenderer:  Failed to set EGL_SWAP_BEHAVIOR on surface 0xa59fccc0, error=EGL_SUCCESS
06-16 18:35:04.586    1353-1353/com.google.process.location V/UserPresentBroadcastReceiver:  Received Intent { act=android.intent.action.USER_PRESENT flg=0x24000010 cmp=com.goog
06-16 18:35:04.610    1310-1310/com.android.systemui D/PhoneStatusBar:  disable: < expand icons alerts system_info back home* recent* clock search* >
06-16 18:35:04.636    1353-1353/com.google.process.location I/Coffee - GoogleTrustAgent:  GoogleTrustAgent created.
06-16 18:35:04.640    1353-1353/com.google.process.location D/Coffee - TrustStateMonitor:  TrustAgent enabled state changed from false to true.
06-16 18:35:04.665    1353-1353/com.google.process.location E/BluetoothAdapter:  Bluetooth binder is null
06-16 18:35:04.667    1353-1353/com.google.process.location V/NFC:  this device does not have NFC support
06-16 18:35:04.739    1353-1353/com.google.process.location I/Coffee - VoiceUnlockTrustlet:  GSA is not installed.
06-16 18:35:04.764    1353-1353/com.google.process.location I/Coffee - TrustletManager:  Starting trustlet Place
06-16 18:35:04.768    1353-1353/com.google.process.location I/Coffee - PlaceTrustlet:  No account is set for trusted places.
06-16 18:35:04.822    1353-1353/com.google.process.location I/Coffee - PlaceTrustlet:  removeWorkFromTrustedPlaces
06-16 18:35:04.822    1353-1353/com.google.process.location I/Coffee - PlaceTrustlet:  No account is set for trusted places.
06-16 18:35:04.932    1353-1353/com.google.process.location D/Coffee - NearbyAlertTracker:  GoogleApiClient connected
06-16 18:35:05.056    1310-1645/com.android.systemui W/EGL_emulation:  eglSurfaceAttrib not implemented
06-16 18:35:05.056    1310-1645/com.android.systemui W/OpenGLRenderer:  Failed to set EGL_SWAP_BEHAVIOR on surface 0xa5952dc0, error=EGL_SUCCESS
06-16 18:35:05.149    1436-1436/com.android.launcher W/IInputConnectionWrapper:  showStatusIcon on inactive InputConnection
06-16 18:35:10.063    1909-1909/edu.calpoly.its.gateway.debug I/Choreographer:  Skipped 182 frames!  The application may be doing too much work on its main thread.
06-16 18:35:10.067    1230-1281/system_process W/AudioTrack:  AUDIO_OUTPUT_FLAG_FAST denied by client
06-16 18:35:12.862    1909-1909/edu.calpoly.its.gateway.debug I/Choreographer:  Skipped 167 frames!  The application may be doing too much work on its main thread.
06-16 18:35:15.661    1909-1909/edu.calpoly.its.gateway.debug I/Choreographer:  Skipped 166 frames!  The application may be doing too much work on its main thread.
06-16 18:35:15.666    1230-1281/system_process W/AudioTrack:  AUDIO_OUTPUT_FLAG_FAST denied by client
06-16 18:35:18.693    1909-1909/edu.calpoly.its.gateway.debug I/Choreographer:  Skipped 181 frames!  The application may be doing too much work on its main thread.
06-16 18:35:23.970    1909-1909/edu.calpoly.its.gateway.debug I/Choreographer:  Skipped 316 frames!  The application may be doing too much work on its main thread.
```

Figure 19: Logcat

F. Conclusion

This project featured parking features, the Points of Interest, the Zipcar, and the Announcements for the UPD, that were not previously in the official Cal Poly mobile application. I was able to create these features for the Cal Poly University Police department. Although the project did not feature a live version of the parking lots and the spaces available, the static data was sufficient for users on campus. There may be more features added to the app in the future, but the added features will help users on campus in the fall quarter when it is released.

G. Senior Project Analysis

Schedule and Costs

	Nov-14	Dec-14	Jan-15	Feb-15	Mar-15	Apr-15	May-15
Plan Specifications with Portal Group and University Police Department *(Note: Should meet with Portal Group and University Police Department every month)*	■	■	■	■	■	■	■
Research	▒	▒					
-Android SDK	▒	▒					
-Java	▒	▒					
Building Shell (All Features)		■	■				
Implementing Features		■	■	■	■		
-Announcements		▒	▒				
-Points of Interest			▒	▒			
-Zipcar				▒	▒		
Milestone 1: All Features Implemented					◊		
Testing (Using Predetermined Database)					■	■	■
-Test Announcements					▒	▒	
-Test Points of Interest					▒	▒	
-Test Zipcar					▒	▒	
-Work on Server Issues							▒
-Test Multiple Devices							▒
Milestone 2: Application Finished							◊
Work on Hosting Data on Cal Poly Server (Optional)							
Build Vehicle Detection System (Optional)					░	░	░

Figure 20: Gantt Chart

a. Resources

1. Android Phone
2. Android SDK
3. Windows Computer
4. Application Server for hosting data
5. Twitter API

b. Skills Required

1. Experience with Java
2. Experience with mobile application design
3. Experience working with servers

H. References

1. Mimbela, Luz E., and Lawrence A. Klein. "SUMMARY OF VEHICLE DETECTION AND SURVEILLANCE TECHNOLOGIES USED IN INTELLIGENT TRANSPORTATION SYSTEMS." (n.d.): n. pag. *Http://www.fhwa.dot.gov/ohim/tvtw/vdstits.pdf*. Federal Highway Administration S (FHWA) Intelligent Transportation Systems Joint Program Office, Nov. 2000. Web.

   This paper was intended to explain vehicle detection and surveillance technologies in detail. It covers a wide variety of technologies such as eumatic road tube inductive loop detectors, piezoelectric sensors, magnetic sensors, weigh in motion (WIM), video image processor, microwave radar infrared sensors, ultrasonic sensors, and passive acoustic array sensors. The paper explains each technology by using vendor-provided information about specific models. This will be useful in learning about vehicle detection, which will be used for the Parking Lot Capacity feature.

2. Amazon.com : Dakota Alert 2500 Wireless Vehicle Detection Probe Sensor (DCPT-2500) : Security Sensors : Camera & Photo. Amazon, n.d. Web. 27 Oct. 2014.<Http://www.amazon.com/Dakota-Alert-DCPT-2500-Wireless-Detection /dp/B0049D2BQ0>.

   This product is a vehicle detection sensor that may be used in the project. The datasheet will be useful in learning about vehicle detection.

3. "Cal Poly Quick Facts." - *Find Out About Academics, Student Body, Campus Size, History, Graduates & Careers, Buildings and More*. Cal Poly, 2012. Web. 27 Oct. 2014. <http://calpolynews.calpoly.edu/quickfacts.html>.

   This site lists the latest statistics on Cal Poly, which is useful for market research for this particular application.

4. "Vehicle Detection » Products & Applications » Banner Engineering Corp."*Vehicle Detection » Products & Applications » Banner Engineering Corp.* Banner Engineering, 2014. Web. 27 Oct. 2014. <http://www.bannerengineering.com/en-US/products/8/ Sensors/44/Vehicle-Detection>.

   This product is another vehicle detection sensor which may be used for the Parking Lot Capacity feature. The datasheet will be useful for learning about vehicle detection.

5. "Getting Started." *Android Developers*. N.p., n.d. Web. 25 Nov. 2014. <https://developer.android.com/training/index.html>.

This website will be extremely helpful in learning about Android application development. It covers the basics of creating an application as well as how to connect the application to a server. There are many other topics that the website covers and will be the primary website that I will use to learn about Android application development.

6. "Design | Android Developers." *Design | Android Developers*. N.p., n.d. Web. 25 Nov. 2014. <https://developer.android.com/design/index.html>.

   This website covers the basic design that Android applications should have. This will help me design the application to fit the University Police Department specifications for the applications look.

Appendix A: Senior Project Analysis

Project Title: Cal Poly Parking Android Application

Student: Ryan Lin

Advisor: Dr. Bridget Benson

1. Summary of Functional Requirements

    This application will help those on the Cal Poly campus in the year 2015 by saving people time, fuel, and money. The following features will be implemented into the application: announcements, zipcar, and points of interest. The announcements feature will allow a third party to update the announcements tab to keep users informed about events that may alter the availability of the roads and parking lots on campus. The zipcar feature allows users to find zipcar locations on campus and reserve them. The points of interest feature will help users find locations on the campus map.

2. Primary Constraints

    There are several issues that may inhibit the development of the application. The primary difficulty would be to program this application efficiently by building on my knowledge of java. I would also have to learn about android development by using programs such as the Android SDK and Eclipse. Another concern for this project is being able to implement the application in a way that will allow users to get the information they need quickly. This means having an easy to use UI and an optimized and seamless user experience. Besides technological concerns, the only difficulty would be getting the authorization to implement this application for the Cal Poly Organization. The last concern would be time. Because of this, only the core features may be implemented and tested on using a preset fictitious database.

3. Economic

    Since the application is limited to the Cal Poly campus, this particular project does not have much of an economical impact. There will be no profit made from the application and the only potential economical benefits would be that the application could save money for users. The costs would include maintaining the application and hosting the data on a server. The actual cost of this is unknown since the application is still in development. The cost of developing the application would only include my time and does not require any paid software.

4. If Manufactured on a Commercial Basis
    a. Estimated number of downloads per year: 800
    b. Estimated manufacturing cost: $0
    c. Estimated annual maintenance cost: $876
    d. Estimated profit per year: $0

5. Environmental

The application itself will have a minimal effect on the environment by consuming power. The only power consumed will be that of the servers that maintain the data and the device of the user.

6. Manufacturability

There will be no manufacturing required for the application.

7. Sustainability

There may be several issues regarding sustainability. There may be several bugs within the application itself that may be discovered in the future. This is because it is impossible to test every case on every different type of Android device there is. Bugs may also arise when newer, more upgraded, systems are used or if new features are implemented in the application.. Another potential issue is the server overload, which may occur when there are too many users at one time. This would require the Portal Group to fix this issue if the application is hosted on the Cal Poly servers. Finally, there may be changes to the number of parking spots in each lot or changes in zipcar locations. This data will have to be modified on the Cal Poly server so that the updated information will appear in the application.

The application can be upgraded in the future by adding more features, yet maintaining a simple and easy to use user interface. Improving the application also includes fixing bugs and making code more efficient.

8. Ethical

Positive ethical implications of the Cal Poly Parking application include saving users time, money, and fuel, increasing convenience on the Cal Poly campus. There should only be a few negative ethical implications since the application is used primarily for information distribution. Any misinformation can be fixed simply by contacting the Portal Group or the University Police Department. The only major negative ethical implication is that the application may cause issues with the users' devices, such as crashes, data loss, corrupt data, and other technical issues. One way to prevent these issues from happening is to test the application many times on many different types of devices. The application will also be reviewed by the Portal Group because it will represent the Cal Poly Organization in some way.

9. Health and Safety

There are very few health and safety concerns with this application. It is strongly advised that users do not use the application while driving.

10. Social and Political

There is no political effect from this application. The only social effect the application would have would be contained within the campus, particularly the University Police Department, the Portal Group, and the users. The Portal Group would be responsible for maintaining and updating the application. The University Police Department would be responsible for information distribution.

11. Development

By developing this application, I will be able to learn about mobile application development and Android systems and increase my proficiency with Java. Since the application will need to display data from a computer, I will learn about servers and how they work with mobile devices.

12. Market Research

The most relatable solution would be the official Cal Poly application, which still does not have these features. For now, people on the Cal Poly campus have to deal with the issues described in the overview. However, there is a similar application out on the Android Apps on Google Play and Apple App Store on iTunes called the FSU Tranz. This application was built with the same purpose of increasing the convenience for users at the Florida State University. This idea is what inspired the project.

This project will be in the mobile application development market, but will be specific to the purpose of helping those in Cal Poly. The application will be developed for the android for the project and may be implemented for the iphone later on due to time constraints. The organization that will be close to this project will consist of various departments of Cal Poly. The customer, the Cal Poly's University Police Department, will be in charge of setting the final requirements for the features of this application. The Portal Group in the ITS Department will be working closely on this project and will assist and make decisions on the project since the group manages various tasks, including the official Cal Poly application. The programmers will help to integrate the features into the official Cal Poly. The goal of this section is to explain the market and the roles of the organizations that will be working closely with the project.

List of organizations in the market

1. Portal Group
2. Cal Poly Organization

There would be no need to compete with others in the market because the purpose of this project is to only solve the needs of those on the campus. There are organizations already in the market, including the Portal Group since the group manages the official Cal Poly application. Since there is no features in this application that solves the needs presented in this proposal, the competition would be virtually nonexistent.

Because this application is primarily for the school, the market is small. Users would mostly consist of students, staff, and visitors. The following table (Table 1) shows the population of students and employees at Cal Poly as of Fall 2011 (Cal Poly Quick Facts). There are potentially more than 21,918 customers including visitors based on this data.

Table 1: Populations of Cal Poly as of Fall 2011

Students	18,762
Faculty	1,244
Staff	1,383
Corporation	445
ASI	84
Total	21,918

The main competitor, the Portal Group, will most likely have other projects that will occupy their time. Since the University Police Department requested this application and would like it to be done in the near future, the Portal Group would most likely would not be able to complete it within the year. Because of this, the window of opportunity would be approximately a year and would not require much effort to enter this market.

The Cal Poly Organization, Portal Group, and University Police Department will work closely on this project the most. If the application isn't integrated into the official Cal Poly application and is made into a standalone, the Portal Group will be able to promote the application through the official Cal Poly application and the my.calpoly.edu site. The Police Department, the customer, can also promote this application and its features.

Table 2: Advantages and Disadvantages of the Application

Advantages	**Disadvantages**
Decreases time and fuel required to find parking	Maintenance required for data feeds and potentially changing data
Decreases cost of fuel for users	Possible bugs
Helps users find and reserve zipcars	
Helps users find electric charging stations	
Simple and easy to use UI	

a. Code

Announcements

NewsReader.java (modified)

