CampuSeek

Final Report

Persuasive Social Games

Steve Hanson

Scott Lee

Prince Woodrow

Fernando Salazar

Department of Computer Science and Engineering

Texas A&M University

May 10, 2011

Table of Contents

1 Executive Summary

In a culture proliferated with media and high-tech gadgets, many people have become familiar with the armchair and lost the motivation to exercise. Forty percent of United States adults are now obese – a 14% increase since 1988. This growing trend is a concern in today's society and needs to be addressed. Our proposed project will attempt to reduce obesity through an interactive game that encourages users to be active. The fact that the game will be *fun* will give students the motivation to get the exercise they need.

Our proposed project is a collaborative game, called CampuSeek that promotes exercise and active lifestyles by encouraging students to explore campus. The game is a campus scavenger-hunt that utilizes an Android phone's GPS in conjunction with Facebook. A series of challenges are posted in the form of a picture of a location on campus or a clue or riddle leading the user to a place on campus. Players earn points when they find the desired location, confirmed by their phone's GPS that detects when they are there. We believe this game will have appeal to college students as a way to have fun and compete with their friends while exploring campus. This serves our goal of providing a fun environment to motivate users toward their weight-loss goals.

CampuSeek achieves the weight loss goal in several ways. First of all, the nature of the game encourages weight loss by requiring users to explore and be active on campus. Second, as the user explores campus, caloric consumption and distance travelled are recorded. Users can then analyze charts of this data and compare them with their weight loss goals, which will encourage continued perseverance toward goals. Finally, several features/incentives are unlocked when the user completes certain exercise challenges (e.g. walk for 10 minutes, run ¼ of a mile, etc.). The desire of users to unlock these incentives will push them toward being active. The entertainment factor of the game coupled with the user's desire to achieve results in the collaborative game environment make CampuSeek a viable means to help college students achieve weight loss goals.

The game will be written for the Android and Facebook platforms. Most of the features will be accessible from both platforms, however each has a different focus. Facebook will be the primary medium for collaboration, competition and detailed charts, graphs and summaries. The phone platform will be used primarily during the challenges. Users use the phone to confirm completion of challenges (via GPS) and to access new challenges and use other features during challenges. There will be limited functionality on the phone platform for browsing completed challenges and comparing friend scores.

The expected results are that people who play this game become active, meet new people and are encouraged to live healthier lifestyles.

2 Project background

In a culture proliferated with media and high-tech gadgets, many people have become familiar with the armchair and lost the motivation to exercise. Forty percent of United States adults are now obese – a 14% increase since 1988. This growing trend is a concern in today's society and needs to be addressed. Our proposed project will attempt to reduce obesity through an interactive game that encourages users to be active. The fact that the game will be *fun* will give students the motivation to get the exercise they need.

2.1 Needs statement

There is a need to promote healthy activity among students in a way that is fun, so that participants are more motivated to continue a workout regimen.

2.2 Goals and objectives

The main goal of our project is to promote fitness by providing an exciting way for people to be active. Since many people lack motivation to be active, if people have a way to enjoy being active, they will

likely do so more often and achieve better results.

Here is a list of objectives that our team has taken into consideration:

- The system should track user workout data to encourage and motivate users
- The system should display workout data in a graphical, intuitive format
- The system should allow users to collaborate with others for motivation
- They system should have a game aspect to provide a fun environment for being active
- The system must be fairly easy to set up (less than 10 minutes)
- The mobile application developed should be different than those work out applications that already exist

2.3 Design Constraints and feasibility

The largest constraint on our project will be the battery life provided by the phone while users are playing the game. The distance traveled by the user will be calculated by interpolating the users location based off of the GPS coordinates that are obtained frequently by the Android-based phone. While GPS is known to drain the battery life, it has been tested getting repeated updates for 3 hours of continuous use while only draining 23% of the battery.

A physical constraint that users may encounter is holding the phone in the process of playing CampuSeek. One of the goals of our game is to encourage people to get out and run, walk, and explore their campus. The user may not want to hold the phone in hand while running or walking. We propose (and have included in our budget) to have phone clips that would attach to the users waste to hold the phone. This also leads to an economic restraint that would require the user to purchase a phone clip if they don't already have one and feel that it would be beneficial.

Some phones have GPS and accelerometers, while other phones do not. This will have to be taken into account during the design of the system, as some users' phones may not be compatible with all of the sensors used in the project.

Another constraint is availability of sensors on the phones that are used. Some phones have GPS and accelerometers, while other phones do not. This will have to be taken into account during the design of the system, as some users' phones may not be compatible with all of the sensors used in the project.

Overall, the CampuSeek idea based on our current preliminary design seems to be feasible, pending tests on the battery life of Android phones using GPS. The system will be fairly complex, however, and may be challenging to complete in time. Our team is confident that most of the features will be able to be completed in the allotted time. Three of the features: the hotter/colder feature, the hints feature and route mapping will only be implemented if there is enough extra time during the semester.

2.4 Literature and technical survey

While CampuSeek seems to be a unique idea as a whole, there are several other products that implement some functionality that is similar. Nike+ GPS is an iOS (iPod/iPhone) system that uses workout-tracking. SCVNGER, Tourality, and MoLo are Android applications that use geocaching/GPS scavenger hunt ideas that are similar to some of CampuSeek's. SportsTracker Pro and Endomondo is a social Android application that tracks workouts. None of these related products use the game medium along with health-promoting functionality. Each of these products will now be discussed.

Nike+ GPS [3]

The Nike+ GPS application uses the iOS platform in conjunction with GPS to time, track routes and record distance of mobile workouts (e.g. walking, cycling, skiing, etc.). Data is recorded by the phone's GPS and sent to Nike's own collaborative environment, nikeplus.com, where users can join challenges, set goals and connect with other athletes. Screenshots of the app in action are shown below in Figure 1.

Figure 1: (left) Nike+ GPS application on iPhone., with workout features including collaboration, timing and calorie calculator. (right) GPS workout charting feature shows route taken using Google Maps.

The GPS workout-tracking functionality is similar to and slightly more robust than what will be provided by CampuSeek. The two applications are very different though. CampuSeek is primarily a scavenger-hunt game with health-promoting activity-charting features. Its target audience is people who have trouble finding the motivation to just go outside to workout. The focus of the Nike+ GPS system is entirely on workout charting and is intended for individuals who are already motivated to workout. For this reason, the Nike+ system does not provide workout incentives as the CampuSeek product does. Users looking solely for workout functionalities should consider Nike+ GPS.

Endomondo [4]

Endomondo is a mobile application that provides workout-tracking features similar to Nike+ GPS. Endomondo also features real-time route tracking from external computers and optional heart-rate integration using Bluetooth heart-rate monitors. Endomondo runs on most mobile platforms including iOS, Android and Windows 7 Mobile and also features a collaborative web interface. Like the Nike+ GPS application, Endomondo is a good choice for workout-charting functionality, but lacks the key gaming feature of CampuSeek that provides the fun atmosphere for those lacking motivation.

Tourality

Tourality is a mobile-based scavenger-hunt game that uses phone's GPS in a similar manner as CampuSeek. In this game, challenges are posted for nearby areas and users compete with others to find the destination. Time is important, and users are encouraged to use vehicles to get to their destinations. This veers away from the health focus that CampuSeek aims to have. There is no specific health focus. The incentives for physical activity in the CampuSeek application are what make it unique from other scavenger-hunting games.

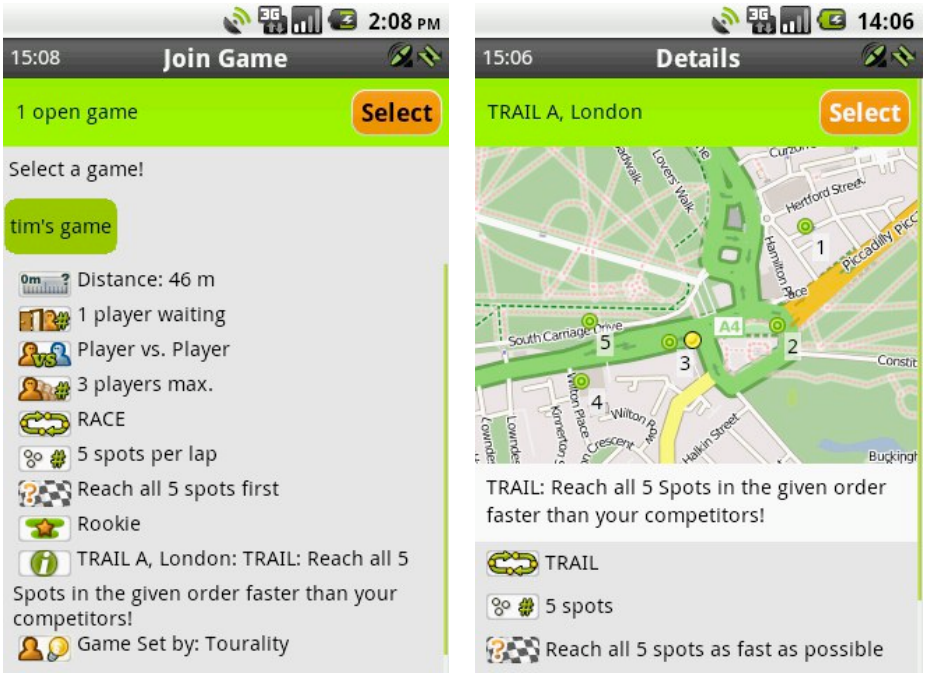


Figure 2: Tourality interface

SCVNGR [5]

SCVNGR is a mobile application for Android that allows the user to engage in Scavenger Hunts using GPS. This application can take pictures and videos with the camera on the device being used. SCVNGR accesses coarse location sources like the cellular network database and the device's GPS to approximate a location for the device. SCVNGR is a game that encourages the user to go out to different stores, restaurants, parks, and other places, while doing challenges and earning points.

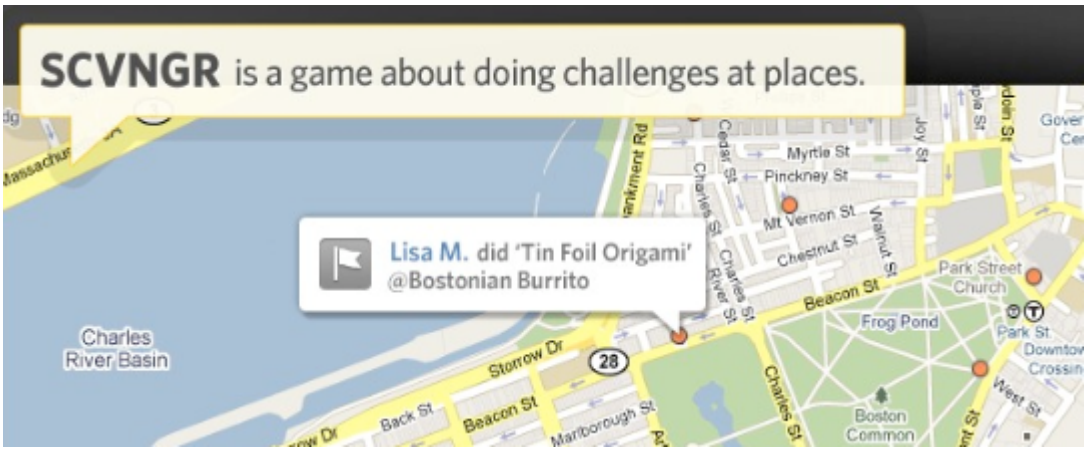


Figure 3: Screenshot of the SCVNGR Android Application [5]

SCVNGR's main purpose is to provide a check-in service for users while allowing them to be a part of a worldwide scavenger hunt. There are currently over 12,000 places that offer rewards for playing this game. The challenges are either user created or created by the businesses. Like some of the other applications discussed, the focus of SCVNGR and CampuSeek are different. While we do encourage the user to go out and explore various places with CampuSeek, our main focus is exhorting the user to be active. SCVNGR does not provide the health incentives or the workout-tracking features that CampuSeek does. Additionally, CampuSeek has a college campus focus, while SCVNGR is worldwide.

SportsTracker Pro [6]

SportsTracker Pro is similar to the NIKE+ GPS application for Android. SportsTracker Pro allows users to compare, train, share and compete in your sport with friends anywhere worldwide. This application offers detailed analysis of pace, altitude, ascents, descents and speed. Users can share their progress on Facebook, Twitter and via email. SportsTracker Pro is predominantly geared toward those who play outdoors sports such as running, cycling, mountain biking, snowboarding, motorcycling, sailing, kayaking, and windsurfing. Its main purpose is to allow those that are participate in outdoor activities to track their course. As with the other workout products discussed previously, SportsTracker Pro lacks the game feature that is the key to making CampuSeek a fun way to become active.

Figure 4: SportsTracker Pro screenshots [6]

Conclusion

None of these related products incorporates the health, gaming, and social combination that CampuSeek does. The goal of CampuSeek is to provide a fun experience for the user so that they will continue healthy activity. Not only will the user be able to track how far they have traveled and how many calories burned, they will also be able to go on a scavenger hunt and compete with friends in the process.

2.5 Evaluation of alternative solutions

After receiving feedback from the professor and several classmates on this project and surveying other technologies, several alternative solutions have come to mind:

- Use accelerometer to record workout information rather than GPS
- Fast-paced challenges where time is a factor
- Weight lifting focus application
- Weight-Watchers type calorie counting application
- Racing style game

The first alternative solution that was heavily considered was to base CampuSeek's core workout recording functionality off of an accelerometer rather than using the GPS hardware. The participant would then wear the phone on their waist and the accelerometer would measure the amount of movement coming from the hip and translate this to a number of steps walked. Essentially, this works in the same fashion as a pedometer. The advantage of this solution compared to the main design is that it could potentially save a large amount of battery life since it will not have to be consistently updating via satellites. A goal of ours is to make this mobile application as energy efficient as possible to ensure the users can spend a decent amount of time playing the game and being active. However, if we were to limit the application to strictly using an accelerometer to track the number of steps taking, it would be difficult to measure the distance the user has traveled and the amount of calories they have burned. We believe this will not encourage a continuation of activity since the users will not have a meaningful representation of

the results they have worked hard to achieve. Whether or not this alternative will be used is contingent on battery-life testing that will be performed using the GPS on Android phones.

Another alternative approach to our gaming application would be to create the challenges in the style similar to that of the show, “The Amazing Race”. Right now our game is a scavenger hunt that users can complete at their own discretion and when they have time. Alternatively, challenges could begin at specific starting times and users or teams would race to complete them first. This is the approach taken by the Tourality application discussed above. One advantage of this style of game is that it could create a more exciting atmosphere as users race in real-time. Also, if users are forced to go quickly, they may engage in more productive activity (e.g. running instead of walking). One disadvantage to this idea is that users may opt to travel by vehicle or other motorized means rather than engage in calorie burning activity, since time is the most important factor. Also, those walking or running would be at a disadvantage to those riding bicycles or in motorized vehicles, creating an unfair environment to those without extra equipment. It would also be less convenient for some users to only be able to complete challenges at specific times. Ultimately we want to promote and help maintain an active lifestyle through gaming applications.