```
package edu.calpoly.its.gateway.news;

/*
 * Displays the list of news feeds
 */

import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

import com.flurry.android.FlurryAgent;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.events.Post;

public class NewsReader extends BaseActivity {
        /** Called when the activity is first created. */
        private final String feedlist = "http://m.calpoly.edu/feeds/newsfeedlist.xml";
        private final ArrayList<Post> PostList = new ArrayList<Post>();
        private ListView mainListView;
        private ArrayAdapter<String> listAdapter;

        //Feed Lists (because objects would be overkill)
  private final ArrayList<String> listOfFeedNames = new ArrayList<String>();
        private final ArrayList<String> listOfFeedURLs = new ArrayList<String>();

        //Caching xml file in internal storage
        private final String filename = "newsfeedlist.xml";
  private String parkingMenuItemName; // set in onCreate
  //private final String parkingMenuItemName = this.getString(R.string.twitter_menu_parking);
  private ProgressDialog ShowProgress;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
    parkingMenuItemName = this.getString(R.string.twitter_menu_parking);
                ShowProgress = ProgressDialog.show(NewsReader.this, "",
                                "Loading...", true);
                new loadingTask().execute(feedlist);

                setContentView(R.layout.newsreadermenu);
                mainListView = (ListView) findViewById(R.id.mainListView);
                listAdapter = new ArrayAdapter<String>(this, R.layout.newssourceentry,
listOfFeedNames);

                mainListView.setOnItemClickListener(new OnItemClickListener() {
                        @Override
                        public void onItemClick(AdapterView<?> parent, View view, int position, long id)
{
         Intent viewFeed;
         String currentlySelectedItem = mainListView.getItemAtPosition(position).toString();
         if (currentlySelectedItem.equals(parkingMenuItemName)) {
            viewFeed = new Intent(NewsReader.this, TwitterFeedView.class);
        }
        else {
            viewFeed = new Intent(NewsReader.this, NewsFeedView.class);
        }
        Bundle feedInfo = new Bundle();
        feedInfo.putString("feedSelection", listOfFeedURLs.get(position));
        feedInfo.putString("feedName", listOfFeedNames.get(position));
        viewFeed.putExtras(feedInfo);
        startActivity(viewFeed);
                        }
                });

                Button refreshButton = (Button) findViewById(R.id.refreshButton);
                refreshButton.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                new CacheTask().execute("");
                                startActivity(new Intent(NewsReader.this, NewsReader.class));
                                finish();
                        }
                });

                actionBar.setTitle("Cal Poly News Reader");
        }

        private class loadingTask extends AsyncTask<String, Void, String> {

                @Override
                protected String doInBackground(String... urls) {
                        listOfFeedNames.clear();
                        listOfFeedURLs.clear();
                        InputStream inputStream;
                        try {
        long updateCache = 604800000L;
        if (!checkAppFile(getApplicationContext(), filename)) {
                                        if (isOnline(getApplicationContext()))
                                                writeAppFile(getApplicationContext(), filename, new
URL(feedlist).openStream());
                                        else
                                                return "no conection";
                                }
                                else if (ifReCache(getApplicationContext(), filename, updateCache))
                                        if (isOnline(getApplicationContext()))
                                                writeAppFile(getApplicationContext(), filename, new
URL(feedlist).openStream());

                                inputStream = readAppFile(getApplicationContext(), filename);

                                XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
                                factory.setNamespaceAware(true);
                                XmlPullParser xpp = factory.newPullParser();
                                xpp.setInput(inputStream, "iso-8859-1");
                                int eventType = xpp.getEventType();
                                Post currentPost = new Post();
                                while (eventType != XmlPullParser.END_DOCUMENT) {
                                        switch(eventType) {
                                                case XmlPullParser.START_TAG:
                                                    if (xpp.getName().equalsIgnoreCase("item"))
                                                        currentPost = new Post();
                                                    else if
(xpp.getName().equalsIgnoreCase("title")) {
                                                        String tmp = xpp.nextText();
                                                        listOfFeedNames.add(tmp);
                                                    }
                                                    else if (xpp.getName().equalsIgnoreCase("link"))
                                                        listOfFeedURLs.add(xpp.nextText());
                                                    break;
                                                case XmlPullParser.END_TAG:
                                                    if (xpp.getName().equalsIgnoreCase("item"))
                                                        PostList.add(currentPost);
                                                    break;
                                                default:
                                                    break;
                                        }
                                        eventType = xpp.next();
                                }

                        } catch (MalformedURLException e1) {
                                e1.printStackTrace();
                        } catch (IOException e) {
            }

                                e.printStackTrace();
                        } catch (XmlPullParserException e) {
                                e.printStackTrace();
                        }

                        return "";
                    }

                    @Override
                    protected void onPostExecute(String s) {
                        ShowProgress.dismiss();
                        setAnimationAdapter(listAdapter, mainListView);

                        if (listOfFeedNames.size() == 0) {
                                Toast connectionError= Toast.makeText(getApplicationContext(),
                                        "An internet connection is required to view this feed",
Toast.LENGTH_LONG);
                                connectionError.setGravity(Gravity.CENTER, 0, 0);
                                connectionError.show();
                        }
                    }

                }

                private class CacheTask extends AsyncTask<String, Void, String> {

                    @Override
                    protected String doInBackground(String... params) {
                        //writeCacheFile();
                        try {
                                writeAppFile(getApplicationContext(), filename, new
URL(feedlist).openStream());
                        } catch (MalformedURLException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                        return "";
                    }
                }
```

```
	public void onStart() {
		super.onStart();
		FlurryAgent.setLogEvents(true);
		FlurryAgent.onStartSession(this, this.getString(R.string.flurryAPI));
		FlurryAgent.logEvent(this.getClass().getSimpleName());
	}

	public void onStop() {
		super.onStop();
		FlurryAgent.onEndSession(this);
	}
}
```

NewsFeedView.java (created by other developer at Cal Poly Portal Group)

```
package edu.calpoly.its.gateway.news;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.Toast;

import com.flurry.android.FlurryAgent;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.events.Post;

public class NewsFeedView extends BaseActivity {
        /** Called when the activity is first created. */
    private ListView newsReaderList;
        private ProgressDialog ShowProgress;
        private  ArrayList<Post> PostList = new ArrayList<Post>();

    @Override
        public void onCreate(Bundle savedInstanceState) {
                ////Log.d("DEBUG", "FeedView has started");
                super.onCreate(savedInstanceState);

                Bundle feedInfo = getIntent().getExtras();
                String feedSelection = feedInfo.getString("feedSelection");
                setContentView(R.layout.newsfeedview);
                newsReaderList = (ListView) findViewById(R.id.newsReaderList);

                ShowProgress = ProgressDialog.show(NewsFeedView.this, "",
                                "Loading...", true);
                new loadingTask().execute(feedSelection);

                if (!isOnline()) {
                        Toast connectionError= Toast.makeText(getApplicationContext(),
                                        "An internet connection is required to view this feed",
Toast.LENGTH_LONG);
                        connectionError.setGravity(Gravity.CENTER, 0, 0);
                        connectionError.show();
                }

                newsReaderList.setOnItemClickListener(new OnItemClickListener() {
                        public void onItemClick(AdapterView<?> parent, View view,
                                        int position, long id) {
```

```
                                        Intent intent = new
Intent(Intent.ACTION_VIEW).setData(Uri.parse(PostList.get(position).getUrl()));
                                        startActivity(intent);

                              }
                    });

                    actionBar.setTitle(feedInfo.getString("feedName"));
          }

          boolean isOnline() {
             ConnectivityManager cm =
                (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

             return cm.getActiveNetworkInfo() != null &&
               cm.getActiveNetworkInfo().isConnectedOrConnecting();
          }

          private class loadingTask extends AsyncTask<String, Void, String> {

                    protected String doInBackground(String... urls) {

                              int parseCount = 0;
                              String next;

                              try {
                                        URL url = new URL(urls[0]);
                                        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
                                        factory.setNamespaceAware(true);
                                        XmlPullParser xpp = factory.newPullParser();
                                        xpp.setInput(url.openStream(), "iso-8859-1");
                                        int eventType = xpp.getEventType();
                                        Post currentPost = new Post();
           int newsListTruncate = 30;
           while (eventType != XmlPullParser.END_DOCUMENT && parseCount < newsListTruncate) {
                                                  switch(eventType) {
                                                            case XmlPullParser.START_TAG:
                                                                      if (xpp.getName().equalsIgnoreCase("item"))
                                                                                currentPost = new Post();
                                                            else if
(xpp.getName().equalsIgnoreCase("title"))
                                                                    currentPost.setTitle(xpp.nextText());
                                                            else if
(xpp.getName().equalsIgnoreCase("description")) {
                                                                    next = xpp.nextText();
                int newsDescTruncate = 80;
                if (next.length() > newsDescTruncate)

        currentPost.setDescription(next.substring(0, newsDescTruncate) + "...");
                                                                    else

        currentPost.setDescription(next);
                                                            }
                                                            else if
(xpp.getName().equalsIgnoreCase("pubDate"))

        currentPost.setPubDate(xpp.nextText());
                                                            else if
(xpp.getName().equalsIgnoreCase("thumbnail"))

        currentPost.setThumbnail(xpp.nextText());
                                                            else if (xpp.getName().equalsIgnoreCase("link"))
                                                                    currentPost.setUrl(xpp.nextText());
                                                            break;
                                                    case XmlPullParser.END_TAG:
                                                            if (xpp.getName().equalsIgnoreCase("item")) {
                                                                    parseCount++;
                                                                    PostList.add(currentPost);
                                                            }
                                                            break;
                                                    default:
                                                            break;
                                            }
                                            eventType = xpp.next();
                                    }

                            } catch (MalformedURLException e1) {
                                    e1.printStackTrace();
                            } catch (IOException e) {
```

```
                    e.printStackTrace();
                } catch (XmlPullParserException e) {
                    e.printStackTrace();
                }

                return "";
            }

            protected void onPostExecute(String s) {
              setAnimationAdapter(new NewsEntryAdapter(NewsFeedView.this, PostList),
newsReaderList);
                ShowProgress.dismiss();
            }
        }

        @Override
    public void onBackPressed() {
      finish();
      overridePendingTransition(BaseActivity.GENERAL_BACK_ANIM_IN,
BaseActivity.GENERAL_BACK_ANIM_OUT);
    }

        public void onStart() {
            super.onStart();
            FlurryAgent.setLogEvents(true);
            FlurryAgent.onStartSession(this, this.getString(R.string.flurryAPI));
            FlurryAgent.logEvent(this.getClass().getSimpleName());
        }

        public void onStop() {
            super.onStop();
            FlurryAgent.onEndSession(this);
        }
}
```

TwitterFeedView.java (created by Ryan Lin)

```
package edu.calpoly.its.gateway.news;

import android.app.ProgressDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;

import com.flurry.android.FlurryAgent;
import com.twitter.sdk.android.Twitter;
import com.twitter.sdk.android.core.AppSession;
import com.twitter.sdk.android.core.Callback;
import com.twitter.sdk.android.core.Result;
import com.twitter.sdk.android.core.TwitterApiClient;
import com.twitter.sdk.android.core.TwitterApiException;
import com.twitter.sdk.android.core.TwitterAuthConfig;
import com.twitter.sdk.android.core.TwitterCore;
import com.twitter.sdk.android.core.internal.TwitterApiConstants;
import com.twitter.sdk.android.core.services.StatusesService;
import com.twitter.sdk.android.tweetcomposer.TweetComposer;
import com.twitter.sdk.android.tweetui.TweetUi;

import com.twitter.sdk.android.core.models.Tweet;
import com.twitter.sdk.android.core.TwitterException;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.events.Post;
import io.fabric.sdk.android.Fabric;

import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class TwitterFeedView extends BaseActivity {

    private String TWITTER_KEY; // set in onCreate
```

```
   private String TWITTER_SECRET;
   private ListView twitterReaderList;
   private ProgressDialog ShowProgress;
   private ArrayList<Post> PostList = new ArrayList<Post>();

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      Bundle feedInfo = getIntent().getExtras();
      String feedSelection = feedInfo.getString("feedSelection");

      TWITTER_KEY = getString(R.string.twitter_consumer_key);
      TWITTER_SECRET = getString(R.string.twitter_consumer_secret);
      TwitterAuthConfig authConfig = new TwitterAuthConfig(TWITTER_KEY, TWITTER_SECRET);
      Fabric.with(this, new Twitter(authConfig));
      Fabric.with(this, new TweetComposer());
      Fabric.with(this, new TwitterCore(authConfig));
      Fabric.with(this, new TwitterCore(authConfig), new TweetUi());

      setContentView(R.layout.newsfeedview);
      twitterReaderList = (ListView) findViewById(R.id.newsReaderList);

      ShowProgress = ProgressDialog.show(TwitterFeedView.this, "",
            "Loading...", true);
      new loadingTask().execute(feedSelection);

      twitterReaderList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
         public void onItemClick(AdapterView<?> parent, View view,
                        int position, long id) {

            Intent intent = new
Intent(Intent.ACTION_VIEW).setData(Uri.parse(PostList.get(position).getUrl()));
            startActivity(intent);

         }
      });

      actionBar.setTitle(feedInfo.getString("feedName"));
   }
```

```
    private class loadingTask extends AsyncTask<String, Void, String> {
        protected String doInBackground(String... urls) {
            // TRY TO GET API TO GET LIST OF TWEETS
            TwitterCore.getInstance().logInGuest(new Callback() {
                @Override
                public void success(Result result) {
                    AppSession guestAppSession = (AppSession) result.data;

                    TwitterApiClient twitterApiClient = TwitterCore.getInstance().getApiClient(guestAppSession);
                    StatusesService statusesService = twitterApiClient.getStatusesService();
                    //statusesService.userTimeline(3007124872L, "burritoboy888", 30, null, null, null, null, null,
null, new Callback<List<Tweet>>() {
                    statusesService.userTimeline(36467478L, "CalPolyPolice", 30, null, null, null, null, null, null,
new Callback<List<Tweet>>() {
                        @Override
                        public void success(Result<List<Tweet>> result) {

                            Post currentPost;
                            int tweetTitleStartIndex;
                            int tweetTitleEndIndex;
                            String tweetTitle;
                            String tweetDescription;
                            String tweetUrl;

                            for (Tweet t : result.data) {
                                //tweetIDList.add(t.id);
                                //tweetTextList.add(t.text);
                                currentPost = new Post();
                                /*if (t.text.contains("&lt;") && t.text.contains("&gt;")) {
                                    tweetTitleStartIndex = t.text.indexOf("&lt;");
                                    tweetTitleEndIndex = t.text.indexOf("&gt;");

                                    if((tweetTitleEndIndex < tweetTitleStartIndex) || tweetTitleStartIndex != 0){
                                        //Incorrect Format
                                        tweetTitle = "Cal Poly Police Announcement";
                                        tweetDescription = t.text;
                                    }
                                    else {
                                        //Correct Format
            tweetTitle = t.text.substring(tweetTitleStartIndex + 4, tweetTitleEndIndex);
            tweetDescription = t.text.substring(tweetTitleEndIndex + 4);
          }
        }
        else {
          //IncorrectFormat
          tweetTitle = "Cal Poly Police Announcement";
          tweetDescription = t.text;
        }*/
        int twitterTitleTruncate = 30;
        int twitterDescTruncate = 80;
        if (t.text.length() > twitterTitleTruncate) {
          tweetTitle = t.text.substring(0, twitterTitleTruncate) + "...";
        }
        else {
          tweetTitle = t.text;
        }
        if (t.text.length() > twitterDescTruncate) {
          tweetDescription = t.text.substring(0, twitterDescTruncate) + "...";
        }
        else {
          tweetDescription = t.text;
        }
        if(tweetDescription.contains("&lt;")){
          tweetDescription = tweetDescription.replaceAll("&lt;", "<");
        }
        if(tweetDescription.contains("&gt;")){
          tweetDescription = tweetDescription.replaceAll("&gt;", ">");
        }
        if(tweetDescription.contains("&amp;")){
          tweetDescription = tweetDescription.replaceAll("&amp;", "&");
        }
        tweetUrl = "https://twitter.com/CalPolyPolice/status/" + t.idStr;
        currentPost.setTitle(tweetTitle);
        currentPost.setDescription(tweetDescription);
        currentPost.setUrl(tweetUrl);
        PostList.add(currentPost);
      }
      for (Post p: PostList) {
      }
```

```
                        setAnimationAdapter(new NewsEntryAdapter(TwitterFeedView.this, PostList),
twitterReaderList);
                        ShowProgress.dismiss();
                    }
                    @Override
                    public void failure(TwitterException exception) {
                        final TwitterApiException apiException = (TwitterApiException) exception;
                        final int errorCode = apiException.getErrorCode();
                        if (errorCode == TwitterApiConstants.Errors.APP_AUTH_ERROR_CODE || errorCode ==
TwitterApiConstants.Errors.GUEST_AUTH_ERROR_CODE) {
                            // request new guest AppSession (i.e. logInGuest)
                            // optionally retry
                        }
                    }
                });
            }