After research, it can be seen that a lot of apps have been developed that keep track of the distance traveled by its users. One possible solution would be to have a mobile application that has different weight lifting exercises for users to try out. It would be available to view on the mobile application, with step-by-step instructions. To incorporate the social aspect, users could upload their own work out regimens to a Facebook application that would display this on the mobile application. The advantage is that it is something different and not common and helps users learn workouts they might have never known about. The disadvantage is that it is not very user interactive and the excitement level that is incorporated with a game just is not there.

Along the lines of changing the intentions of the mobile application, a weight-watchers application could be implemented that allows users to input the meals they eat throughout the day and calculate the number of calories consumed. The Facebook aspect would create a game that awards users points for consuming fewer calories throughout the day (within a healthy boundary). The advantage to this is that it adds a gaming twist to the weight-watchers concept while awarding users for eating fewer calories. The disadvantage is that users might consume less than the healthy number of calories to consume a day just to receive more points and be ranked higher than their peers. Also, users could enter fraudulent caloric consumption to achieve higher scores. This would lead to an inverse effect of what we are striving for.

The final alternative design would potentially be to have a racing style game. Two users could set up a “match” via Facebook and then at an agreed upon time, they could start a race for a set distance. The winner would be whoever completed this task first. The GPS on the phone would verify the distance they have run and notify the user when they have reached the set distance. The advantage of this game is that it promotes an active lifestyle with a competitive edge and is something that is not currently in any mobile platform market, however this would be geared more toward those who already work out and not target newcomers who need motivation as much as the current CampuSeek design.

Overall, we feel very confident that the CampuSeek idea will best perform the goals that we seek to address, primarily that we develop a fun way for users to achieve weight loss goals while maintaining motivation.

3 Final design

3.1 System description

The design of our system is broken into two main components: a Facebook application and an Android application. These serve as the two mediums through which the user accesses the system. Both of these platforms interface with a central database that stores user and challenge information. GPS on the mobile phone is leveraged in this system from within the Android environment to provide challenge and workout statistics. Finally, Facebook provides detailed health statistics for users to use to examine their performance. A graphical representation of this system can be seen below in Figure 5, below.

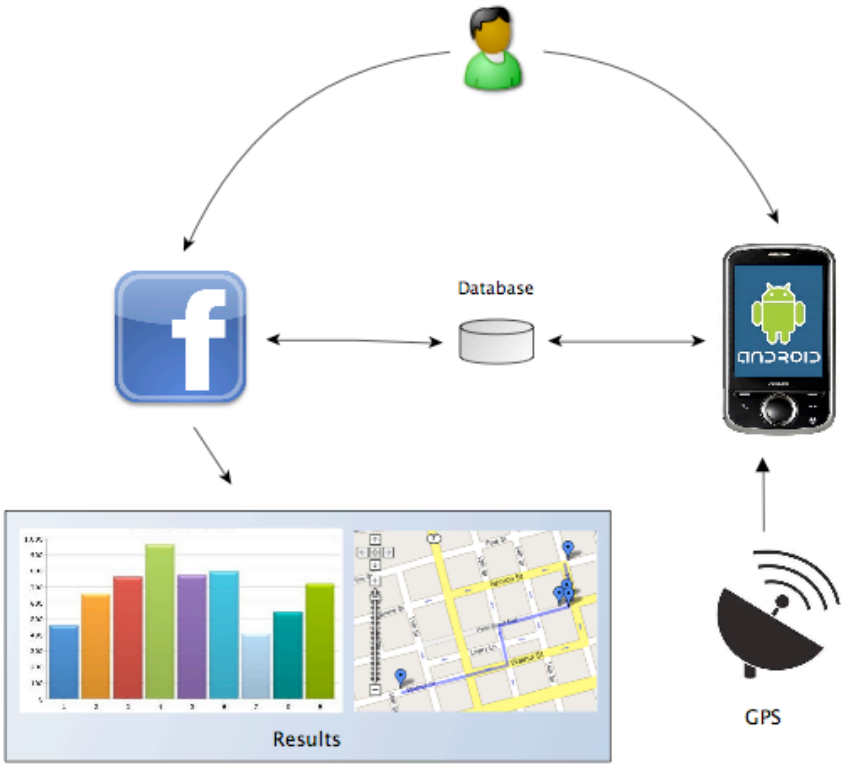


Figure 5: modular description of system

As seen in Figure 6, the design of our mobile Android application relies upon the GPS hardware that is incorporated with the Android-based mobile phone that we will be using. Users will access various scavenger challenges through the CampuSeek application on their phone and venture out throughout their college campus to find the desired objects. Once at the location, the GPS will confirm they are in the correct place and award the user with a predetermined number of points. The Android application is being developed in the Java programming language using an Android emulator and programming API.

The second module, the Facebook web application, will be a “home base” for users to view progress and compare their results to their friends. Users can choose to “follow” their Facebook friends,

which results in their progress being displayed in easily accessible ways, so that others may compare their progress to that of their friends. The user will also have the option to view their activity statistics through the Facebook web application to further motivate and promote an active lifestyle. These statistics will consist of information on how many calories the user is burning, in addition to distance travelled. These data will be displayed graphically so that users can view trends in their workout performance and compare it with their friends. Each of the modules that make up the system will now be described.

3.2 Complete module-wise specifications

As mentioned previously, the system is composed of two main modules: Facebook application and the Android application. The Facebook application module includes health charting features as well as organized challenge viewing. The Android application module incorporates GPS and gameplay features. Both modules coordinate with a centralized database to store and access information about the users and challenges.

Android Application

Android offers a custom plug-in, Android Development Tools (ADT), for the Eclipse IDE that is designed to give a powerful integrated environment in which developers can build android applications. ADT allows you to quickly set up new Android projects using Eclipse, create an application UI, and debug android applications using the Android SDK tools. ADT also allows developers to export versions of their Android application in the easily shared APK Android format. A virtual mobile device emulator is also included within the Android SDK. This emulator allows developers to develop, test, and prototype android applications without using a physical device. The emulator is very useful because it mimics all of the hardware and software features of a typical mobile device. The emulator cannot make phone calls, but that is not a downside in our case because our application does not use that feature. The emulator provides a variety of navigation and control keys that you press by using your mouse or keyboard. To model and test applications more easily, the emulator utilizes Android Virtual Device configurations, which allow you to define certain hardware aspects of your emulated phone and allow you to create many different configurations to test many Android platforms and hardware modifications. The emulator also allows the developer to simulate application interrupts such as SMS messages or phone calls.

The main purpose of the Android application is to provide the game aspect functionality of the CampuSeek experience. Users will be allowed to view their current challenge as well as the challenges they have remaining and ones they have already done. Along with this aspect, they will be able to pull up both their rank in the game and a chart of their distance traveled over a number of days. The user interfaces is illustrated below in Figure 6.

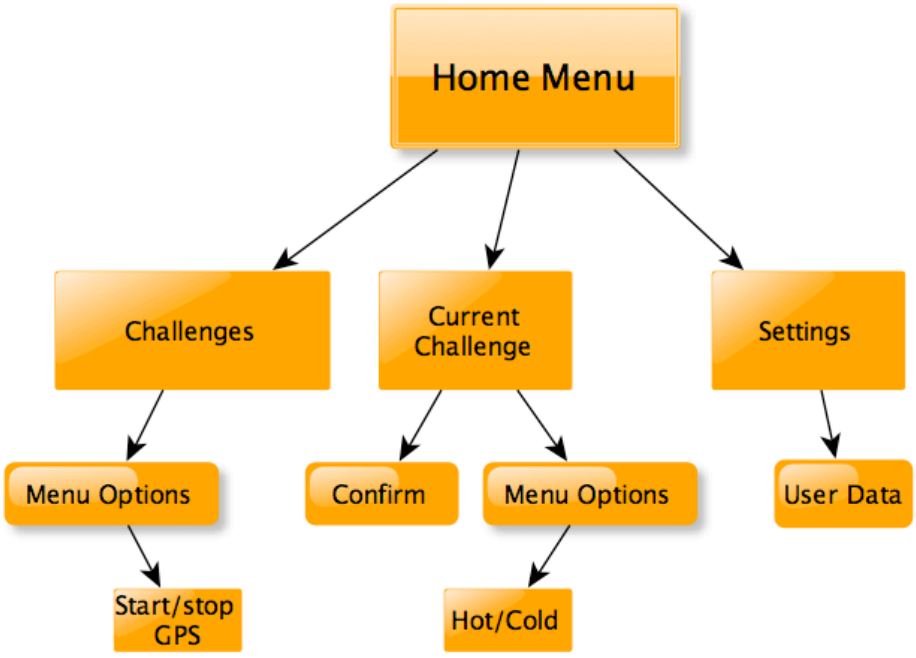


Figure 6: Android user interface

For the GUI, we implemented a Tab Layout. The tab layout allows the screen to be switched quickly by the pressing of tab buttons. The tab content can be implemented in one of two ways: using tabs to change views within the same Activity (code operation), or use the tabs to change between entirely separate Activities. In our application, separate Activities are used for each tab. We currently have 4 tabs in our layout: Challenges, Current Challenges, Progress, and Settings. Each tab has its own class, which extends the Android Activity class. Each tab page also consists of buttons, which link that page to another page. Here we incorporated a screen switching method. When a button that exists within a tab is pressed, a new screen is opened. Each new screen is also implemented as its own class.

Getting the user's current location is handled within the CurrentChallenge class. The "Current Challenge" tab will contain an image (using the ImageView module) of the object they are trying to find on Texas A&M's campus. This picture will match the one that is displayed on their Facebook Application challenges page. Users will then also have a graphical button that will be pressed when they believe they are at their current location. The code behind this uses a function setOnClickListener to determine whether the user has clicked it or not. For now, when the button is clicked, the phone passes in data provided by the network provider. This reads in a geo-location based on either a cellphone tower or wireless router (whichever one is broadcasting a stronger signal at the time). A pop-up notification appears displaying the users longitude and latitude values at that given instance. Ultimately, a simple check function will be implemented once the longitude/latitude values for the challenges are achieved and stored. Users will click the button to confirm their location and the GPS will compare their current location values with those stored in the database corresponding to the current challenge. Another class will be created to store the user's coordinates at set time intervals to calculate the distance they have traveled since using the app.

Integrating the Android application with the data that is used by the Facebook application has proven difficult. Initially, we simply required the Android user to click a button to open the link on our Facebook application. The Facebook application does not display well, however, within the Android application and is not an acceptable solution. We instead decided to integrate user login into the application itself. When

the user first runs the application, their Facebook user ID will be stored which will allow interaction with the database in conjunction with their secret password. Using an HTTP post, the Android application can connect to PHP scripts that are on our server that hosts the Facebook application. The PHP scripts that have been written provide the queries that we need and all that we have to feed into the HTTP post are the attributes and values that we need. In our case, we pass in the users Facebook ID number and all of their corresponding game data is returned to be utilized by the application.
Data management will be handled according to the flow illustrated below in Figure 7.

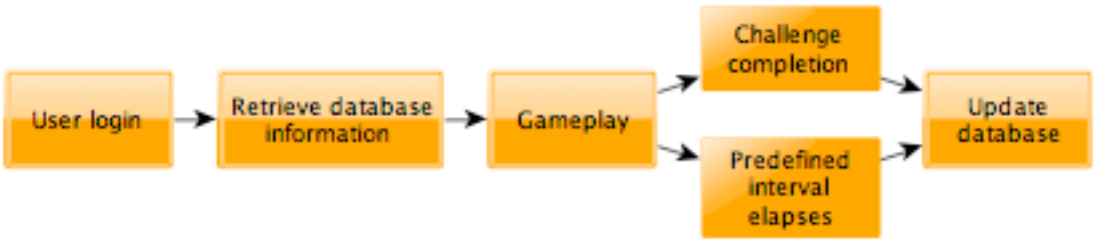


Figure 7: Android data handling

User and challenge data are initially retrieved from the central MySQL database at application startup. When the user completes a challenge or when a predefined time interval elapses (e.g. every 5 minutes), new data is uploaded back to the server.

Facebook-interface

Whereas the Android interface is used mostly for game-play features, the Facebook interface is primarily for collaborating, competing, organizing challenges and reviewing statistics *outside* of the game-play environment. The Facebook application provides an environment where players can see the big picture of which challenges they have left to complete and organize their various goals. Users can also compete with their friends. The Facebook application has an option for users to "follow" their friends. The performance of friends can then be compared in a competitive environment. Friends who are "followed" will show up in the user's feed, and statistics will be displayed ranking the user against their followed friends and charting their performances next to each other. Friends can also help each other reach their goals by comparing performance toward set goals.

Finally, the Facebook application will be the primary interface where users will access health statistics to chart their progress. The "Health View" on the Facebook application will give users access to all of their caloric consumption and distances travelled in a detailed graph format that shows changes over time. Users will be able to plan weight-loss goals and compare their performance with their goals as they progress through the game. The specifics on the health aspect will now be discussed.

As users travel on campus to complete challenges, their phone is constantly collecting data, which is updated to the central database. This information is used to display statistics on the user's performance. The Google Chart API [8] will be used to display data in an informative and readable fashion. This API has tools for displaying many types of charts that will be useful for our statistics.

The mobile application will also compute the calories burned with regards to the distance traveled by the user. The algorithm we are using for this is based off of US Army Fitness Manual [8]. When you register to use the mobile application you will be asked to enter your weight. The GPS will be able to monitor and provide the Facebook web application with the average speed at which the user has traveled throughout the course of the game. The time played will also be taken into account and the following algorithm will be applied:

Calories Burned = 0.64 calories / # of minutes / body weight (in lbs.)

The above algorithm is based off of the idea that the user traveled 5 MPH. The actual algorithm used will be able to scale this formula in a more robust fashion if the user travels faster or slower.

The web application is hosted on a separate server, which was purchased and set up for this project. It is embedded into the Facebook environment using an iframe, per Facebook's specifications. The web application is written using PHP and JavaScript for programming, CSS for layout and styling, and HTML for content display. The interplay of the various web languages is illustrated below in Figure 8.