            @Override
            public void failure(TwitterException exception) {
                Log.e("TwitterFeedView", "Authentication error for Cal Poly app Twitter Feed");
                // unable to get an AppSession with guest auth
            }
        });
        return "";
    }

    protected void onPostExecute(String s) {
/*          setAnimationAdapter(new NewsEntryAdapter(TwitterFeedView.this, PostList),
twitterReaderList);
        ShowProgress.dismiss();*/
    }
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_twitter_feed_view, menu);
    return true;
  }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onBackPressed() {
        finish();
        overridePendingTransition(BaseActivity.GENERAL_BACK_ANIM_IN,
BaseActivity.GENERAL_BACK_ANIM_OUT);
    }

    public void onStart() {
        super.onStart();
        FlurryAgent.setLogEvents(true);
        FlurryAgent.onStartSession(this, this.getString(R.string.flurryAPI));
        FlurryAgent.logEvent(this.getClass().getSimpleName());
    }

    public void onStop() {
        super.onStop();
        FlurryAgent.onEndSession(this);
    }
}
```

Points of Interest

CalPolyMap.java (modified)

```
package edu.calpoly.its.gateway.map;
```

```
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.location.Location;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;

import java.util.ArrayList;
import java.util.Arrays;

import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.WelcomePage;
import edu.calpoly.its.gateway.oauth.OAuthLogin;

public class CalPolyMap extends BaseMap {

   public static final String MAP_TYPE = "map_type";
```

```
    public static final int TYPE_ESCORT_VAN = 0;
    public static final int TYPE_BLUE_LIGHTS = 1;
    public static final int TYPE_BIKE_RACKS = 2;
    public static final int TYPE_PARKING_LOTS = 3;
    public static final int TYPE_CHARGE_POINTS = 4;
    public static final int TYPE_ZIPCAR = 5;

    private boolean validMap = false;

    private static final String PREFS_BIKE_RACKS = "bike_racks";
    private static final String PREFS_BLUE_LIGHTS = "blue_lights";
    private static final String PREFS_ESCORT_VAN = "escort_van";
    private static final String PREFS_PARKING_LOTS = "parking_lots";
    private static final String PREFS_CHARGE_POINTS = "charge_points";
    private static final String PREFS_ZIPCAR = "zipcar";
    private static final String PREFS_SATELLITE = "satellite";

    protected MarkPOIOnMap marker;
    private ArrayList<Polyline> currentRoute;

    private Location target;

    /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.cpmap);


//        overridePendingTransition(BaseActivity.GENERAL_ANIM_IN, BaseActivity.GENERAL_ANIM_OUT);

        // Reduce the header size for small devices
        dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(dm);

        buildingArray = getResources().getStringArray(R.array.building_array);
        buildingList = Arrays.asList(buildingArray);
        String[] buildingLatitudeArray = getResources().getStringArray(R.array.latitude_array);
        String[] buildingLongitudeArray = getResources().getStringArray(R.array.longitude_array);
        for (int i=0;i<buildingList.size(); i++) {
          latitudeMap.put(buildingList.get(i), Double.parseDouble(buildingLatitudeArray[i]) / 1000000);
          longitudeMap.put(buildingList.get(i), Double.parseDouble(buildingLongitudeArray[i]) / 1000000);
        }

        getSupportActionBar().setTitle("Cal Poly Map");

        // Set default point and configure map
        defPoint = new LatLng(CALPOLY_LATITUDE, CALPOLY_LONGITUDE);
        validMap = setupMap();

        marker = new MarkPOIOnMap(this, cpMap);

        //prepare the soft keyboard
        inputMethodManager = (InputMethodManager)
    getSystemService(Context.INPUT_METHOD_SERVICE);

        // This widget provides the building numbers and names for searching the map
        autoTextView = (AutoCompleteTextView) findViewById(R.id.autocomplete_buildings);
        ArrayAdapter<String> buildings_adapter;
        if (dm.widthPixels <= 320)
          buildings_adapter =
              new ArrayAdapter<String>(this, R.layout.cpmapautotextsmall, buildingArray);
        else
          buildings_adapter =
              new ArrayAdapter<String>(this, R.layout.cpmapautotextlarge, buildingArray);
        autoTextView.setAdapter(buildings_adapter);
        autoTextView.setThreshold(0);
        autoTextView.setVisibility(View.INVISIBLE);
        autoTextView.setOnItemClickListener(new AutoTextOnItemClickListener());
        marker.reprintMap();

        cpMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
          @Override
          public void onInfoWindowClick(Marker marker) {
            CalPolyMap.this.routeTo(CalPolyMap.this.currentLocation(), marker.getPosition());
            target = new Location("Target");
            target.setLatitude(marker.getPosition().latitude);
            target.setLongitude(marker.getPosition().longitude);
            if (marker.getPosition().equals(FileManager.getBikeLocation(CalPolyMap.this))) {
              target.setSpeed(-1);
            }
            marker.hideInfoWindow();
          }
        });

        currentRoute = new ArrayList<Polyline>();

        int type = getIntent().getIntExtra(MAP_TYPE, -1);
        SharedPreferences prefs = getSharedPreferences(OAuthLogin.APP_PREFS, Activity.MODE_PRIVATE);
        if (prefs.getBoolean(PREFS_SATELLITE, false)) {
          cpMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
        }
        if (type == -1) {
          marker.setShowBikeRacks(prefs.getBoolean(PREFS_BIKE_RACKS, false));
          marker.setShowBlueLights(prefs.getBoolean(PREFS_BLUE_LIGHTS, false));
          marker.setShowEscortVan(prefs.getBoolean(PREFS_ESCORT_VAN, false));
          marker.setShowParkingLots(prefs.getBoolean(PREFS_PARKING_LOTS, false));
          marker.setShowChargePoints(prefs.getBoolean(PREFS_CHARGE_POINTS, false));
          marker.setShowZipcar(prefs.getBoolean(PREFS_ZIPCAR, false));
        } else {
          switch (type) {
            case TYPE_ESCORT_VAN:
              marker.setShowEscortVan(true);
              break;
            case TYPE_BLUE_LIGHTS:
              marker.setShowBlueLights(true);
              break;
            case TYPE_BIKE_RACKS:
              marker.setShowBikeRacks(true);
              break;
            case TYPE_PARKING_LOTS:
              marker.setShowParkingLots(true);
              break;
            case TYPE_CHARGE_POINTS:
              marker.setShowChargePoints(true);
              break;
            case TYPE_ZIPCAR:
              marker.setShowZipcar(true);
```

```
            break;
        }
    }
    marker.reprintMap();
}

void routeTo(LatLng from, LatLng to) {
    stopRoute();
    new GetDirection(this, from.latitude + "," + from.longitude,
                to.latitude + "," + to.longitude).execute();
}

void stopRoute() {
    if (currentRoute != null) {
        for (Polyline p : currentRoute) {
            p.remove();
        }
        currentRoute.clear();
    }
}

LatLng currentLocation() {
    Location l = cpMap.getMyLocation();
    return new LatLng(l.getLatitude(), l.getLongitude());
}

@Override
protected void onPause() {
    super.onPause();
    hideSearchTools();
    // when our activity is paused, we want to deregister for location updates to preserve battery life
    if (cpMap != null)
        cpMap.setMyLocationEnabled(false);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.map, menu);
    menu.setGroupEnabled(R.id.bike_rack_options, marker.showBikeRacks);
    menu.findItem(R.id.bike_racks).setChecked(marker.showBikeRacks);
    menu.findItem(R.id.blue_lights).setChecked(marker.showBlueLights);
    menu.findItem(R.id.escort_van).setChecked(marker.showEscortVan);
    menu.findItem(R.id.parking_lots).setChecked(marker.showParkingLots);
    menu.findItem(R.id.charge_points).setChecked(marker.showChargePoints);
    menu.findItem(R.id.zipcar).setChecked(marker.showZipcar);
    menu.findItem(R.id.menu_satellite)
        .setChecked(cpMap.getMapType() == GoogleMap.MAP_TYPE_HYBRID);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    SharedPreferences.Editor edit =
            getSharedPreferences(OAuthLogin.APP_PREFS, Activity.MODE_PRIVATE).edit();
    switch (item.getItemId()) {
        case R.id.search:
            if (isAutoTextViewShown)
                hideSearchTools();
            else
                showSearchTools();
            return true;
        case R.id.bike_racks:
            marker.setShowBikeRacks(!item.isChecked());
            invalidateOptionsMenu();
            marker.reprintMap();
            edit.putBoolean(PREFS_BIKE_RACKS, !item.isChecked());
            edit.apply();
            return true;
        case R.id.escort_van:
            invalidateOptionsMenu();
            marker.setShowEscortVan(!item.isChecked());
            marker.reprintMap();
            edit.putBoolean(PREFS_ESCORT_VAN, !item.isChecked());
            edit.apply();
            return true;
        case R.id.blue_lights:
            invalidateOptionsMenu();
            marker.setShowBlueLights(!item.isChecked());
            marker.reprintMap();
            edit.putBoolean(PREFS_BLUE_LIGHTS, !item.isChecked());
            edit.apply();
            return true;
        case R.id.parking_lots:
            invalidateOptionsMenu();
            marker.setShowParkingLots(!item.isChecked());
            marker.reprintMap();
            edit.putBoolean(PREFS_PARKING_LOTS, !item.isChecked());
            edit.apply();
            return true;
        case R.id.charge_points:
            invalidateOptionsMenu();
            marker.setShowChargePoints(!item.isChecked());
            marker.reprintMap();
            edit.putBoolean(PREFS_CHARGE_POINTS, !item.isChecked());
            edit.apply();
            return true;
        case R.id.zipcar:
            invalidateOptionsMenu();
            marker.setShowZipcar(!item.isChecked());
            marker.reprintMap();
            edit.putBoolean(PREFS_ZIPCAR, !item.isChecked());
            edit.apply();
            return true;

        case R.id.menu_satellite:
            invalidateOptionsMenu();
            if (item.isChecked()) {
                cpMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
            } else {
                cpMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
            }
            edit.putBoolean(PREFS_SATELLITE, !item.isChecked());
            edit.apply();
            return true;

        case R.id.menu_save_bike:
            if (FileManager.saveBikeLocation(this, cpMap.getMyLocation())) {
                Toast.makeText(this, R.string.toast_location_save_success, Toast.LENGTH_SHORT)
```