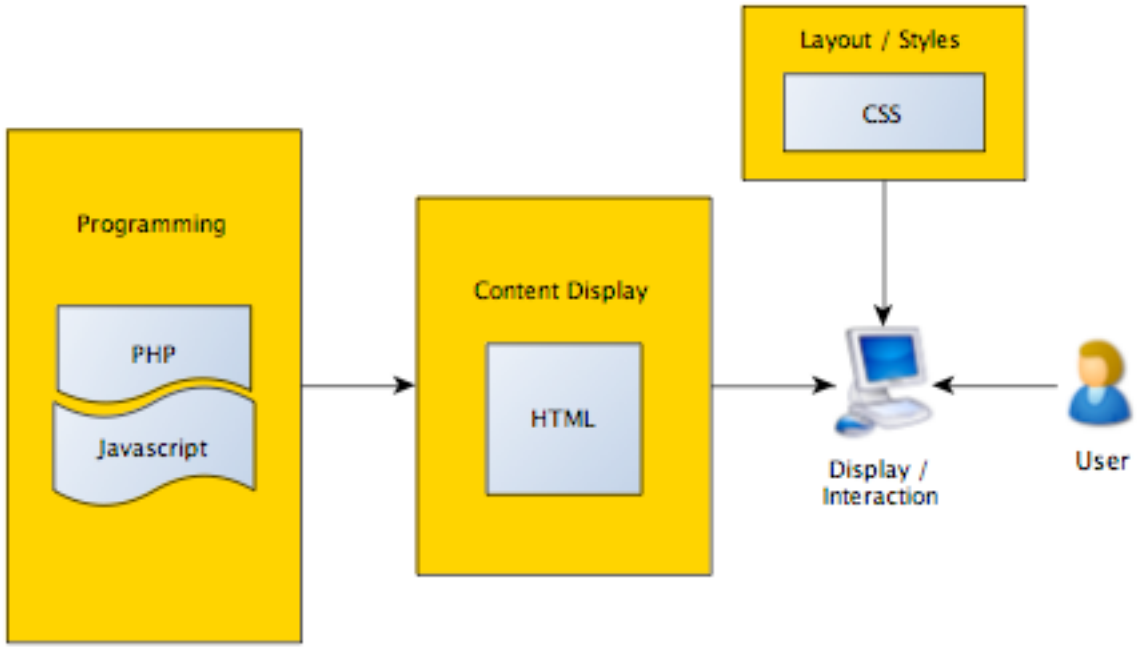


Figure 8: Web language interplay

PHP is used for almost all of the web programming, with the exception of some JavaScript and JQuery used for some page height auto-sizing callbacks. The Facebook API is made available for PHP and JavaScript, though the PHP API is used for most of this system. All pages used in the system are PHP pages, with some content of each page generated dynamically through the support of the auxiliary database. The database is implemented using MySQL with two tables to store User and Challenges information, respectively. The schemas for the two tables are described below:

Users		**Challenges**	
Attributes	Data Types	Attributes	Data Types
id	long	id	long
name	varchar	level	integer
android	boolean	description	varchar
level	integer (1,2,3)	image URL	varchar
challenges completed	list	latitude	double
current challenge	integer	longitude	double
calorie goal	integer		
calories burned	integer		
weight	integer		
height	integer		
distance travelled	integer		
following (list of ids)	list		

Table 1: Database table schemas

For the Users table, the ID stored is the same as the user's Facebook ID (which is the only Facebook user data that can be persistently stored and still comply with Facebook's privacy policy). This is used to match Facebook users when logged-in with their CampuSeek information. Also, the database keeps track of whether the user has registered their phone on the Android application. The 'level' attribute stores the user's level of 1, 2 or 3, corresponding to beginner, intermediate or advance, respectively. Also, the "following" attribute stores a list of the user IDs of the other CampuSeek users that each user is following.

The Challenges table stores information about each challenge so that the information can be dynamically generated in the Facebook application and so that the challenge completion can be confirmed within the Android app (latitude and longitude). The Facebook application uses information from the database on most of its pages. The navigation for the Facebook application can be seen below in Figure 9.

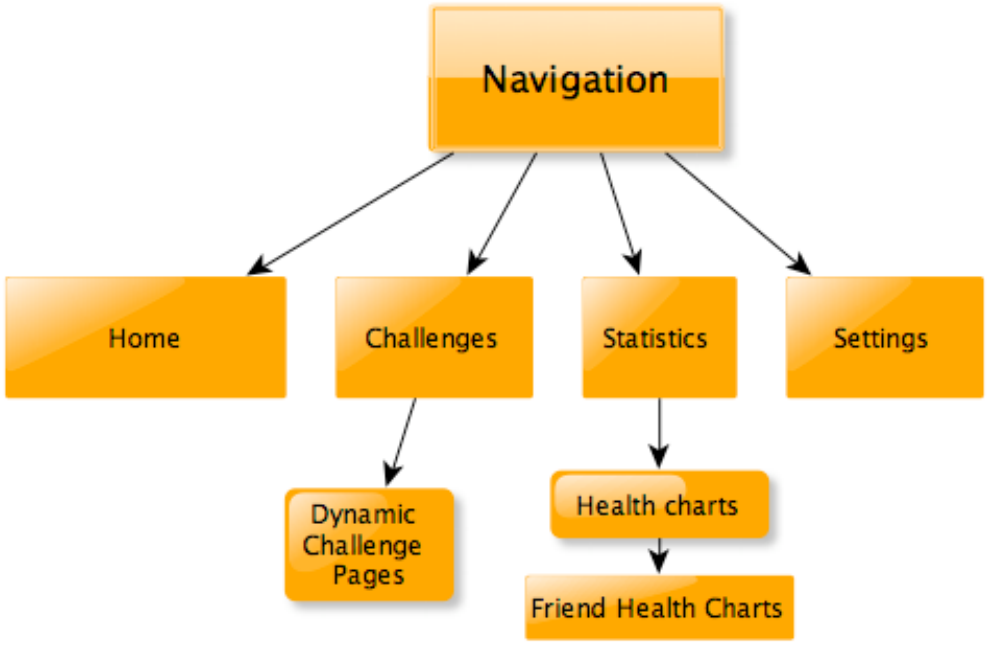


Figure 9: Facebook navigation

On the Challenge page, all of the challenges are loaded dynamically and displayed from the MySQL database. They are presented according to their difficulty level (beginner, intermediate and hard) in the form of a clickable image. Because the challenges are loaded and presented dynamically from the database, the web application is built to support the later addition of more challenges. With the creation of a simple form, CampuSeek users would be able to create and submit their own challenges, which would then be displayed to other users. Additionally, when each challenge is clicked on from the Challenges page, a new page is loaded dynamically that shows the challenge image along with a short description of what the challenge entails. At this point, most aspects of the web application are dynamic, enabling future expansion.

PHP is the primary language used for the web programming. Facebook provides PHP and JavaScript API's to access their modules and authentication functionalities. Because of its facility of interacting with MySQL databases along with its Facebook API support, PHP was an obvious choice as the main programming language for the system.

3.3 Approach for design validation

The validation and testing procedures of our application included a GPS test, intuitiveness test, a health statistics test and a phone compatibility test. The GPS test ensured that user current location gathering is accurate and can be stored locally within the phone for further data manipulation (distance traveled, calories burned). The intuitiveness test simply tested the ease of the Android application for users to ensure there the learning curve is not steep. Finally, the phone-compatibility test ensured that the Android application can be installed and distributed to many Android devices.

4 Implementation notes

Android

The Android application serves as the main platform for gameplay. Several technologies and methodologies were used to make the application come together, including Facebook integration using Facebook API's Single-Sign-On, use of background services and threads, and SQL database integration. These will now be discussed.

Single-Sign-On

The Facebook platform, using Single-Sign-On (SSO), lets you bring users and their friends to your mobile apps, creating a more engaging and personalized experience. The Facebook Platform provides a SDK for Android. If your mobile app runs on a platform without a Facebook SDK or runs in a mobile browser, the web-based Facebook Login, Platofrm Dialogs, and Graph API can be used directly. SSO lets users sign into our app using their Facebook identity. If they are already signed into the Facebook Android app on their device they do not have to type in their username and password. Since they are signing into CampuSeek with their Facebook identity, we have access to their profile information and social graph. This is important because we use the users Facebook id to organize data in the CampuSeek database.

The first time the user logs into CampuSeek, either via Facebook or their android phone, they will be prompted with a dialog box that allows the user to grant CampuSeek permission to access their information. If the user presses "Allow"', CampuSeek is authorized and we will have acces to the user's profile and social graph through the "facebook" instance. If the user presses "Don't Allow", CampuSeek will not be authorized and we will not have access to the user's data. By default, the user is asked to authorize the app to access basic information that is available publicly or by default on Facebook. If an app needs more than this basic information to function, you must request specific permissions from the user. We only need to access the basic information that is available publicly for CampuSeek. Methods used to access the Graph API are called synchronously and will block the UI, so we used a separate thread and then marshaled the results back tot he UI thread. The Facebook SDK provides a class, AsyncFacebookRunner that we utilized for this purpose.

Before we began development with the Facebook Android SDK, we had to install the Android SDK and dev tools, Git (the source control client we use for this SDK) and then clone the latest version of the SDK from GitHub repository.

Once everything was installed, we created a new Android project which was used for the Facebook Android SDK source. This project is used as a reference project for our app. Once the Facebook SDK was referenced, we had to modify the app manifest file to allow the app to make network calls to Facebook. This was accomplished by adding the following code to the AndroidManifest.xml file in the project:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Next we had to export the signature for our app so that Facebook can use it to ensure taht users are only communicating with our app on Android. This was done by running keytool. Keytool is a key and certificate management utility that comes with the Java SDK when you install it on your computer. It enables users to administer their own public/private key pairs and associated certificates for use in self-

authentication (where the user authenticates himself/herself to other users/services) or data integrity and authentication services, using digital signatures. It also allows users to cache public keys (in the form of certificates) of their communicating peers.

The following shows how we exported the key for our app using the debug defaults specificed by the Android SDK and Eclipse:

```
keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore
| openssl sha1 -binary
| openssl base64
```

Keytool generates a string that we had to register in the Mobile & Devices section of the Developer App for CampuSeek on the Facebook page. The default debug key can be used for development purposes but we had to create our own key when we were ready for testing and distribution on non-test phones.

When the application is loaded, the user is brought to a screen as shown below, where they log in:

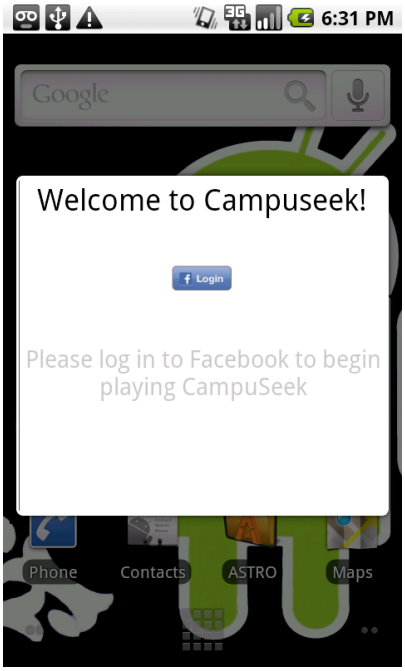


Figure 10: SSO - before logging in

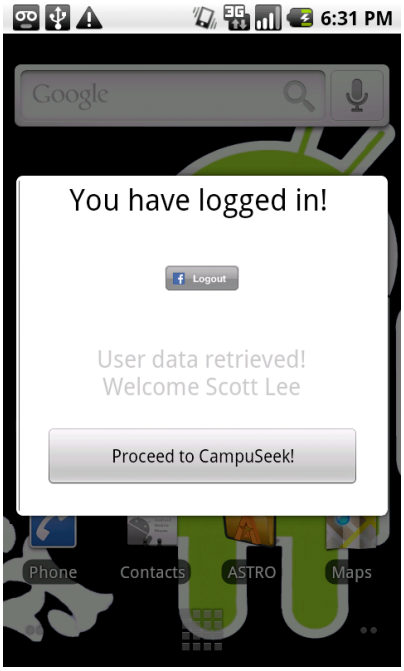


Figure 11: SSO - after logging in

The *proceed to CampuSeek* button is not be visible unless the user successfully logs in. This prevents users that do not have log in information from playing the game.

Some code snippets of the Login view and Single-Sign-On implementation are shown below.

LoginButton.java - “init” function declaration and implementation:

```
public void init(final Activity activity, final Facebook fb,
                 final String[] permissions) {
       mActivity = activity;
```

```
        mFb = fb;
        mPermissions = permissions;
        mHandler = new Handler();

        setBackgroundColor(Color.TRANSPARENT);
        setAdjustViewBounds(true);
        setImageResource(fb.isSessionValid() ?
                         R.drawable.logout_button :
                         R.drawable.login_button);
        drawableStateChanged();

        SessionEvents.addAuthListener(mSessionListener);
        SessionEvents.addLogoutListener(mSessionListener);
        setOnClickListener(new ButtonOnClickListener());
    }
```

LoginButton.java - “ButtonOnClickListener” function declaration and implementation:

```
private final class ButtonOnClickListener implements OnClickListener {

        mFacebook = new Facebook(APP_ID);
        AsyncFacebookRunner mAsyncRunner = new
AsyncFacebookRunner(mFacebook);


        public void onClick(View arg0) {
            if (mFb.isSessionValid()) {
                SessionEvents.onLogoutBegin();
                AsyncFacebookRunner asyncRunner = new
AsyncFacebookRunner(mFb);
                asyncRunner.logout(getContext(), new
LogoutRequestListener());
            } else {
                mFb.authorize(mActivity, mPermissions,
                              new LoginDialogListener());
            }
        }
    }
```

Data Retrieval

Once the user is logged in using Single-Sign-On, the application has access to the user’s Facebook ID as discussed above. This is a unique ID given to each Facebook user. The application then uses this ID to query the CampuSeek central database to get the user’s game information. All of the central database queries/updates that are performed by the application are done by sending an HTTP POST to a PHP script on our server. This script responds with results of the query or a success message if it is an update command. In this case, the script responds with the user’s game information, which our application then downloads and parses using the JSON protocol.

Using JSON to retreive the users id and name:

```
public void onComplete(final String response, final Object state) {
            try {
```

```
                // process the response here: executed in background thread
                Log.d("Facebook-Example", "Response: " +
response.toString());
                JSONObject json = Util.parseJson(response);
                userID = json.getString("id");
                userName = json.getString("name");

            } catch (JSONException e) {
                Log.w("Facebook-Example", "JSON Error in response");
            } catch (FacebookError e) {
                Log.w("Facebook-Example", "Facebook Error: " +
e.getMessage());
            }
        }
```

Application User Interface

The Android application is a collection of Activities (screens the user sees) which are connected together through the user interface using a TabHost widget. The TabHost creates a means for the user to easily switch between the various views (e.g. Challenges, Current Challenge, etc).

Challenge Data Retrieval

Once the user information is retrieved, the user is taken to the application main page, which is a list of challenges they can complete. Before this page can load, however, the challenges must be retrieved from the database. This is done using the data retrieval method discussed above. A script downloads the information from the database and it is parsed using JSON into a static class that is used by the application. Challenge data is not retrieved again unless the user chooses to refresh the challenges page using the Refresh menu option. At this point however, the challenge images are not downloaded. The image URLs have been retrieved, and the images are downloaded for each specific challenge when/if it is selected.