```
            .show();
    }
    marker.reprintMap();
    return true;
case R.id.menu_find_bike:
    LatLng location = FileManager.getBikeLocation(this);
    if (location == null) {
        Toast.makeText(this, R.string.toast_location_error, Toast.LENGTH_SHORT).show();
    } else {
        PanAndZoomMap zoom = new PanAndZoomMap(marker.bikeMarker.getPosition(), cpMap);
        Handler handler = new Handler(Looper.getMainLooper());
        handler.post(zoom);
        marker.bikeMarker.showInfoWindow();
    }
    return true;
case R.id.menu_find_brs:
    ArrayList<Rack> brs = marker.getBRS();
    Rack temp = null;
    for (Rack r : brs) {
        if (temp != null) {
            Location rloc = new Location("test");
            rloc.setLatitude(r.getLocation().latitude);
            rloc.setLongitude(r.getLocation().longitude);
            Location tloc = new Location("test");
            tloc.setLatitude(r.getLocation().latitude);
            tloc.setLongitude(r.getLocation().longitude);
            if (rloc.distanceTo(cpMap.getMyLocation()) <
                tloc.distanceTo(cpMap.getMyLocation())) {
                temp = r;
            }
        } else {
            temp = r;
        }
    }
    if (temp != null) {
        PanAndZoomMap zoom = new PanAndZoomMap(temp.getLocation(), cpMap);
        Handler handler = new Handler(Looper.getMainLooper());
        handler.post(zoom);
        temp.getMarker().showInfoWindow();
    }
```

```
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

@Override
public boolean onSearchRequested() {
    showSearchTools();
    return false;
}

/* Show the autocomplete text box and the soft keyboard */
private void showSearchTools() {
    autoTextView.setText("");
    autoTextView.setVisibility(View.VISIBLE);
    isAutoTextViewShown = true;
    inputMethodManager.toggleSoftInput(InputMethodManager.SHOW_FORCED, 0);
}

/* Hide the autocomplete text box and the soft keyboard */
private void hideSearchTools() {
    // Hide soft keyboard
    inputMethodManager.hideSoftInputFromWindow(autoTextView.getWindowToken(), 0);
    autoTextView.setVisibility(View.INVISIBLE);
    isAutoTextViewShown = false;
}

/* If user taps the back button, hide the soft keyboard and autocomplete text box if they are shown;
 * otherwise, let the system handle it.
 * back button
 * (non-Javadoc)
 * @see android.app.Activity#onBackPressed()
 */
@Override
public void onBackPressed() {
    if (isAutoTextViewShown)
        hideSearchTools();
    else {
        Intent intent = new Intent(getApplicationContext(), WelcomePage.class);
```

```
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        }
    }


    private class AutoTextOnItemClickListener implements OnItemClickListener {
        /*
         * When user selects an item in the autocomplete text box, clear the search box and keyboard from
the screen,
         * and display the building's location on the map.
         * (non-Javadoc)
         * @see
android.widget.AdapterView.OnItemClickListener#onItemClick(android.widget.AdapterView,
android.view.View, int, long)
         * @param autoTextView1 The AdapterView where the click happened.
         * @param view the view within the AdapterView that was clicked (provided by the adapter)
         * @param position The position of the view in the adapter
         * @param id The row id of the item that was clicked.
         */
        public void onItemClick(AdapterView<?> autoTextView, View view, int position, long id) {
            hideSearchTools();
            // Get the value of the item the user selected
            try {
                buildingInfoText = autoTextView.getItemAtPosition(position).toString();
            } catch (NullPointerException e) {
                e.printStackTrace();
                buildingInfoText = "";
            }

            LatLng newPoint = new LatLng((latitudeMap.get(buildingInfoText)),
                              (longitudeMap.get(buildingInfoText)));
            if (validMap) {
                MarkerOptions currentBuilding = new MarkerOptions();
                currentBuilding.position(newPoint);
                currentBuilding.title(buildingInfoText);
                marker.setCurrentBuilding(currentBuilding);
                marker.reprintMap();
                cpMap.animateCamera(CameraUpdateFactory.newLatLngZoom(newPoint, 15));
                Log.d("DEBUG", "Dropping pin for building " + buildingInfoText + " at point: " +
                        newPoint.latitude + ", " + newPoint.longitude);
            }
        }
    }
}
```

MarkPOIOnMap.java

```
package edu.calpoly.its.gateway.map;

import android.content.Context;
import android.content.res.Resources;
import android.content.res.XmlResourceParser;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.GoogleMap.InfoWindowAdapter;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;
import java.util.TimeZone;

import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.escortvan.Location;

public class MarkPOIOnMap implements InfoWindowAdapter, EscortVanOnParseListener {

  private static final String RACK = "Rack";
  private static final String LATITUDE = "Latitude";
  private static final String LONGITUDE = "Longitude";
  private static final String COUNT = "Name";

  protected boolean showBikeRacks;
  protected boolean showEscortVan;
  protected boolean showBlueLights;
  protected boolean showParkingLots;
  protected boolean showChargePoints;
  protected boolean showZipcar;

  private ArrayList<Rack> racks;
  private ArrayList<LatLng> blueLights;
  private ArrayList<Location> escortVan;
  private ArrayList<LatLng> parkingLots;
  private ArrayList<String> parkingLotsName;
  private ArrayList<String> parkingLotsSubtitle;
  private ArrayList<LatLng> chargePoints;
  private ArrayList<String> chargePointsName;
  private ArrayList<String> chargePointsSubtitle;
  private ArrayList<LatLng> Zipcar;
  private ArrayList<String> ZipcarName;
  private ArrayList<String> ZipcarSubtitle;
  private ArrayList<PolylineOptions> currentRoute;
  private ArrayList<Polyline> savedRoute;
  private MarkerOptions currentBuilding;


  private GoogleMap map;
  private Context context;
  Marker bikeMarker;
  private Bitmap rack;
  private Bitmap.Config config;
  private int size;

  public MarkPOIOnMap(Context context, GoogleMap map) {
     this.map = map;
     this.context = context;
     map.setInfoWindowAdapter(this);
     configureBikeRacks();
     configureBlueLights();
     configureEscortVan();
     configureParkingLots();
     configureChargePoints();
     configureZipcar();
  }

  public void reprintMap() {
     map.clear();
     if (showBikeRacks) {
        for (Rack r : racks) {
           MarkerOptions marker = new MarkerOptions();
           marker.position(r.getLocation());
           marker.title("");
           if (r.getCount() == -1) {
              marker.title("-1");
              Bitmap res = Bitmap.createScaledBitmap(
                    BitmapFactory.decodeResource(context.getResources(), R.drawable.brs),
                    size, size, false);
              marker.icon(BitmapDescriptorFactory.fromBitmap(res));
           } else {
              marker.icon(BitmapDescriptorFactory
                          .fromBitmap(drawTextToBitmap(r.getCount() + "")));
           }
           r.setMarker((map.addMarker(marker)));
        }
        if (FileManager.getBikeLocation(context) != null) {
           MarkerOptions marker = new MarkerOptions();
           marker.position(FileManager.getBikeLocation(context));
```

```
            marker.title("My Bike");
            Bitmap res = Bitmap.createScaledBitmap(
                    BitmapFactory.decodeResource(context.getResources(), R.drawable.bike), size,
                    size, false);
            marker.icon(BitmapDescriptorFactory.fromBitmap(res));
            bikeMarker = map.addMarker(marker);
        }
    }
    if (showBlueLights) {
        for (LatLng l : blueLights) {
            MarkerOptions marker = new MarkerOptions();
            marker.position(l);
            map.addMarker(marker);
        }
    }
    if (showParkingLots) {
        int i = 0;
        for (LatLng l : parkingLots) {
            MarkerOptions marker = new MarkerOptions();
            marker.position(l);
            marker.title(parkingLotsName.get(i));
            marker.snippet(parkingLotsSubtitle.get(i));
            map.addMarker(marker);
            i++;
        }
    }
    if (showChargePoints) {
        int i = 0;
        for (LatLng l : chargePoints) {
            MarkerOptions marker = new MarkerOptions();
            marker.position(l);
            marker.title(chargePointsName.get(i));
            marker.snippet(chargePointsSubtitle.get(i));
            map.addMarker(marker);
            i++;
        }
    }

    if (showZipcar) {
        int i = 0;
```

```
      for (LatLng l : Zipcar) {
        MarkerOptions marker = new MarkerOptions();
        marker.position(l);
        marker.title(ZipcarName.get(i));
        marker.snippet(ZipcarSubtitle.get(i));
        map.addMarker(marker);
        i++;
      }
    }

    if (showEscortVan) {
      if (escortVan != null) {
        for (Location l : escortVan) {
          MarkerOptions marker = new MarkerOptions();
          marker.snippet(
              timeUntilNextVan(l.getStartTime(), l.getEndTime(), l.getInterval()));
          marker.position(new LatLng(l.getLat(), l.getLng()));
          marker.title("Escort Van");
          map.addMarker(marker);
        }
        if (escortVan.size() == 0) {
          Toast.makeText(context, "The escort van isn't running now.", Toast.LENGTH_SHORT)
            .show();
        }
      }
    }
    savedRoute = new ArrayList<Polyline>();
    if (currentRoute != null) {
      for (PolylineOptions p : currentRoute) {
        savedRoute.add(map.addPolyline(p));
      }
    }
    if (currentBuilding != null) {
      map.addMarker(currentBuilding);
    }
  }

  @Override
  public View getInfoContents(Marker marker) {
    View v = View.inflate(context, R.layout.poi_info_window, null);
    TextView tv = (TextView) v.findViewById(R.id.title);
    TextView sub = (TextView) v.findViewById(R.id.subtitle);
    tv.setText(marker.getTitle());
    sub.setText(marker.getSnippet());
    if (marker.getTitle() != null && marker.getTitle().equals("-1")) {
        tv.setText("Bike Repair Station");
    }
    else if(marker.getTitle() != null && marker.getTitle().contains("Spots")) {
        tv.setVisibility(View.GONE);
        sub.setVisibility(View.GONE);
    }
    Log.i("TEST TITLE", "title: " + tv.getText());
    Log.i("TEST SUBTITLE", "subtitle: " + sub.getText());
    return v;
}

@Override
public View getInfoWindow(Marker marker) {
    return null;
}

public ArrayList<Rack> getBRS() {
    ArrayList<Rack> brs = new ArrayList<Rack>();
    for (Rack r : racks) {
        if (r.getCount() == -1) {
            brs.add(r);
        }
    }
    return brs;
}

private void configureBikeRacks() {
    racks = new ArrayList<Rack>();
    size = dpToPx(32);
    rack = Bitmap.createScaledBitmap(
            BitmapFactory.decodeResource(context.getResources(), R.drawable.rack), size, size,
            false);

    config = rack.getConfig();
    if (config == null) {
        config = Bitmap.Config.ARGB_8888;
    }

    try {
        XmlResourceParser parser = context.getResources().getXml(R.xml.racks);
        int state = parser.getEventType();

        while (state != XmlPullParser.END_DOCUMENT) {
            if (state == XmlPullParser.START_TAG && parser.getName().equals(RACK)) {
                double lat = -1;
                double lng = -1;
                int count = -1;
                while (!(parser.getName().equals(RACK) && state == XmlPullParser.END_TAG)) {
                    if (state == XmlPullParser.START_TAG && parser.getName().equals(LATITUDE)) {
                        parser.next();
                        lat = Double.parseDouble(parser.getText());
                    } else if (state == XmlPullParser.START_TAG &&
                            parser.getName().equals(LONGITUDE)) {
                        parser.next();
                        lng = Double.parseDouble(parser.getText());
                    } else if (state == XmlPullParser.START_TAG &&
                            parser.getName().equals(COUNT)) {
                        parser.next();
                        if (!parser.getText().equals("BRS")) {
                            count = Integer.parseInt(parser.getText());
                        }
                    }
                    state = parser.next();
                }
                racks.add(new Rack(lat, lng, count));
            }
            state = parser.next();
        }
    } catch (XmlPullParserException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
    private void configureBlueLights() {
        int[] latitudeArray;
        int[] longitudeArray;

        blueLights = new ArrayList<LatLng>();

        latitudeArray = context.getResources().getIntArray(R.array.BlueAlertLatitude_Array);
        longitudeArray = context.getResources().getIntArray(R.array.BlueAlertLongitude_array);
        for (int i = 0; i < latitudeArray.length; i++) {
            LatLng requestedPoint =
                    new LatLng(latitudeArray[i] / 1000000.0, longitudeArray[i] / 1000000.0);
            blueLights.add(requestedPoint);
        }
    }

    private void configureParkingLots() {
        int[] latitudeArray;
        int[] longitudeArray;
        String[] parkingLotsNameArray;
        String[] parkingLotsSubtitleArray;

        parkingLots = new ArrayList<LatLng>();
        parkingLotsName = new ArrayList<String>();
        parkingLotsSubtitle = new ArrayList<String>();

        latitudeArray = context.getResources().getIntArray(R.array.ParkingLotLatitude_Array);
        longitudeArray = context.getResources().getIntArray(R.array.ParkingLotLongitude_Array);
        parkingLotsNameArray = context.getResources().getStringArray(R.array.parking_lot_array);
        parkingLotsSubtitleArray =
context.getResources().getStringArray(R.array.parking_lot_subtitle_array);
        for(int i = 0; i < latitudeArray.length; i++){
            LatLng requestedPoint = new LatLng(latitudeArray[i] / 1000000.0, longitudeArray[i] / 1000000.0);
            parkingLotsName.add(parkingLotsNameArray[i]);
            parkingLotsSubtitle.add(parkingLotsSubtitleArray[i]);
            parkingLots.add(requestedPoint);
        }
    }
    private void configureChargePoints() {
        int[] latitudeArray;
        int[] longitudeArray;
        String[] chargePointsNameArray;
        String[] chargePointsSubtitleArray;

        chargePoints = new ArrayList<LatLng>();
        chargePointsName = new ArrayList<String>();
        chargePointsSubtitle = new ArrayList<String>();

        latitudeArray = context.getResources().getIntArray(R.array.ChargePointsLatitude_Array);
        longitudeArray = context.getResources().getIntArray(R.array.ChargePointsLongitude_Array);
        chargePointsNameArray = context.getResources().getStringArray(R.array.charge_points_array);
        chargePointsSubtitleArray =
context.getResources().getStringArray(R.array.charge_points_subtitle_array);
        for(int i = 0; i < latitudeArray.length; i++){
          LatLng requestedPoint = new LatLng(latitudeArray[i] / 1000000.0, longitudeArray[i] / 1000000.0);
          chargePointsName.add(chargePointsNameArray[i]);
          chargePointsSubtitle.add(chargePointsSubtitleArray[i]);
          chargePoints.add(requestedPoint);
        }
      }

      private void configureZipcar() {
        int[] latitudeArray;
        int[] longitudeArray;
        String[] ZipcarNameArray;
        String[] ZipcarSubtitleArray;

        Zipcar = new ArrayList<LatLng>();
        ZipcarName = new ArrayList<String>();
        ZipcarSubtitle = new ArrayList<String>();

        latitudeArray = context.getResources().getIntArray(R.array.Zipcar_latitude);
        longitudeArray = context.getResources().getIntArray(R.array.Zipcar_longitude);
        ZipcarNameArray = context.getResources().getStringArray(R.array.Zipcar_location_title);
        ZipcarSubtitleArray = context.getResources().getStringArray(R.array.Zipcar_location_subtitle);
        for(int i = 0; i < latitudeArray.length; i++){
          LatLng requestedPoint = new LatLng(latitudeArray[i] / 1000000.0, longitudeArray[i] / 1000000.0);
          ZipcarName.add(ZipcarNameArray[i]);
          ZipcarSubtitle.add(ZipcarSubtitleArray[i]);
          Zipcar.add(requestedPoint);
        }
```

```
}

private void configureEscortVan() {
   new DownloadVanData(this).execute();
}

Bitmap drawTextToBitmap(String gText) {
   float scale = context.getResources().getDisplayMetrics().density;
   Bitmap bitmap = rack.copy(config, true);

   Canvas canvas = new Canvas(bitmap);
   Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
   paint.setColor(Color.WHITE);
   paint.setTextSize((int) (14 * scale));
   paint.setShadowLayer(1f, 0f, 1f, Color.WHITE);
   Rect bounds = new Rect();
   paint.getTextBounds(gText, 0, gText.length(), bounds);
   int x = (bitmap.getWidth() - bounds.width()) / 2;
   int y = (bitmap.getHeight() + bounds.height()) / 2;

   canvas.drawText(gText, x, y, paint);

   return bitmap;
}

int dpToPx(float dp) {
   Resources resources = context.getResources();
   DisplayMetrics metrics = resources.getDisplayMetrics();
   return (int) (dp * (metrics.densityDpi / 160f));
}

@Override
public void onParseSchedule(String message, String availability,
                     ArrayList<Location> locations) {
   this.escortVan = locations;
   if (showEscortVan) {
      reprintMap();
   }
}
```

```
public void setShowBikeRacks(boolean showBikeRacks) {
   this.showBikeRacks = showBikeRacks;
}

public void setShowEscortVan(boolean showEscortVan) {
   this.showEscortVan = showEscortVan;
}

public void setShowBlueLights(boolean showBlueLights) {
   this.showBlueLights = showBlueLights;
}

public void setShowParkingLots(boolean showParkingLots) {
   this.showParkingLots = showParkingLots;
}

public void setShowChargePoints(boolean showChargePoints) {
   this.showChargePoints = showChargePoints;
}

public void setShowZipcar(boolean showZipcar) {
   this.showZipcar = showZipcar;
}

public ArrayList<Polyline> getCurrentRoute() {
   return savedRoute;
}

public void setCurrentRoute(ArrayList<PolylineOptions> currentRoute) {
   this.currentRoute = currentRoute;
   reprintMap();
}

public void setCurrentBuilding(MarkerOptions currentBuilding) {
   this.currentBuilding = currentBuilding;
}

String timeUntilNextVan(String startTime, String endTime, int[] interval) {
   if (startTime == null || endTime == null) {
      return "Ooops. Not sure when the van will be here next.";
```

```
        }
        Calendar currentTime =
            Calendar.getInstance(TimeZone.getTimeZone("America/Los_Angeles"), Locale.US);
        int currMin = currentTime.get(Calendar.MINUTE);
        int currHour = currentTime.get(Calendar.HOUR_OF_DAY);
        int currDay = currentTime.get(Calendar.DAY_OF_WEEK);

        Calendar startCal = Calendar.getInstance();
        Calendar endCal = Calendar.getInstance();
        SimpleDateFormat format = new SimpleDateFormat("hh:mmaa", Locale.US);

        try {
          startCal.setTime(format.parse(startTime));
          endCal.setTime(format.parse(endTime));
        } catch (ParseException e) {
          e.printStackTrace();
        }

        if (currHour >= startCal.get(Calendar.HOUR_OF_DAY) ||
          (currHour == endCal.get(Calendar.HOUR_OF_DAY) && currMin < interval[0]) &&
          currDay != Calendar.FRIDAY && currDay != Calendar.SATURDAY) {
          int minutes = 60;
          for (int i : interval) {
            int diff = i - currMin;
            if (diff < 0) { //if it is negative, then wrap to the next hour;
              diff += 60;
            }
            if (diff < minutes) {
              minutes = diff;
            }
          }
          return "Next escort van scheduled to arrive in " + minutes + " min";
        }
        return "Escort Van runs " + startTime + " - " + endTime + " (Sun - Thurs)";
    }
}
```

Zipcar

Zipcar.java (created by Ryan Lin)

```
package edu.calpoly.its.gateway.zipcar;

/*
 * The main page for events.  Will import the high-level xml feed of event feeds.
 * This is the page that is displayed upon pressing the "Events" page on the main menu.
 */

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.FrameLayout;
import android.widget.ListView;
import android.widget.Toast;

import com.flurry.android.FlurryAgent;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.escortvan.Location;
import edu.calpoly.its.gateway.map.CalPolyMap;
import edu.calpoly.its.gateway.map.DownloadVanData;
import edu.calpoly.its.gateway.recreation.*;
import android.support.v4.app.Fragment;

public class Zipcar extends BaseActivity {

  //Main xml feed
  private final String feedlist = "http://m.calpoly.edu";

  //Add new feeds here!
  //private final ArrayList<String> listOfFeedNames = new ArrayList<String>();
  //ArrayList<String> listOfZipcarURLs;
  ArrayList<String> listOfZipcarNames;
  ArrayList<Location> locations;
  private Activity activity;
  String reserveURL;
  private Context context;