The challenges are displayed in an Android ListView widget. An Android Spinner widget is used to allow the user to select the level of challenge to display (beginner, intermediate, advanced). User's are not allowed to view challenges for a level they have not yet reached. The ListView is populated using an array of strings which each represent the text that is displayed. The algorithm for populating the array, called names is shown below. After this function runs, the ListView is updated to display the Strings that are in the names array. The ch_indices array maps the challenge's position in the names array to its challenge id, which is used to retrieve more challenge information when the user selects a challenge.

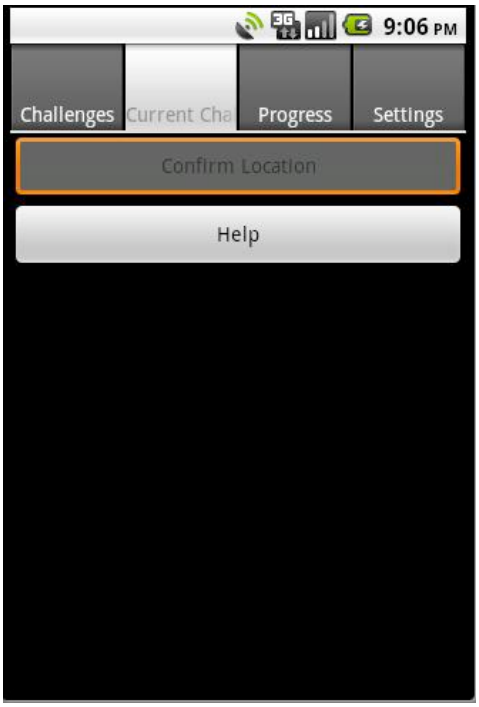
```
void displayChallenges(int level) {
        if(CurrentChallenges.challengedata == null)
            return;
        names = new ArrayList<String>();
        ch_indices = new ArrayList<Integer>();
        for(int i=0;i<CurrentChallenges.challengedata.length;i++) {
            if(CurrentChallenges.challengedata[i].level == level) {
                names.add(CurrentChallenges.challengedata[i].name);
                ch_indices.add(i);
            }
        }
}
```

Code Snippet 1: Populates ListView using names array

When the user selects a challenge in the ListView, they are taken to the CurrentChallenge tab.

Current Challenge Tab

Using TabHost as our main GUI for CampuSeek, each tab is treated as its own activity. The activity class is a vital part of the GUI as it is the part of the program that interacts with the user. The specific layout chosen for the Current Challenge Tab is a combination of both a Scrollview and a linear layout. The Current Challenge Tab underwent numerous graphical changes to ensure the intuitiveness would benefit the user. Initially nothing was shown on this tab other than a help button (which would later become an advanced feature) and a confirmation button.



Code Snippet 2: Previous design

Eventually, this was altered to include a picture of the challenge as well as a short description so the user could remember what they were seeking. The code was also modified so that once a user selected a challenge from the Challenge tab, the Current Challenge tab would open. To have the Current Challenge tab open to the appropriate challenge, the *current_challenge variable* had to be appropriately set. This was done by looking up the challenge id of the challenge at position *position* in the ListView. The view is then switched to the second tab (Current Challenge).

```
public void onListItemClick(ListView parent, View v, int position, long id)
{
        if(names == null) {
            getChallengeIdsNames();
            return;
        }
        if(CurrentChallenges.challengedata.length > 0)
            CurrentChallenges.current_challenge = ch_indices.get(position);

        TabHost tabHost =  (TabHost)
getParent().findViewById(android.R.id.tabhost);
        tabHost.setCurrentTab(1);
    }
```

Code Snippet 3: Setting the current challenge when a challenge is selected from the ListView

The Scrollview was later added to the Current Challenges page after the team decided to do a combination of pictorial challenges as well as riddles which may require more vertical space. The original linear layout allows for textviews, buttons and images to be stacked on top of each other, however the range of view is limited to the size of the device the user has. ScrollView allows for a much larger page, as the user can scroll. This allowed for larger pictures to be used to help the users easily identify their destination.

To add to the dynamic aspect, images for the current challenges are not initially stored locally within the application but rather downloaded. When a challenge is selected a check is performed to decide if the image for that challenge has already been downloaded. If it has not been it will be downloaded from the CampuSeek server and saved for faster retrieval. This is done using a map that associates a bitmap with its challenge id (as the key). When Bitmaps are downloaded, they are added to the map. Before each download, the algorithm first checks to see if the Bitmap is already in the map. This process can be seen below. It can also be seen that the description data for the challenge is parsed from HTML using the built-in Java class, HTML.

```
Bitmap bitmap;
if(bitmaps.containsKey(current_challenge)) { // check if in map already
    bitmap = bitmaps.get(current_challenge); // retrieve from map
}
else { // not already retrieved to map, download
    bitmap =
DownloadImage("http://campuseek.info"+challengedata[current_challenge].img);
    bitmaps.put(current_challenge,bitmap); // add to map
}
img.setImageBitmap(bitmap);
String desc = challengedata[current_challenge].description;
desctext.setText(Html.fromHtml(desc));
```

Code Snippet 4: Bitmap download and storage

A bonus feature to aid in the hunt for challenges around campus was the hotter/colder feature which also uses the GPS logging service to determine whether the user is getting closer or further away from their destination. If this option is chosen, a separate class is opened via a new intent. A timer is initialized with a pre-determined amount of time for each user and will begin to count down when this menu option is chosen. Once the user has found the challenge or exits out of the hotter/colder window, the timer is put on pause and their time remaining is sent to our database through na HTTP Post.

Challenge Verification

The confirmation process is also handled within this portion of the Current Challenge Tab but will only be touched on briefly as a more precise detailed explanation is given later on. As noted before, challenge data is stored into an array. Once a challenge is selected the data pertaining to that challenge from the array is passed into a temporary local variable, current challenge. The latitude and longitude of the challenge selected is passed into the confirmation function which utilizes the location provider from the GPS service that runs in the background in its own thread.

The click of the confirmation button is recognized by using a callback function that passes the longitude and latitude into a function *distanceBetween*(). This function has parameters of two sets of longitude/latitude values and then an array to store the results. For our specific purpose, we pass in the user's current longitude and latitude values and calculate the distance from the associated current longitude and latitude values that are stored in the database for that specific current challenge. The distance is calculated between the two points and the result is store into 4 member array. By default the distance is calculated and measured in meters, our result is then multiplied by the conversion of 3.28 to convert this into feet so the user has a better visualization of how far they are from the destination if they are not within range. Each challenge is attributed a certain precision to which the user must be. This allows for the user to be within eye sight of larger challenges and close up to challenges that are smaller in stature. The GPS hardware on the phone is accurate up to a meter which allows for accurate comparison values to ensure the user is at the correct destination. A code snippet used for challenge verification is shown below.

```
public void onLocationChanged(Location loc)
        {
          double lat = loc.getLatitude();
          double lon = loc.getLongitude();
          float [] results = new float[3];
          float distance; // in feet

          Location.distanceBetween(lat,lon,
                   challengedata[current_challenge].latitude,
                   challengedata[current_challenge].longitude,results);

          distance = (float) (results[0] * 3.28);
          if(distance < 150) { // completed challenge!
              Toast.makeText(getBaseContext(),"Success! " + distance + "
              feet!",Toast.LENGTH_LONG).show();
              if(StaticUserData.chs_completed.lastIndexOf(
                 challengedata[current_challenge].id) == -1)
                StaticUserData.chs_completed.add(challengedata[current_challenge].id);
                Collections.sort(StaticUserData.chs_completed);
...
```

An alert dialog appears on the screen informing the user of their success or their failure. The AlertDialog is similar to a Toast notification, except it allows buttons, styling and extra functionality. Once the user has successfully completed a challenge, the app calls a PHP script to update the database with the user's achievement.

```
String chs = "";
for(int i=0;i<StaticUserData.chs_completed.size();i++)
      chs += String.valueOf(StaticUserData.chs_completed.get(i)) + ",";
chs = chs.substring(0, chs.length()-1);
```

```
String result = Query.DBQuery("android-updatechallengecomplete.php", "id",
String.valueOf(StaticUserData.id), "chs", chs);
Toast.makeText(getBaseContext(),
                              result + "\n" + chs,Toast.LENGTH_LONG).show();
```

GPS / Background Service

A primary component of CampuSeek was the GPS module that served various purposes for the main objective we set out to accomplish. The GPS module is used to ensure the user is at the correct challenge destination as well as to calculate the distance the user has traveled since starting the application so that their health statistics can be graphed.

The GPS system was configured to run in a background service. This was so that constant updates would not hang up the main application thread. The GPS service is called *Logger* in the application and handles GPS updates, as well as time-interval database updates, which happen in a separate thread within the service using a Java *TimerTask*. An Android Service was used for the GPS rather than just a raw thread because the Android API provides functionality to easily start services and synchronize information between Services and Activities. This was necessary, as our application Activities require the use of the GPS module from the service. The service is configured to run in the background as soon as the user logs in to the application using Single-Sign-On.

```
campuseek.setOnClickListener(new View.OnClickListener() {
    public void onClick(View arg0) {
    startService(new Intent(Login.this, Logger.class)); // start the service
    startActivity(new Intent(Login.this, Tabs.class));
}
```

The GPS service is utilized by many different classes. For an Activity to be able to access the GPS, it must *bind* itself to the service. A callback to *onBind()* is implemented within the service and classes that want to act as a client calls *bindService()*. Once this is called, the *onServiceConnected()* callback is called in the binding activity and the Activity can now access Service resources (such as GPS coordinates in this case).

The algorithm for the calculating the distance traveled is not too difficult. A function *requestLocationUpdates()* can be used with a set of parameters to instruct the GPS module when to pull the users current longitude and latitude. In our case our function looked as follows:

```
locationManager.requestLocationUpdates(provider, 1500,gpsMinDistance, locationListener);
```

Our parameters in this case were utilizing the *GPS_PROVIDER* as well as updating every 1500ms. To calculate the distance the user travels, every time the GPS receives a new update, the new coordinates are compared to the previous coordinates using the built-in *distanceBetween()* function. This calculated distance is then added to the total distance the user has travelled. The database is updated by adding the new distance to the old at a set interval (currently 7 seconds), at which point the local distance variable is reset to 0 and readings resume. The following code snippet displays the two functions mainly responsible for calculating the distance the user walks throughout game play.

```
private void updateMeasurement(){
            distance +=
calcGeoDistance(startLat,startLon,currentLat,currentLon);
            Toast.makeText(getBaseContext(), "Distance: " +
RoundDecimal(distance,2),
            Toast.LENGTH_SHORT).show();
}


// return distance in feet
private double calcGeoDistance(final double lat1, final double lon1, final
double lat2, final double lon2)
{
            final float[] results = new float[3];
            Location.distanceBetween(lat1, lon1, lat2, lon2, results);
            return results[0]*3.28;
}
```

A callback to the *onGPSUpdate()* function ensures that the *updateMeasurement()* function is repetitively called. The logging service is started using a function called *startLogging()* that contains the *requestionLocationUpdate()* function as well as a timer for how often it the thread is executed.

```
public void startLogging(){

        scanTask = new TimerTask() {
            public void run() {
                        Settings.updateDistance(distance);
                        distance=0;
                }
        };

        t = new Timer();
        t.schedule(scanTask, 500, 7000);
        locationListener = new MyLocationListener();
        if(locationManager == null)
            locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        Criteria criteria = new Criteria();

        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setAltitudeRequired(false);
        criteria.setBearingRequired(false);
        criteria.setCostAllowed(true);
        criteria.setPowerRequirement(Criteria.POWER_LOW);

        locationManager.requestLocationUpdates(provider, 1500,gpsMinDistance,
locationListener);

        Toast.makeText(getBaseContext(), "GPS started",
Toast.LENGTH_LONG).show();

    }
```

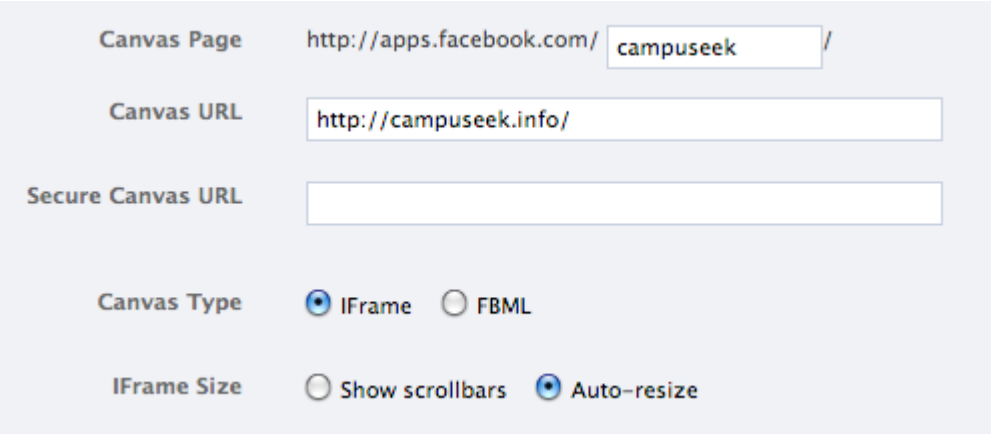
An accompany *stopLogging()* function cancels the thread and removes all GPS coordinates from memory.

Privacy was a big topic of concern for our project seeing we were using the phones GPS to track the distance traveled by users. To ensure user privacy, no GPS coordinates were permanently stored on the phone (the method has been previously discussed). An added feature was the ability for the user to turn off the GPS service at any time. Pressing the menu button on the phone's hardware brings up a menu item that enables a toggling of the GPS service – on/off.

This added item to our menu ensures that the user has complete control over how the GPS interacts with them.

Facebook

The first decision we encountered when developing the Facebook web application was how to integrate our site with Facebook. The two options were to use Facebook's FBML (Facebook Markup Language) or iframes. The first option provides for easy integration of our site with Facebook and would allow us to take advantage of Facebook's style information through the use of their CSS elements. The second option, iframes, involves creating the Facebook application on our own server and integrating it with Facebook by just including the site as an iframe. This is all done through the Facebook Developer application settings menu.



We chose to use iframes after learning that Facebook will be phasing out FBML in the coming year and that iframes are encouraged by most developers.

Now that the Facebook application was setup, we began searching for the right web host to host our application. The requirements we were looking for in a host included:

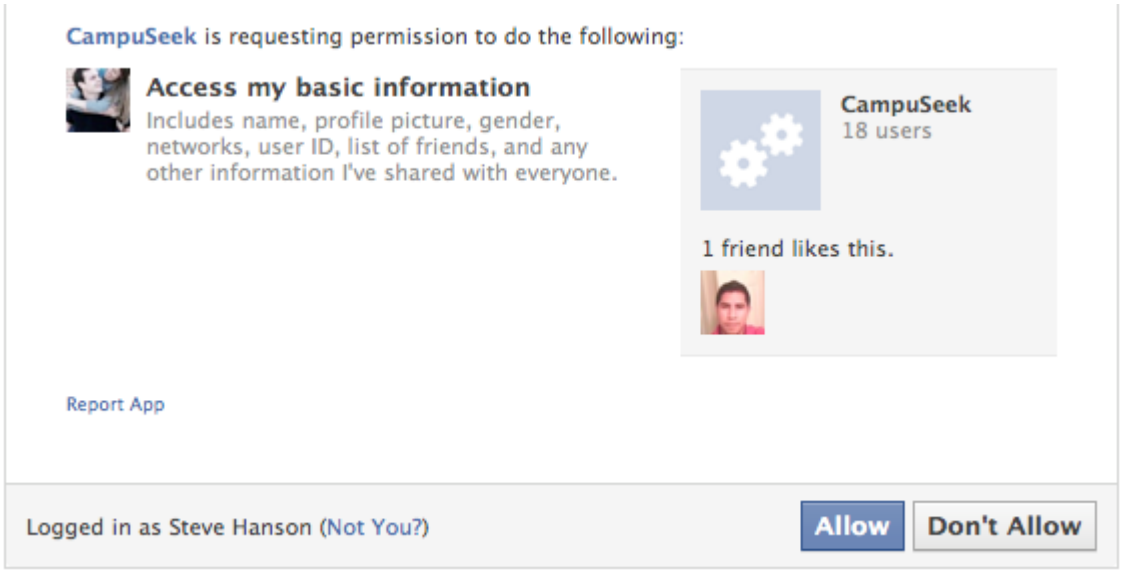
- Must provide access to PHP 5
- Must provide MYSQL database
- Must allow database access from different domains (for Android access)
- Must allow at least 100MB data storage

We first decided to use the free hosting provided by ByetHost (http://byethost.com), but realized after some frustration that some essential PHP functionality had been disabled (the fopen and curl libraries) that were necessary for the Facebook authentication protocol (OAuth 2.0). We then upgraded to the ByetHost premium service, which cost $11.95/3 months, and achieved access to all of the necessary functionality that we needed. Our web domain is *http://campuseek.info* which was provided freely with the hosting plan.

The first step in developing our application is integrating Facebook's authentication module, OAuth 2.0, into our application. The OAuth process has three steps: application authentication, user authentication and application authorization. The first two steps are performed simultaneously. When the user goes to our application page, application authentication is performed by sending an authentication request to Facebook supplying Facebook with our application ID and redirect URL that will be loaded when the OAuth authentication process succeeds. The string is in the format as follows:

```
https://www.facebook.com/dialog/oauth?
    client_id=YOUR_APP_ID&redirect_uri=YOUR_URL
```

Facebook uses our application ID and the server from which the request is sent to verify that it is our app making the request. It then performs user authentication by presenting the user with a login prompt if they are not logged in. If they are already logged in, this step has already been performed. Finally, application authorization is performed. This is done by presenting the user with a dialog informing the user of which information it would like to use.



CampuSeek only uses basic user information, including the user's user ID, name, picture and list of friends. The user can allow if they wish to proceed. If the user accepts, the authentication process is complete, and they are redirected to the redirect-URL. The user only authorizes the application on their first visit to the application. The user can deauthorize the application at any time in their privacy settings.

Once the user has gone through the authorization process, they land on the CampuSeek introduction page. Because the user has authorized the application, CampuSeek can request user data using the Facebook PHP Graph API. An associative array, *$me*, is initialized by calling to the Facebook Graph using the following code:

```
$me = $facebook->api("/me");
```

Now the application can access the user's information with the *$me* variable. For instance, *$me['id']* stores the user's ID and *$me['name']* stores the user's name. These are the only two pieces of data used directly by the *$me* variable in our application. Another request is made, however, to retrieve the user's friends:

```
$friends = $facebook->api('/me/friends');
```

This data is retrieved for the *Friend Following* feature that will be described in more detail momentarily. Only public information is made available to our application for each of the user's friends, such as their Facebook ID, name and picture thumbnail. This information is available publicly by Facebook and is part of Facebook's privacy agreement for all users. Additionally, if users do not wish for this to be public, they can set this in their Facebook privacy settings.

Database Integration

For our application, a database is necessary to store information about each user, including their challenges completed, distance travelled, friends following and various preferences and settings. The database is shared by the Facebook application and the Android application.

Users		**Challenges**	
Attributes	Data Types	Attributes	Data Types
id	long	id	long
name	varchar	level	integer
android	boolean	description	varchar
level	integer (1,2,3)	image URL	varchar
challenges completed	list	latitude	double
current challenge	integer	longitude	double
calorie goal	integer		
calories burned	integer		
weight	integer		
height	integer		
distance travelled	integer		
following (list of ids)	list		

Figure 12: Database design

The first challenge was to figure out a way to integrate the database with Facebook and the Android application. We determined that storing users in the database using Facebook's user ID as the key would allow easy integration. When the user lands on the Facebook application page, their CampuSeek information is retrieved from the database by calling the following PHP function which we defined in our *functions.php* script.

```php
function cs_get_user_info($me) {
    $name = $me['name'];
    $id = $me['id'];
    $query = 'SELECT * FROM Users where id=' . $id;
    $result = mysql_query($query)
              or die('Query failed: '. mysql_error());
    if(mysql_num_rows($result) == 0) { // user not in database
        echo 'User '.$id.' not in database. Adding ...<br/>';
        $query = "INSERT INTO `Users` (id,name,date_started)
                  VALUES('$id','$name',CURDATE())";
        mysql_query($query)
              or die('Query failed: ' . mysql_error());
    }
    $arr = mysql_fetch_array($result, MYSQL_ASSOC);
    mysql_free_result($result);
    return $arr;
}
```

The function is called by each page that needs access to this information. The general flow for database access on the CampuSeek pages is as follows:

```php
$db = cs_connect_db();
$user = cs_get_user_info($me);
$challenges = cs_get_challenges_info();
cs_close_db($db);
```

Notice that this page also accesses challenge information from the database using a similar function in our *functions.php* file, *cs_get_challenges_info()*. Connecting to and closing the connection to the database are done using the following PHP functions from the *functions.php* file:

```php
function cs_connect_db() {
    // Connecting, selecting database
    $db = mysql_connect('', 'campusee_usr', 'aggies11')
        or die('Could not connect: ' . mysql_error());
    mysql_select_db('campusee_db')
        or die('Could not select database');
    return $db;
}

function cs_close_db($db) {
    // Closing connection
    mysql_close($db);
}
```

The page now has access to user game information, such as which challenges they have completed and which friends they are following. Now that the basic structure and function of the Facebook application

have been explained, the individual pages (home page, friend following, challenges, current challenge and health statitistics) and their implementation will be described in detail.

Home page

The CampuSeek homepage is the simplest of all of the application's pages. This page only displays a welcome to the user, with a personalized message (name retrieved using the Facebook API '*$me['name']*'), instructions for how to get started with the game, and a news feed that displays user interaction. This page does not even use any information from the database (user or challenge tables). On page load, however, a query is made to the database to determine if the user is currently registered with the database. If not, the user is added to the database, with all default settings.

Follow Friends Page

This page allows the user to choose friends to follow who are also playing CampuSeek. The algorithm behind this page works by first querying the Facebook API to retrieve the user's list of friends. This list is then cross-listed with the Facebook user IDs of the users in the database to achieve a list of the user's friends who are in the CampuSeek system. Some of the steps for this operation are shown below:

```
    // get all userids in database
    $users = cs_get_users();
    $userids = array();
    foreach($users as $use)
        $userids[] = $use['id']; // add user ID to array

    // store array of {name, id} of all friends on campuseek
    // called $friendids
    $friends = $facebook->api('/me/friends');
    $friendids = array();
    foreach($friends['data'] as $friend) {
        // if friend ID is in array of user IDs of CampuSeek
        if(in_array($friend['id'],$userids))
            $friendids[] = $friend; // add friend to list to display
    }
// retrieve users following from database
$already_following = cs_get_following($me['id']);
```

These friends are then displayed to the user with a check-box option to follow them which is filled if the friend is already being followed. The code for displaying the friends to the user is shown below, along with the actual output on the website:

```
<div class="box1">
<h3> Your Friends on CampuSeek </h3>
<br /><br/>
<form name="input" action="friends.php" method="get">
<?php
$i=1;
foreach($friendids as $buddy)       // Display each friend
{
```

```
    if(in_array($buddy['id'],$already_following))
        $checked = "checked";
    else
        $checked = "";
    ?>
    <div class="row<?=$i?>">
    <table><tr><td width="300">
    <h4><?=$buddy['name']?></h4><br/>
    <img
src="https://graph.facebook.com/<?=$buddy['id']?>/picture?type=large"
height="120" />
    </td><td>
    <center><input type="checkbox" name="follow[]"
            value="<?=$buddy['id']?>" <?=$checked?>/>
              follow</center>
    </td></tr></table>
    </div><!-- row -->
    <?php
    ($i==1) ? $i=2 : $i=1;
}   // end foreach loop, display submit button
    ?> // end PHP, begin HTML
<br/>
<center><input type="submit" value="Apply Changes" /></center>
</form>
</div> <!--  box1 end -->
```

Challenges Page

The Challenges page displays pictures of each challenge, grouped by challenge level (beginner, intermediate, advanced). The user cannot view challenges at a level higher than they are currently on. For example, an intermediate player cannot view advanced-level challenges until they advance to that level. The challenges are retrieved by querying the database for challenges of each level:

```
$challenges_level[] = cs_get_challenges_for_level(1); // beginner
$challenges_level[] = cs_get_challenges_for_level(2); // intermed
$challenges_level[] = cs_get_challenges_for_level(3); // advanced
```

The *cs_get_challenges_for_level($level)* function queries the database with this query and returns the results:

'SELECT * FROM Challenges where level=' . $level;

After the query, the *$challenges_level* array stores challenges for each level. For example,

$challenge_level[0][0]['id'] is the id of the first challenge of beginner level.
$challenge_level[1][2]['img'] is the url of the image of the third challenge of the intermediate level.

The challenges are then displayed on the page:

```
<?php

$challenge_comp = array();
$level_names = array("Beginner","Intermediate","Advanced");
$challenge_comp = explode(",",$user['ch_completed']);

for($i=0; $i<3; $i++) {
    echo $text[$i]; // if they aren't on appropriate level yet
    echo '<div class="box1"><h3>'.$level_names[$i].'</h3><br />';

    if($user['level'] < ($i+1)) { // not on high enough level yet
        echo '<center><img src="img/q_mark.png" width="200" />';
    }
    else {
        //Display images for challenge level
        foreach($challenges_level[$i] as $chall) {
            echo '<div class="box2">';
            echo '<a
href="http://apps.facebook.com/campuseek/challenge_pg.php?ch=';
            echo $chall['id'];
            echo '" target="blank"><img src="';
            echo $chall['img'];
            echo '" width="179" height="169" align ="right" border ="0"
/>';
            if(in_array($chall['id'],$challenge_comp)) {
                echo '<div style="z-index: 100; left: 12px; position:
```

```
relative; top: -150px">';
                    echo '<img src="img/checkmark.png" width="150"></div>';
                }
                echo '</a></div>';
            }
        }
        echo '</div> <!-- Challenge level box end -->';
        echo '<div class="clear"></div>';
}
?>
```

Current Challenge page

The Current Challenge page is a dynamic page that is loaded when a user clicks on a challenge to view. This page displays the title, image and description of the challenge, as well as the friends of the user who have completed it.

Challenge information is loaded dynamically when this page is loaded. When a user selects a challenge from the Challenges page, the challenge ID is sent to the Current Challenge page through HTTP GET. The Current Challenge page retrieves this ID through a GET request and loads the corresponding challenge information through a database query.

```
$num = $_GET['ch'];  // get the challenge number!

$challenge = cs_get_challenge_info($num); // get the challenge info!
```

The page then verifies that the user is allowed to view the challenge page:

```
// Not authorized to view challenge: redirect
if($challenge['level'] > $user['level']) {
     header( 'Location: http://campuseek.info/challenges.php' ) ;
}
```

Finally, after checking which friends have completed the challenge, the content of the page is displayed, which was mostly loaded dynamically:

```
<img src="<?=$challenge['img']?>" style="float:left;padding" width="300">
<p>Welcome to Challenge <i><?=$challenge['name']?></i>!
<?=$challenge['description']?> // prints the challenge description
<div class="clear"></div><br/>
<table width="700" border="0" r>
<tr bgcolor="#fff">
<td>Friends who've completed:</td>
<td>Friends who've <i>not</i> completed:</td></tr>
<tr>
<td><?=$friends_completed?></td>
<td><?=$friends_not_completed?></td>
```

```
</tr></table>
<br/><a href="http://campuseek.info/friends.php">Follow more
friends!</a><br/><br/><br/>
```

As can be seen, the content from this page is almost completely from the database, with little content hard-coded into the page. This makes the application very easy to expand with new challenges.

Health Statistics

The health statistics page allows the users to view weekly health goals and distance traveled. Furthermore users can view weekly health goals and distance traveled for friends that are the user is currently following. We first began by getting the users start date from the database. This is then converted to a usable date. Some of the steps for this operation are shown below.

```
$arr = get_user_date($me['id']);
   $d = $arr['UNIX_TIMESTAMP(date_started)'];
   date = getdate($d);

   //used to display week range below for graph below
   $weekStarts = getWeeks($d);
   $size = count($weekStarts); //count of array of weeks

    foreach($weekStarts as $week){  //used to display weeks in the format
m/d/y
        $userWeeks .= date("m/d",$week)."|";
       }
```

We first began by getting the UNIX_TIMESTAMP, and store it as an array to be used in our getWeeks function. This function is located in our stat-functions.php file, which takes the date started of each user and returns the date of each week (this function is shown below). This information is necessary as weekly health statistics are being displayed. The size of the array of weeks is determined, this is used within are graphical display charts which will be discussed later. Further manipulation is made to display out week ranges in the format m/d and the | character. This format is necessary as it is used within the Google Charts API to display week ranges.

```
// returns array of timestamps for startdate of each week
   function getWeeks($startdate) {
        $today = $startdate;
        //$arr = array();
        $numweeks = floor(intval(($today-$startdate)/86400)/7 +1);
        for($i=0; $i<$numweeks; $i++){
            $arr[] = $today;
            $today = $today + 86400*7; // move up a week
        }
        return $arr;
   }
```

Distance is stored as a string of values in feet within our database. For each distance we calculate the miles traveled, this value is used to calculate the amount of calories burned.