  //Caching xml file in internal storage
  //private final String filename = "zipcarfeedlist.xml";

  // --Commented out by Inspection (7/8/14, 1:43 PM):private static final int MENU1 = Menu.FIRST;

  //Initializing the list from strings.xml
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    context = getApplicationContext();
    //activity = getBaseActivity();
    //ArrayList<Location> locations;
    setContentView(R.layout.activity_zipcar);

    actionBar.setTitle("Zipcar");

    if (!isOnline(this)) {
        displayMessage(this, "No internet connection");
        return;
    }

    createProgressDialog();
    progressDialog.show();
    new loadingTask().execute(feedlist);

  }
  /*public View onCreateView(LayoutInflater inflater, ViewGroup container,
                  Bundle savedInstanceState) {

  }*/
  private class loadingTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {

        String[] zipcarNameArray;
        //String[] zipcarURLArray;

        listOfZipcarNames = new ArrayList<String>();
        reserveURL = context.getResources().getString(R.string.Zipcar_reserveURL);
        //listOfZipcarURLs = new ArrayList<String>();

        zipcarNameArray = context.getResources().getStringArray(R.array.Zipcar_names);
        //zipcarURLArray = context.getResources().getStringArray(R.array.Zipcar_url);
        for(int i = 0; i < zipcarNameArray.length; i++) {
            listOfZipcarNames.add(zipcarNameArray[i]);
            //Log.i("Zipcar: configureZipcar", listOfZipcarNames.get(i));
            //listOfZipcarURLs.add(zipcarURLArray[i]);
        }

        /*if (listOfFeedNames.size() != 0) {
            //Log.d("DEBUG", "Rewriting cache file");
            writeCacheFile();
        }
        listOfFeedNames.clear();
        listOfFeedURLs.clear();
        InputStream inputStream;
```

```
if (!checkCache()) {
   if (isOnline()) {
      writeCacheFile();
   }
} else if (ifReCache()) {
   if (isOnline()) {
      writeCacheFile();
   }
}
inputStream = readCacheFile();

try {
   XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
   factory.setNamespaceAware(true);
   XmlPullParser xpp = factory.newPullParser();
   xpp.setInput(inputStream, null);
   int eventType = xpp.getEventType();
   while (eventType != XmlPullParser.END_DOCUMENT) {
      switch (eventType) {
         case XmlPullParser.START_TAG:
            if (xpp.getName().equalsIgnoreCase("title"))
               listOfFeedNames.add(xpp.nextText());
            else if (xpp.getName().equalsIgnoreCase("link"))
               listOfFeedURLs.add(xpp.nextText());
            break;
         case XmlPullParser.END_TAG:
            break;
         default:
            break;
      }
      eventType = xpp.next();
   }

} catch (MalformedURLException e1) {
   e1.printStackTrace();
} catch (IOException e) {
   e.printStackTrace();
} catch (XmlPullParserException e) {
   e.printStackTrace();
```

```
        }*/

        return "";
    }

    @Override
    protected void onPostExecute(String s) {
        progressDialog.dismiss();
        //Set the Adapter
        ZipcarSectionAdapter adapter =
                new ZipcarSectionAdapter(Zipcar.this, listOfZipcarNames, false);
        ListView recFeedsList = (ListView) findViewById(R.id.mainList);
        setAnimationAdapter(adapter, recFeedsList);

        recFeedsList.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                if (position == listOfZipcarNames.indexOf("Register")) {
                    startActivity(new Intent(Zipcar.this, RegisterZipcar.class));
                } else if (position == listOfZipcarNames.indexOf("Reserve")) {
                    startActivity(BaseActivity.newWebIntent(listOfZipcarNames.get(position),
                            reserveURL,
                            SINGLE_PAGE));
                    //startActivity(new Intent(Zipcar.this, Intramurals.class));
                } else {
                    startActivity(new Intent(Zipcar.this, ZipcarLocation.class));
                }
            }
        });

        if (listOfZipcarNames.size() == 0) {
            Toast connectionError = Toast.makeText(getApplicationContext(),
                    "An internet connection is required to view this feed",
                    Toast.LENGTH_LONG);
            connectionError.setGravity(Gravity.CENTER, 0, 0);
            connectionError.show();
        }
    }

}
```

```
// --Commented out by Inspection START (7/8/14, 1:43 PM):
//// --Commented out by Inspection START (7/8/14, 1:43 PM):
////       private class CacheTask extends AsyncTask<String, Void, String> {
////
////                  @Override
////                  protected String doInBackground(String... params) {
// --Commented out by Inspection STOP (7/8/14, 1:43 PM)
//                               writeCacheFile();
//                               return "";
//                    }
//         }
// --Commented out by Inspection STOP (7/8/14, 1:43 PM)

    /*
     *Opens the xml feed link and writes it into internal storage. Cache
     *file is deleted once the application has been uninstalled.
     */
    /*private void writeCacheFile() {
        this.deleteFile(filename);
        try {
            int nib;
            InputStream reader = (new URL(feedlist)).openStream();
            FileOutputStream writeFile = openFileOutput(filename, Context.MODE_PRIVATE);
            while ((nib = reader.read()) != -1) {
                writeFile.write(nib);
            }
            writeFile.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //Returns a InputSource so that SAXHelper can parse xml feed
    private InputStream readCacheFile() {
        FileInputStream readFile = null;
        try {
            readFile = openFileInput(filename);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return readFile;
    }*/

    //Returns true if device is connected, otherwise false
    private boolean isOnline() {
        ConnectivityManager cm =
                (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

        return cm.getActiveNetworkInfo() != null &&
                cm.getActiveNetworkInfo().isConnectedOrConnecting();
    }

    //Returns true the cache need to be updated, otherwise false
    /*private boolean ifReCache() {
        File file = getBaseContext().getFileStreamPath(filename);
        if (file.exists()) {
            Date lastModified = new Date(file.lastModified());
            Date currDate = Calendar.getInstance().getTime();
            int updateCache = 604800000;
            if ((currDate.getTime() - lastModified.getTime()) >= updateCache) {
                return true;
            }
        }
        return false;
    }*/

    //Return true if cache exist, otherewise false
    /*private boolean checkCache() {
        File file = getBaseContext().getFileStreamPath(filename);
        return file.exists();
    }*/

    public void onStart() {
        super.onStart();
        FlurryAgent.logEvent(this.getClass().getSimpleName());
    }
}
```

RegisterZipcar.java (created by Ryan Lin)

```
package edu.calpoly.its.gateway.zipcar;

/*
 * The main page for events.  Will import the high-level xml feed of event feeds.
 * This is the page that is displayed upon pressing the "Events" page on the main menu.
 */

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.Toast;

import com.flurry.android.FlurryAgent;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;

public class RegisterZipcar extends BaseActivity {

  //Main xml feed
  private final String feedlist = "http://m.calpoly.edu";

  //Add new feeds here!
  ArrayList<String> listOfRegisterNames;
  ArrayList<String> listOfRegisterURLs;
  Context context;

  //Caching xml file in internal storage
  //private final String filename = "fitnessfeed.xml";

  // --Commented out by Inspection (7/8/14, 1:43 PM):private static final int MENU1 = Menu.FIRST;

  //Initializing the list from strings.xml
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register_zipcar);
    context = getApplicationContext();
    createProgressDialog();
    progressDialog.show();
    new loadingTask().execute(feedlist);

    actionBar.setTitle("Register");
  }

  private class loadingTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {

      String[] RegisterNameArray;
      String[] RegisterURLArray;

      listOfRegisterNames = new ArrayList<String>();
      listOfRegisterURLs = new ArrayList<String>();
```

```
RegisterNameArray = context.getResources().getStringArray(R.array.Zipcar_register);
RegisterURLArray = context.getResources().getStringArray(R.array.Zipcar_registerURL);
for(int i = 0; i < RegisterNameArray.length; i++) {
   listOfRegisterNames.add(RegisterNameArray[i]);
   Log.i("RegisterZipcar: configureRegisterZipcar", listOfRegisterNames.get(i));
   listOfRegisterURLs.add(RegisterURLArray[i]);
}

/*if (listOfRegisterNames.size() != 0) {
   //Log.d("DEBUG", "Rewriting cache file");
   writeCacheFile();
}
listOfRegisterNames.clear();
listOfRegisterURLs.clear();
InputStream inputStream;

if (!checkCache()) {
   if (isOnline()) {
      writeCacheFile();
   }
} else if (ifReCache()) {
   if (isOnline()) {
      writeCacheFile();
   }
}
inputStream = readCacheFile();*/

/*try {
   XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
   factory.setNamespaceAware(true);
   XmlPullParser xpp = factory.newPullParser();
   xpp.setInput(inputStream, "iso-8859-1");
   int eventType = xpp.getEventType();
   while (eventType != XmlPullParser.END_DOCUMENT) {
      switch (eventType) {
         case XmlPullParser.START_TAG:
            if (xpp.getName().equalsIgnoreCase("title"))
               listOfFeedNames.add(xpp.nextText());
            else if (xpp.getName().equalsIgnoreCase("link"))
               listOfFeedURLs.add(xpp.nextText());
```

```
                break;
            case XmlPullParser.END_TAG:
                break;
            default:
                break;
        }
        eventType = xpp.next();
    }

} catch (MalformedURLException e1) {
    e1.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (XmlPullParserException e) {
    e.printStackTrace();
}*/

return "";
}

@Override
protected void onPostExecute(String s) {
    progressDialog.dismiss();
    //setListAdapter(new RecreationSectionAdapter(Fitness.this, listOfFeedNames, true));
    ListView registerList = (ListView) findViewById(R.id.mainList);
    setAnimationAdapter(new ZipcarSectionAdapter(RegisterZipcar.this, listOfRegisterNames, true),
            registerList);

    registerList.setOnItemClickListener(new OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> arg0, View v, int position, long id) {
            startActivity(BaseActivity.newWebIntent(listOfRegisterNames.get(position),
                    listOfRegisterURLs.get(position),
                    SINGLE_PAGE));
        }          });

    if (listOfRegisterNames.size() == 0) {
        Toast connectionError = Toast.makeText(getApplicationContext(),
                "An internet connection is required to view this feed",
                    Toast.LENGTH_LONG);
                connectionError.setGravity(Gravity.CENTER, 0, 0);
                connectionError.show();
            }
        }