Calories burned are calculated by taking the user's body weight in pounds times .73 times the number of miles that was previously determined. These operations are shown below, the string of calories burned is used in are chart to display a Calorie Goal versus Calories Burned Chart. The calorie goal is a field determined by the user in our setting page discussed later.

```
/* approximate number of calories you expend by running a certain
distance can be found by taking 0.73 times your body weight in pounds
times the number of miles */
    foreach($usr_distance as $calc_calburn){
        $mile_dist .= $calc_calburn * .0001893939. ",";
        // feet to miles
        $new_mile_dist = substr($mile_dist, 0, -1);
        $calc_calburn2 .= $calc_calburn * .0001893939 *.73 *
$user['weight']. ",";
        $newcalc_calburn2 = substr($calc_calburn2, 0, -1);
        }
```

Chart Display

The Google Chart API is used to display the distance traveled and calorie goal vs calories burned chart in are health statistics page. The Google Chart API returns a chart image in response to a URL GET or POST request. All the information about the chart, such as chart data, size, colors, and labels, are part of the URL. Vertical bar charts were used to display data for distance traveled and calorie goal vs calories burned. They are displayed in are application using an <img> tag. The code used to display our bar charts are displayed below.

```
<div class="box1">
<center><h2> Welcome to Your Workout Performance</h2></center>
<center><img style="border:1px solid black;"
             src="http://chart.apis.google.com/chart?chxl=1:|<?=$userWeeks?>
&chxr=0,0,<?=$max_x?>|1,0,<?=$size?>
&chxs=0,676767,10,0,lt,676767|1,676767,10,0,lt,676767
&chxt=y,x&chbh=a&chs=650x220&cht=bvg&chco=A2C180&chds=0,
<?=$max_x?>&chd=t:<?=$new_mile_dist?>&chtt=Distance+Traveled+(per+week+in+miles)"
width="650" height="220" alt="Distance Traveled(Per Week)" /></center>
<p></p>
<center><img style="border:1px solid black;"
             src="http://chart.apis.google.com/chart?chxl=0:|<?=$userWeeks?>
&chxr=0,0,<?=$size?>|1,0,<?=$ssize?>
&chxs=0,676767,10,lt,676767|1,676767,10,0,lt,676767&chxt=x,y
&chbh=a,1&chs=650x240&cht=bvg&chco=3366CC,7777CC&chds=0,
<?=$ssize?>,0,<?=$ssize?>&chd=t:<?=$user['cal_goal']?>|<?=$newcalc_calburn2?>
&chdl=Calorie+Goal|Calories+Burned
&chtt= Calorie+Goal+vs+Calories+Burned+(per+week)" width="650" height="240"
alt="Calorie Goal vs Calories Burned" /></center>
<br>
```

```
<a href="http://campuseek.info/settings.php">Don't forget to set your weekly
calorie goal!</a>
</br>
</div>
```

Although the Google Chart API is a powerful graphical chart displaying tool, it lacked some of the functions needed to make a truly dynamic graph. For example there was no way to auto range data values for our charts. As a result functions were created to get the max data points and scale the chart. This was done to give a better perspective of the data values exhibited in each graph. The output shown by our website is shown below.

Friend Following Health Statistics

Another feature implemented in the health statistics page is a friend following feature. This feature allows the user to display distance traveled charts and calorie goal vs calories burned charts for each friend they are following. We first begin by getting the array of friends the user is following. We then load each friend's calorie goal, distance traveled and date started. To display health charts we retrieve the same information as done previously for the user. This information includes UNIX_TIMESTAMP, interval weeks, size of array of week, calculated distance in miles and calculated calories burned. The friend following health statistics code is shown below.

```
foreach($following as $friend) {
   $friendw = '';
   $friend_calc_calburn2 = '';
   $friend_mile_dist = '';
```

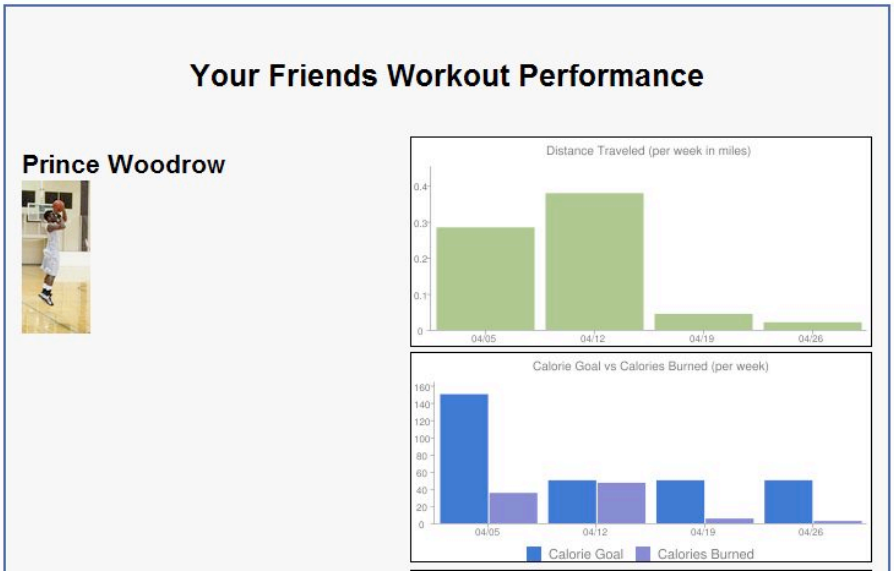
```
    $friend_info = cs_get_user_info_simple($friend);
    $arr2 = get_user_date($friend);
    $friend_d = $arr2['UNIX_TIMESTAMP(date_started)'];
    $friend_weeks = getWeeks($friend_d);
    $friend_size = count($friend_weeks);

    $friend_calgoal = $friend_info['cal_goal'];
    $friend_calburn = $friend_info['cal_burned'];
    $friend_distance = $friend_info['distance'];
        $friend_distance2 = array();
        $friend_distance2= explode(',',$friend_info['distance']);
        foreach($friend_distance2 as $friend_calc_calburn){
        $friend_mile_dist .= $friend_calc_calburn*.0001893939. ",";
        $new_friend_mile_dist = substr($friend_mile_dist, 0, -1);
        $friend_calc_calburn2 .= $friend_calc_calburn *
        .0001893939 * .73 * $friend_info['weight']. ",";
         $newfriend_calc_calburn2=substr($friend_calc_calburn2,0, -1);
          }
    $new_friend_mile_dist_range=explode(',',$new_friend_mile_dist);
    $max_range_friend = max($new_friend_mile_dist_range);
    $max_xfriend = $max_range_friend *1.2;
    $friend_name = $friend_info['name'];
    $friend_id = $friend_info['id'];
        foreach($friend_weeks as $friend_w){
            $friendw .= date("m/d",$friend_w). "|";
            }
```

Charts are displayed in the same fashion as described above, using the Google Chart API. The code and output displayed on our web page is shown below.

```
<div class="row<?=$i?>">
<table>
<tr>
<td width="375">
<h4><?=$friend_name?></h4><br/>
<img src="https://graph.facebook.com/<?=$friend_id?>/picture?type=large"
     height="125" /></td>
<center><td><img style="border:1px solid black;"
src="http://chart.apis.google.com/chart?chxl=1:|<?=$friendw?>
&chxr=0,0,<?=$max_xfriend?>|1,0,<?=$friend_size?>
&chxs=0,676767,7.5,0,lt,676767|1,676767,7.5,0,lt,676767
&chxt=y,x&chbh=a&chs=375x170&cht=bvg&chco=A2C180
&chds=0,<?=$max_xfriend?>&chd=t:<?=$new_friend_mile_dist?>
&chtt=Distance+Traveled+(per+week+in+miles)
&chts=676767,10" width="375" height="170"
                alt="Distance Traveled(Per Week)" /></td></center>
<tr><td></td>
<center><td><img style="border:1px solid black;"
src="http://chart.apis.google.com/chart?chxl=0:|<?=$friendw?>
```

```
&chxr=0,0,<?=$friend_size?>|1,0,<?=$fsize?>
&chxs=0,676767,7.5,lt,676767|1,676767,7.5,0,lt,676767
&chxt=x,y&chbh=a,1&chs=375x170&cht=bvg&chco=3366CC,7777CC
&chds=0,<?=$fsize?>,0,<?=$fsize?>&chd=t:
        <?=$friend_calgoal?>|<?=$newfriend_calc_calburn2?>
  &chdl=Calorie+Goal|
  Calories+Burned&chdlp=b&chtt=Calorie+Goal+vs+Calories+Burned+
  (per+week)&chts=676767,10" width="375" height="170"
  alt="Calorie Goal vs Calories Burned" /></td></center>
</tr>
</tr>
</table>
</div>
```



Settings

The settings page is a basic application page. This page is where users can provide some information about themselves. The most important feature this page provides is the weekly calorie goal input and weight fields provided by the user. This fields are used in conjunction with the health statistics page. These fields are necessary to calculate calories burned and display calorie goal versus calories burned charts. The setting page displays the users name, weekly calorie goal, weight and height. Each field is set in a form style updated by the user at their discretion. When the field is updated it is automatically saved to the database with the corresponding input

5 Experimental results

GPS Test

Installation, Phone Compatibility Test

To test the installation and phone compatibility, the CampuSeek Android application was downloaded from the Internet and installed onto three different Android phone models. To install the application, users were directed to the URL: http://campuseek.info/app/campuseek.apk. After directing their phones to the URL, the Android phones downloaded and installed the application. Each installation succeeded and the program successfully launched on each phone. The game accurately loaded user and challenge information and allowed the users to complete challenges, with accurate GPS measurements on each phone. There were no compatibility issues experienced, although the application was not tested on an Android phone with Android 2.0 or lower as such a phone was not available and CampuSeek does not officially support Android phones with operating systems below 2.1.

Basic Application Functionality Test
The application functionality was tested through rigorous hands-on use of the game. Our team downloaded the application onto three Android phones and played the game around campus. Various simulations were performed and the results are shown below with assigned scores (out of 3):

Functionality	Score (/3)	Description
User login	3	Worked flawlessly for new and returning users
GPS outdoors	3	Performed flawlessly for all phones
GPS indoors	1	Could not acquire signal (no indoor challenges)
Challenge verification	3	Worked as expected. Fails if GPS has not initialized
Hotter / Colder	2	Worked as expected Updates come in too frequently if signal is strong. User must walk fast for the hotter/colder not to fluctuate Fails if GPS has not initialized

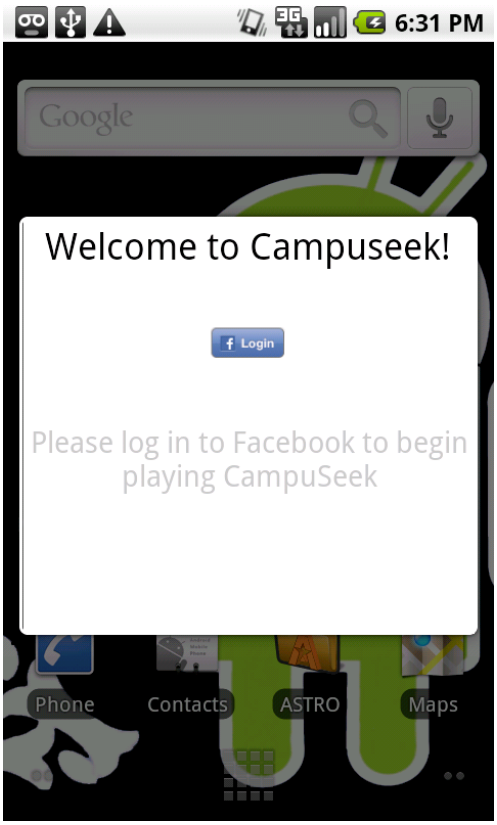
Hotter / Colder Test
The hotter/colder feature was tested simply by walking with the phone toward and away from a known destination. The update time was observed, as well as the accuracy of the hotter/colder indication. The hotter/colder feature was found to update very frequently – as often as GPS updates are received. We put a limit of a maximum of one update per 1.5 seconds in our program. Therefore, we experienced hotter/colder updates averaging about every 1.5 seconds as expected. The fast update time did result in some oscillating between hot and cold when the user is walking slowly. Overall, the feature performed successfully, correctly indicating the direction the user is walking (closer or farther from the destination) with a rapid update rate.

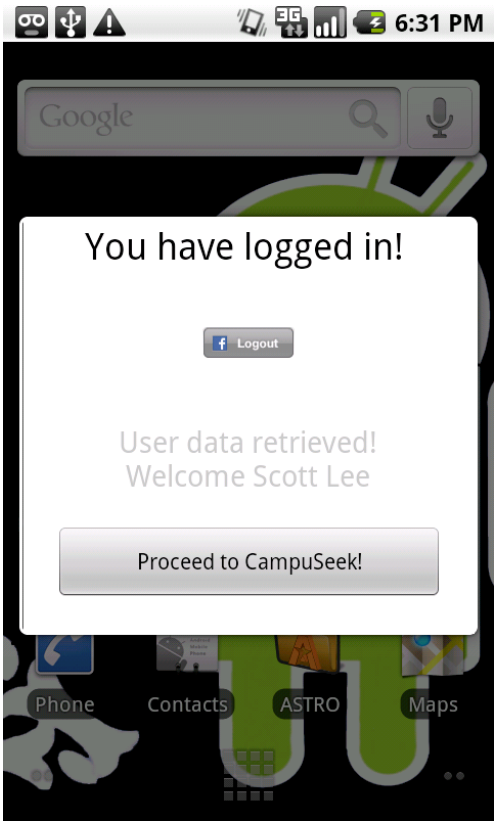
Intuitiveness Test
To test the application for intuitiveness, several friends were asked to play the game, without prior knowledge of the application interface. Each user was able to install and play the game with minimal help. The Facebook application has 17 users, who have found the application page and registered with the application. The Facebook interface is a simple menu interface that has been rated by classmates as being very intuitive to navigate.

Results

Android

The main login screen can be seen below. After the user chooses to login, their data is retrieved from the databases and a welcome message is displayed as can be seen below.





After the user data has been retrieved, the user can proceed to the game environment. The “Challenges” tab is the initial view shown to the user. It is set to show challenges for whichever level the user is currently on. Different levels can be selected using the Spinner widget as shown in the images below. Notice in Figure 15 that the “Advanced” option is not available, because the user is only on the intermediate level.