    }

            /*@Override
    public void onListItemClick(ListView l, View v, int position, long id) {
                        if (position == listOfFeedNames.indexOf("Personal Training")) {
                                    Intent recWeb = new
Intent("edu.calpoly.its.gateway.EVENTSWEBLINK").putExtra("dispLink", listOfFeedURLs.get(position));
                                    recWeb.putExtra("feedName", listOfFeedNames.get(position));
                                    startActivity(recWeb);
                        }
                        else {
                                    Intent el = new
Intent("edu.calpoly.its.gateway.EVENTSLISTING").putExtra("feedSelection",
listOfFeedURLs.get(position));
                                    el.putExtra("feedName", listOfFeedNames.get(position));
                                    startActivity(el);
                        }
            }*/

// --Commented out by Inspection START (7/8/14, 1:43 PM):
//// --Commented out by Inspection START (7/8/14, 1:43 PM):
////        private class CacheTask extends AsyncTask<String, Void, String> {
////
////                    @Override
////                    protected String doInBackground(String... params) {
// --Commented out by Inspection STOP (7/8/14, 1:43 PM)
//                                  writeCacheFile();
//                                  return "";
//                      }
//          }
// --Commented out by Inspection STOP (7/8/14, 1:43 PM)

    /*
     *Opens the xml feed link and writes it into internal storage. Cache
```

```
 *file is deleted once the application has been uninstalled.
*/
/*private void writeCacheFile() {
   this.deleteFile(filename);
   try {
      int nib;
      InputStream reader = (new URL(feedlist)).openStream();
      FileOutputStream writeFile = openFileOutput(filename, Context.MODE_PRIVATE);
      while ((nib = reader.read()) != -1) {
         writeFile.write(nib);
      }
      writeFile.close();
   } catch (Exception e) {
      e.printStackTrace();
   }
}

//Returns a InputSource so that SAXHelper can parse xml feed
private InputStream readCacheFile() {
   FileInputStream readFile = null;
   try {
      readFile = openFileInput(filename);
   } catch (Exception e) {
      e.printStackTrace();
   }
   return readFile;
}*/

//Returns true if device is connected, otherwise false
private boolean isOnline() {
   ConnectivityManager cm =
         (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

   return cm.getActiveNetworkInfo() != null &&
         cm.getActiveNetworkInfo().isConnectedOrConnecting();
}

//Returns true the cache need to be updated, otherwise false
/*private boolean ifReCache() {
   File file = getBaseContext().getFileStreamPath(filename);
```

```
        if (file.exists()) {
            Date lastModified = new Date(file.lastModified());
            Date currDate = Calendar.getInstance().getTime();
            int updateCache = 604800000;
            if ((currDate.getTime() - lastModified.getTime()) >= updateCache) {
                return true;
            }
        }
        return false;
    }

    //Return true if cache exist, otherewise false
    private boolean checkCache() {
        File file = getBaseContext().getFileStreamPath(filename);
        return file.exists();
    }*/

    @Override
    public void onBackPressed() {
        finish();
//                      overridePendingTransition(BaseActivity.GENERAL_BACK_ANIM_IN,
BaseActivity.GENERAL_BACK_ANIM_OUT);
    }

    public void onStart() {
        super.onStart();
        FlurryAgent.logEvent(this.getClass().getSimpleName());
    }
}
```

ZipcarLocation.java (created by Ryan Lin)

```
package edu.calpoly.its.gateway.zipcar;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.afollestad.materialdialogs.MaterialDialog;
import com.flurry.android.FlurryAgent;
import com.nispok.snackbar.Snackbar;
import com.nispok.snackbar.SnackbarManager;
import com.nispok.snackbar.enums.SnackbarType;
import com.telly.mrvector.MrVector;

import java.util.ArrayList;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;
import edu.calpoly.its.gateway.map.CalPolyMap;
import edu.calpoly.its.gateway.map.DownloadVanData;
import edu.calpoly.its.gateway.map.EscortVanAdapter;
import edu.calpoly.its.gateway.map.EscortVanOnParseListener;

public class ZipcarLocation extends BaseActivity
{
    static Context context;
    static ArrayList<Integer> listOfZipcarLatitudes;
    static ArrayList<Integer> listOfZipcarLongitudes;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        context = getApplicationContext();
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_zipcar_location);
        if (savedInstanceState == null)
        {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new ZipcarFragment())
                    .commit();
        }
    }
    public static class ZipcarFragment extends Fragment// implements EscortVanOnParseListener
    {
        private Activity activity;
        //private ArrayList<Location> locations;
        //private ArrayList<Integer> latitude;
        //private ArrayList<Integer> longitude;
        private ListView locationList;
        private FrameLayout container;
        public ZipcarFragment()
        {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState)
        {
            activity = getActivity();
            View rootView = inflater.inflate(R.layout.fragment_zipcar, container, false);
            if (!isOnline(activity)) {
                //onBackPressed();
                displayMessage(activity, "No network connection");
            }

            ((ZipcarLocation) activity).createProgressDialog();
            ((ZipcarLocation) activity).progressDialog.show();
            //new DownloadVanData(this).execute();

            actionBar.setTitle("Zipcar");
            locationList = ((ListView) rootView.findViewById(R.id.zipcar_locations));
            /*locationList.setOnItemClickListener(new AdapterView.OnItemClickListener()
            {
                @Override
                public void onItemClick(AdapterView<?> adapterView, View view, int i, long l)
                {*/
                    int[] zipcarLatitudeArray;
                    int[] zipcarLongitudeArray;
                    listOfZipcarLatitudes = new ArrayList<Integer>();
                    listOfZipcarLongitudes = new ArrayList<Integer>();
                    //listOfZipcarURLs = new ArrayList<String>();

                    zipcarLatitudeArray = context.getResources().getIntArray(R.array.Zipcar_latitude);
                    zipcarLongitudeArray = context.getResources().getIntArray(R.array.Zipcar_longitude);
                    //zipcarURLArray = context.getResources().getStringArray(R.array.Zipcar_url);
                    for(int j = 0; j < zipcarLatitudeArray.length; j++) {
                       listOfZipcarLatitudes.add(zipcarLatitudeArray[j]);
                       listOfZipcarLongitudes.add(zipcarLongitudeArray[j]);
                       //Log.i("Zipcar: configureZipcar", listOfZipcarNames.get(i));
                       //listOfZipcarURLs.add(zipcarURLArray[i]);
                    }
                    //if (locations.size() > 0)
                    //{
                       //not the header position
                       //if (i != 0)
                       //{
                          //Location loc = locations.get(i - 1); //offset because of the header
                          Intent intent = new Intent();
                          intent.setClass(activity, CalPolyMap.class);
                          intent.putExtra("lat", listOfZipcarLatitudes/*.get(i)*/);
                          intent.putExtra("lng", listOfZipcarLongitudes/*.get(i)*/);
                          intent.putExtra(CalPolyMap.MAP_TYPE, CalPolyMap.TYPE_ZIPCAR);
                          startActivity(intent);
                       //}
                    //}
                /*}
            });*/
            this.container = (FrameLayout) getActivity().findViewById(R.id.container);
            return rootView;
        }
        public void onStart() {
            super.onStart();
            FlurryAgent.logEvent(this.getClass().getSimpleName());
        }

        /*public void onParseSchedule(String message, String availability,
                            ArrayList<Location> locations) {
            //if locations == null, no attached locations
            this.locations = locations;

            LayoutInflater inflater = (LayoutInflater)
    activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            View header = inflater.inflate(R.layout.escortvan_list_header, null);
            ImageView infoImage = (ImageView) header.findViewById(R.id.section_icon);
            infoImage.setImageDrawable(MrVector.inflate(getResources(), R.drawable.vector_info));
            locationList.addHeaderView(header, null, false);

            if (message != null && message.length() > 0) {
                Toast.makeText(activity, message, Toast.LENGTH_LONG).show();
                String[] items = {message};
                ArrayAdapter<String> adapter =
                       new ArrayAdapter<String>(activity, android.R.layout.simple_list_item_1, items);
                locationList.setAdapter(adapter);
            }
            if (availability != null && availability.length() > 0) {
                TextView tv = (TextView) header.findViewById(R.id.section_text);
                tv.setText(availability);
            }
            if (locations != null && locations.size() > 0) {
                locationList.setAdapter(new EscortVanAdapter(activity, locations));
            }
            Snackbar snackbar =
                    Snackbar.with(activity.getApplicationContext()) // context
                          .type(SnackbarType.MULTI_LINE)
                          .text(activity.getResources().getString(R.string.real_time)) // text to be displayed
                          .duration(Snackbar.SnackbarDuration.LENGTH_INDEFINITE)
                          .actionLabel("GOT IT") // action button label
                          .actionColor(Color.YELLOW);// action button label color
            ((EscortVanService) activity).progressDialog.cancel();
            SnackbarManager.show(snackbar, container); // activity where it is displayed
        }*/
    }
    }
```

ZipcarSectionAdapter.java (created by Ryan Lin)

```
package edu.calpoly.its.gateway.zipcar;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;

import edu.calpoly.its.gateway.BaseActivity;
import edu.calpoly.its.gateway.R;

class ZipcarSectionAdapter extends BaseAdapter {
    private final ArrayList<String> data;
    private static LayoutInflater inflater = null;
    private final ArrayList<Integer> listImages;
    // --Commented out by Inspection (7/8/14, 1:43 PM):boolean fitnessSection;

    ZipcarSectionAdapter(Activity a, ArrayList<String> d, boolean registerSection) {
        data = d;
        inflater = (LayoutInflater) a.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        listImages = new ArrayList<Integer>();
        if (registerSection) {
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
        } else {
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            listImages.add(R.drawable.groupexercise);
            /*listImages.add(R.drawable.intramurals);
            listImages.add(R.drawable.polyescapes);*/
        }
    }

    @Override
    public int getCount() {
        return data.toArray().length;
    }

    @Override
    public Object getItem(int position) {
        return position;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    public static class ViewHolder {
        public TextView label;
        public ImageView image;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View vi = convertView;
        ViewHolder holder;
        if (convertView == null) {
            vi = inflater.inflate(R.layout.recsectionentry, null);
            holder = new ViewHolder();
            holder.label = (TextView) vi.findViewById(R.id.title);
            holder.image = (ImageView) vi.findViewById(R.id.thumb);
            vi.setTag(holder);
        } else
            holder = (ViewHolder) vi.getTag();

        BaseActivity.recycleBitmap(holder.image);
```

```
        holder.label.setText(data.get(position));
        holder.image.setImageResource(listImages.get(position));
        return vi;
    }
}
```