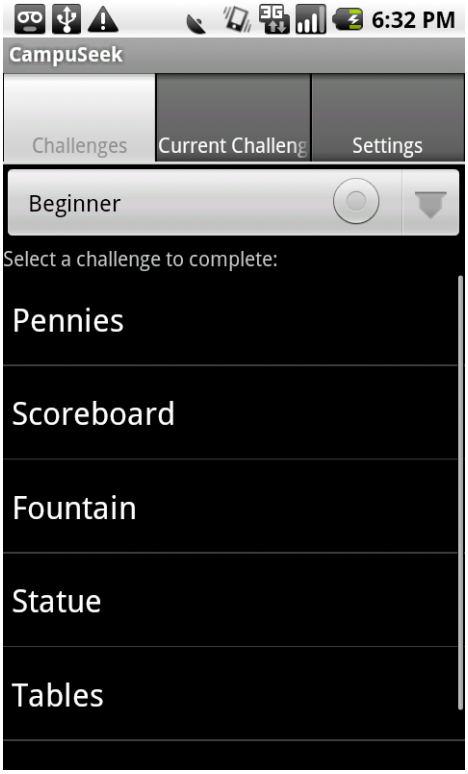


Figure 13: Challenges - beginner

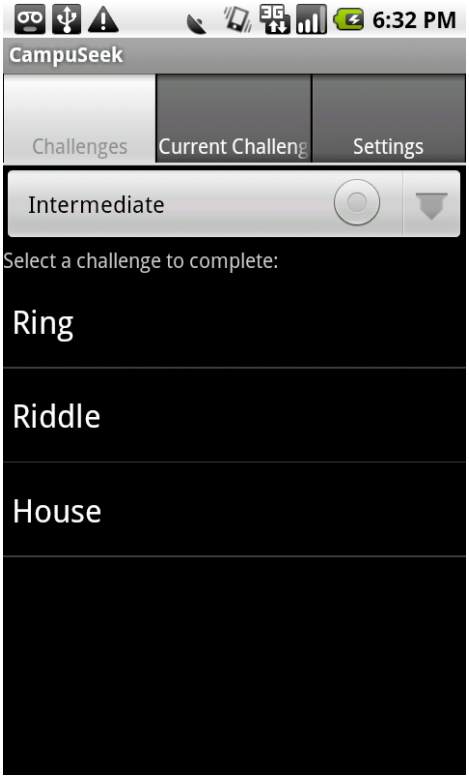


Figure 14: Challenges - intermediate

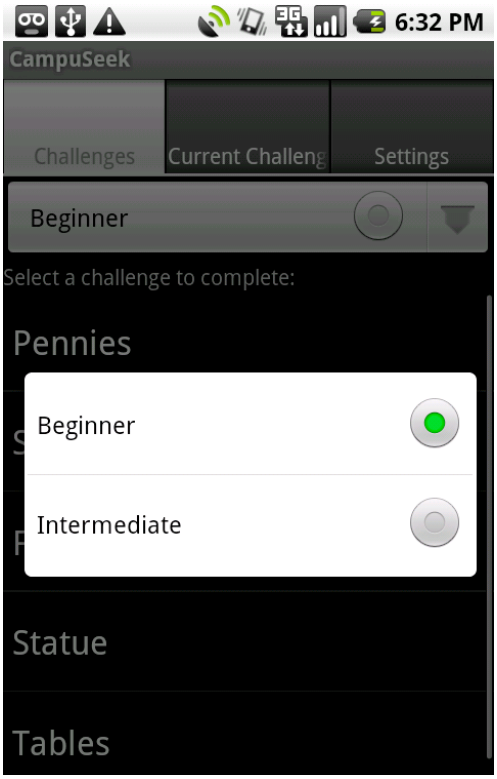


Figure 15: Level selection, advanced not available for

intermediate

When the user selects a challenge, they are taken to the “Current Chalenge” tab. This tab correctly displays the challenge image and description, which are pulled from the database, as shown below in Figure 16.

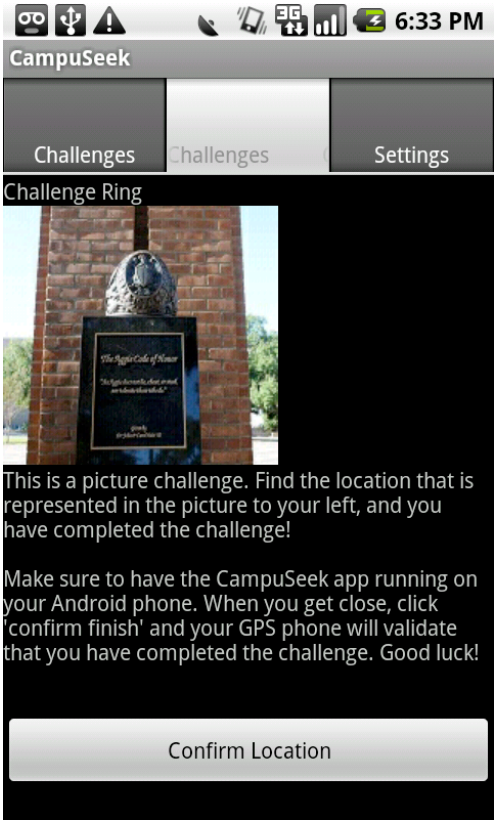


Figure 16: Current challenge page

Several features are accessible through the use of a menu, which is enabled with the Android *menu* button. The menu is contextual to whichever view is currently displayed. The menu for the *Current Challenges* view is shown below in Figure 17. As can be seen, the *hotter/colder* feature is accessible through the menu. This feature can be seen in action in Figure 18 and Figure 19 below.

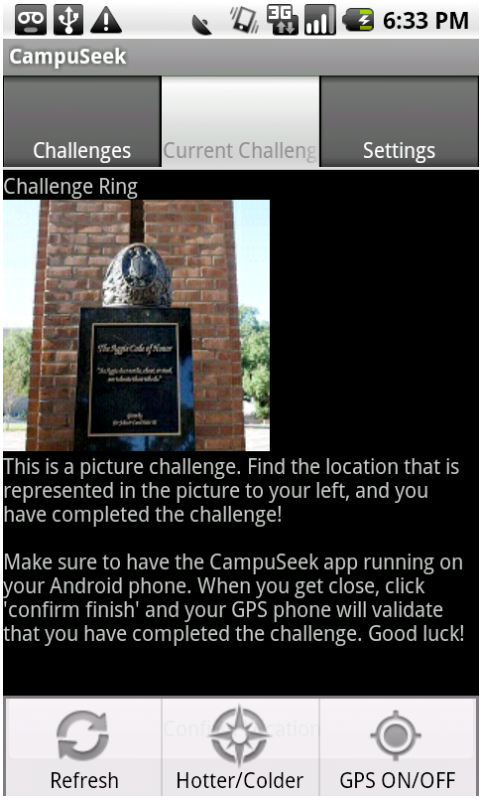


Figure 17: Menu

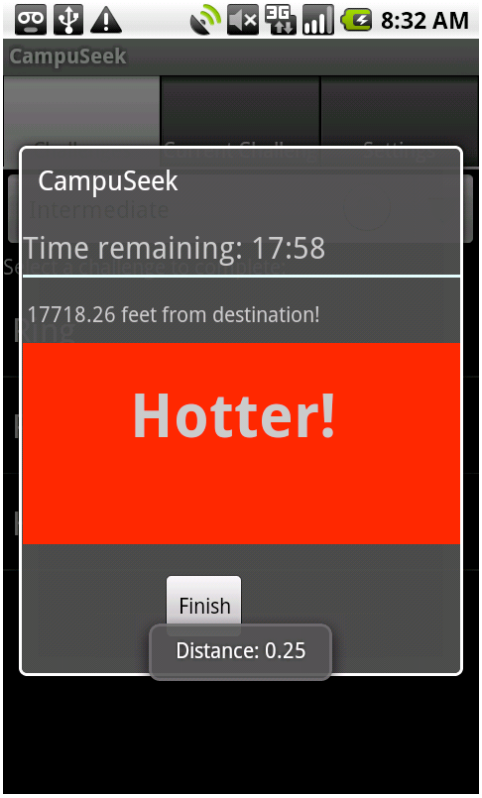


Figure 18: Hotter/Colder - hotter

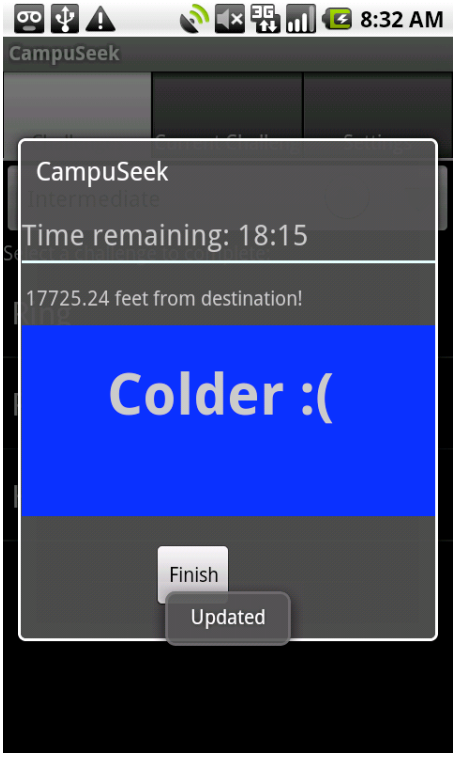


Figure 19: Hotter/Colder - colder

The contextual menu also offers the ability to start and stop the GPS service. This is to address social and privacy concerns of users playing the application who wish to disable the GPS when not in use. Finally, the success and failure challenge verification screens are shown below in Figure 20 and Figure 21.

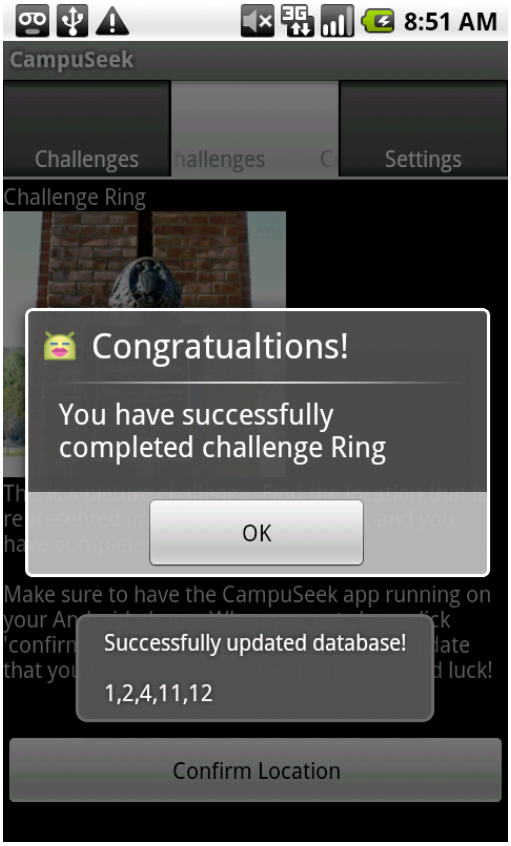


Figure 20: Success

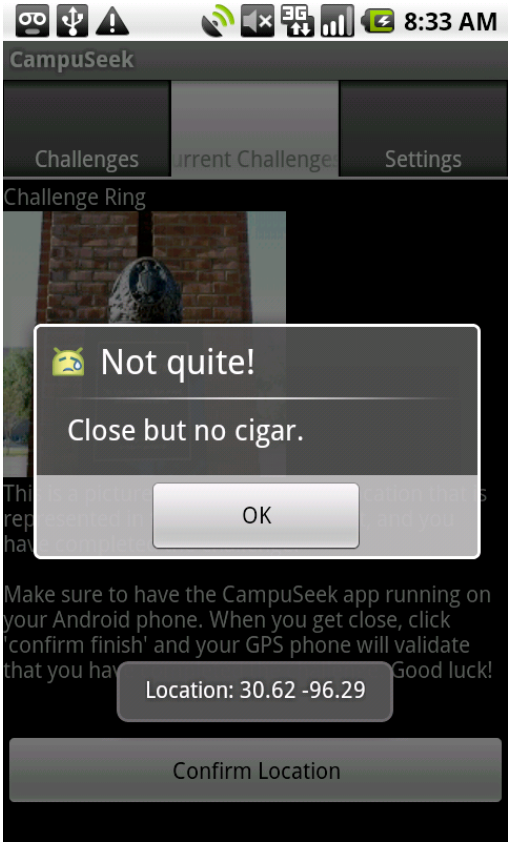


Figure 21: Failure

Facebook

The Facebook application homepage is shown below in Figure 22. It features instructions for beginning playing the game, as well as a Facebook news-feed module.

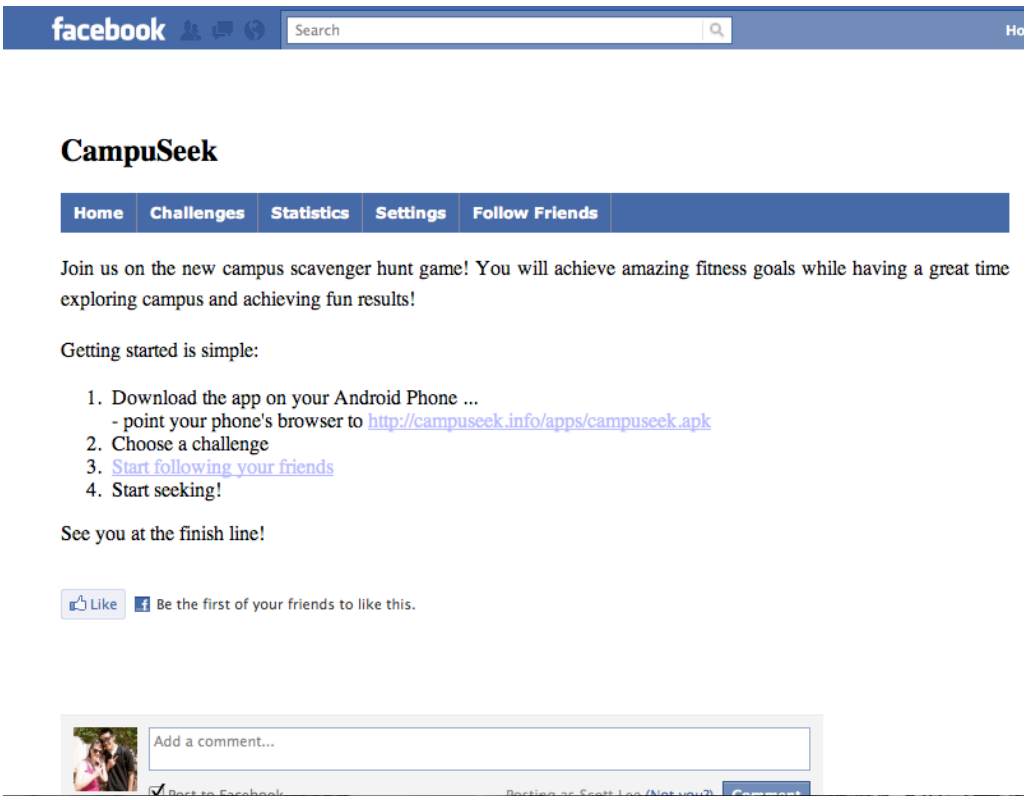


Figure 22: Facebook homepage

The *Challenge* page is shown below in Figure 23. It lists all of the challenges, sorted by beginner, intermediate and advanced. Users can only see challenges up to their current skill level. The *Follow Friends* page is also shown below in Figure 24. On this page, users select which of their friends they would like to follow on the application.

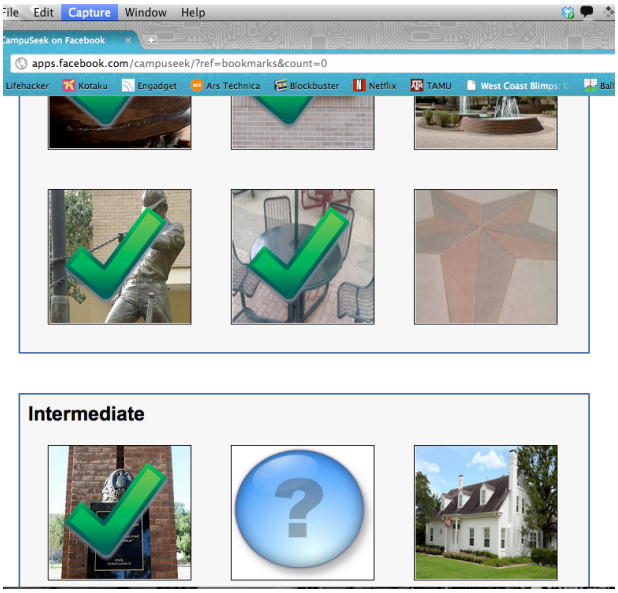


Figure 23: Facebook challenges page

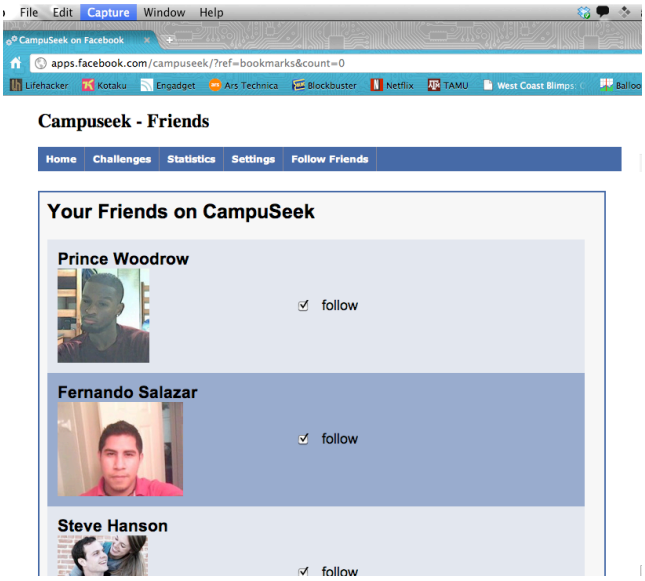


Figure 24: Facebook follow friends

The workout statistics page is shown below. In Figure 25, user statistics can be seen for distance travelled and calories burned, arranged by week. In Figure 26 on the right, friend statistics are shown.

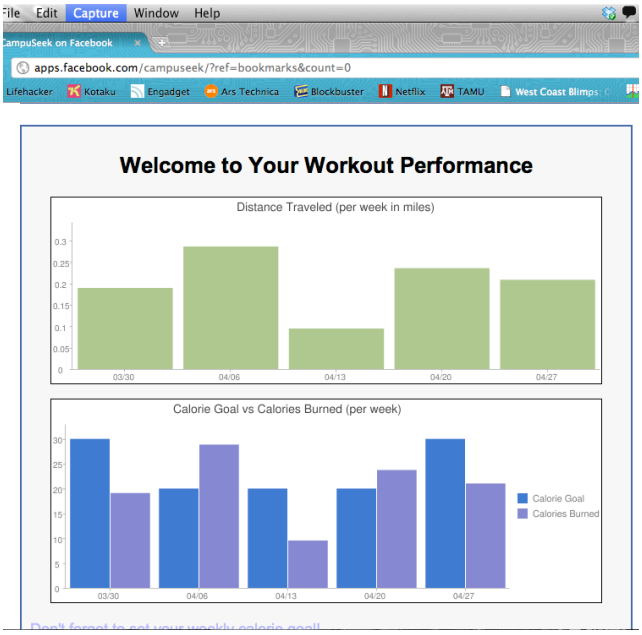


Figure 25: Facebook health statistics

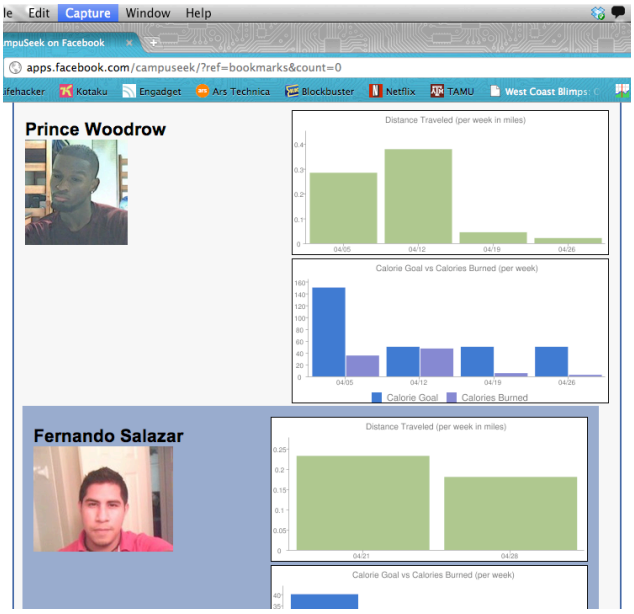
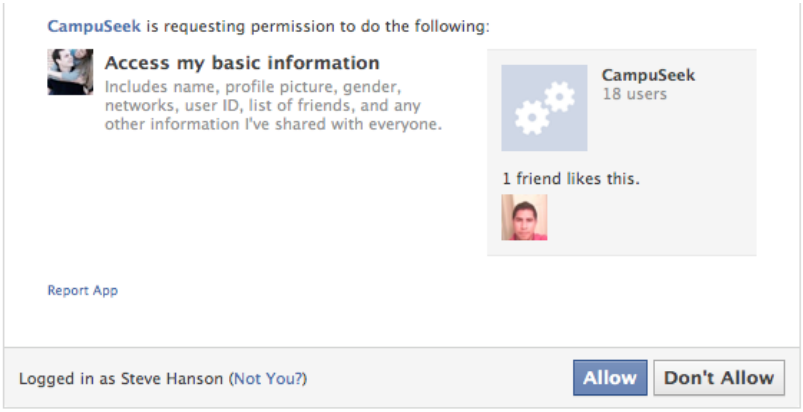


Figure 26: Facebook friend statistics

6 User's manuals

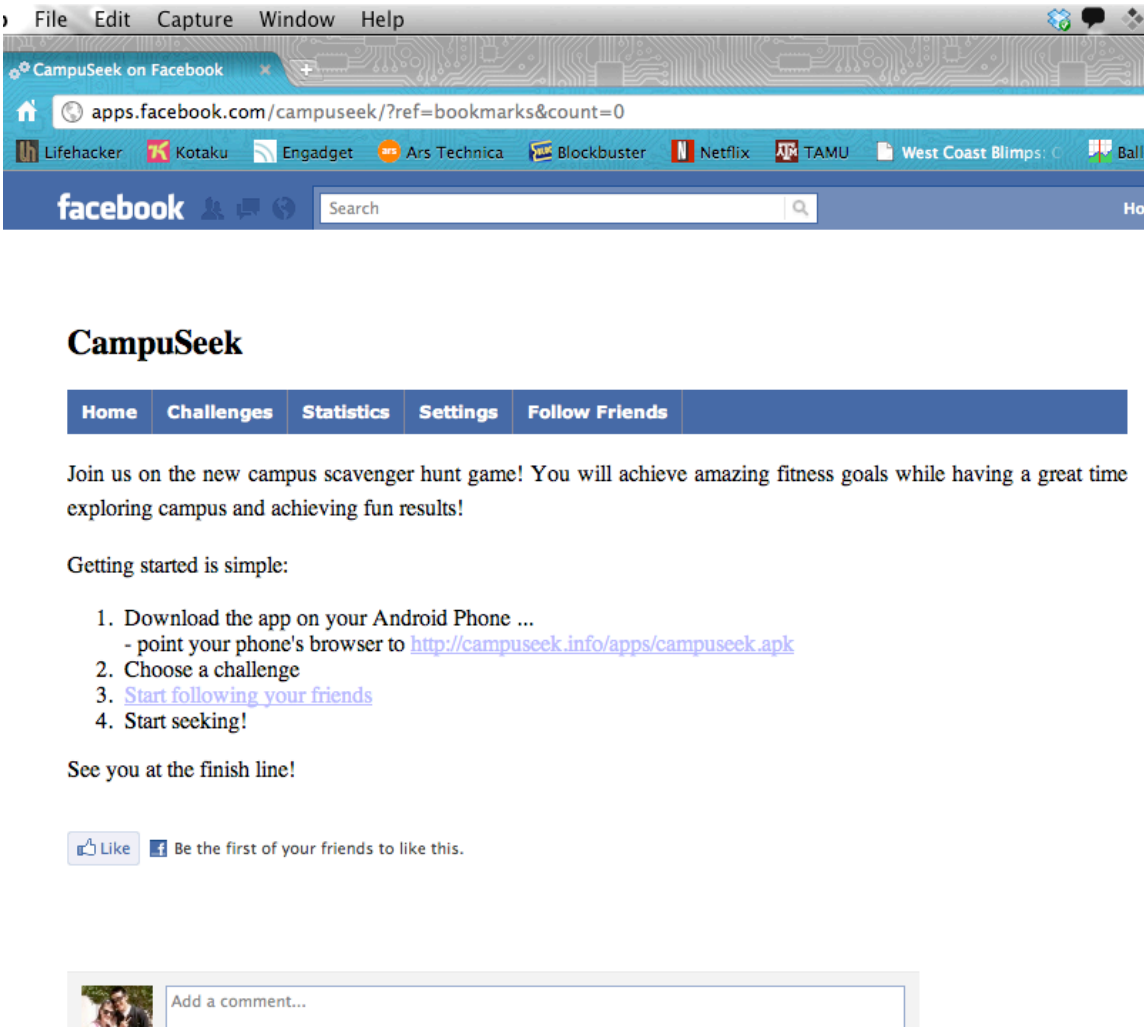
CampuSeek is mainly a software system, with the sole hardware dependency being the user's Android-based mobile device as well as a personal computer or any other device that they can use to access their Facebook account.

There are two software components that seamlessly interact with each other and both are readily available for the public to obtain – the Android application, and the Facebook application. The user can start by downloading and installing the Android application or by registering with the Facebook application. If the user chooses to first register with the Facebook application, they can simply search for CampuSeek on Facebook's main search page, and it will lead them to our Facebook Application that they can choose to visit (alternatively, users can go straight to http://apps.facebook.com/campuseek). Once they do this, they will be prompted to allow/not allow the CampuSeek application in a dialog similar to that shown below:



Code Snippet 5: Facebook authorization request

The CampuSeek Facebook application respects user privacy by only storing the user's Facebook ID and name. No data is stored that compromises user identity, nor is any data shared or available to third parties. Click "Allow" to proceed to the application. At this point, the user is registered with the application and can begin viewing challenges and following friends. They are initially taken to the home page as can be seen below. The next step is to download the Android application and begin solving some challenges!



Notice that instructions are given on the Facebook page for getting started with the Android portion of the application. These instructions are simple to follow and are repeated here in more detail:

Getting started on Android:

1. Point phone's browser to http://campuseek.info/apps/campuseek.apk
    a. Download will begin automatically
2. Select the newly downloaded file to begin installation
    a. Installation should only take a few seconds
    b. The user will be prompted to allow/not allow the application
        i. Click ALLOW.
    c. If the user has not enabled third-party applications (those not from the Android Marketplace), on their phone, they will be automatically prompted to do so with a warning. Follow the instructions to enable the application installation
3. The application is now installed! Log-in using the Facebook Login button
    a. Users must be registered Facebook members as login with Facebook is the only method currently supported

CampuSeek strives to have an intuitive user interface so that it appeals to anyone, even those who might not call themselves "tech-savvy". Our main audience are those who are striving for motivation to exercise or those who simply want to be active while having a good time. From the Facebook Application, once a user has registered, they will be asked to enter submit their height and weight so that an accurate measurement of the number of calories burned can be be recorded. They will also be asked to add their goal of how many calories they want to burn a week so that their progress can be graphed alongside their goals. The Facebook layout is quite easy to navigate as the different headings are self-explanatory. The user can view the challenges they have available/completed, find their friends who are playing the game and view those health statistics as well.

The CampuSeek Android application is no different as it strives to be as intuitive as possible so that there is a minimal learning curve. The main features include viewing challenges and selecting the challenges they want to complete. Upon opening the app, a user will be asked to log into Facebook using the Single-Sign on method. This only occurs once as the phone will remember this until the user logs out. Upon data retrieval, the user will then be taken to the Challenges page. Here a user can select any challenge they wish to complete as well as use the spinner to select one of three levels. Initially, only beginning is available, but as they complete more challenges, the other two levels will open up. When a user believes they are within proper range of the challenge, they will select the confirmation button on the current challenge page and an alert dialog will appear, notifying them of their success or their failure. To view already completed challenges, users can easily navigate to the Settings Tab to view a numerical list of the challenges done.

Added features include toggling the GPS service as well as a bonus feature, hotter/colder time. These are both accessible by hitting the menu button on the device and selecting the respective item. Initially, the GPS service is started with the application, so if the user wants to turn it off, they simply press "toggle GPS" on the menu list. To turn it back on, they hit the same item. With hotter/colder time the use will choose the hotter/colder item on the menu. This brings up a window that will turn red/blue as the user gets closer/further from the destination, respectively. To exit out of hotter/colder time before time is

expired, the user can select “finish” and their timer will be paused until they choose to use this feature again.

7 Course debriefing

Team management

Throughout the course of this project, our team worked well together, which is what enabled us hto succeed in our project. Openness was encouraged in the group and polite disagreements were common. Nevertheless, the team always came to a conclusion and agreed on an approach for all of the tasks faced. Work was divided up evenly between all members, including the team leader. There were two main sub-groups: Android development and Facebook development. Each group was assigned two team members, with one member eventually moving to Android development once Facebook was nearing completion.

Thoughts on repeating the project

If we were to repeat this project, we would do most things similarly. Throughout the course of the project, we were faced with design decisions, which we sometimes went back-and-forth on, but in the end, the choices our team made were informed and worked out well. If we could go back and change something, we probably would have put more of an emphasis on testing in the end so that we would have more quantifiable data to base our results analysis on.

Safety and ethical concerns

If future expansion allowed for users to add challenges, then concerns would arise pertaining to the safety of the challenge destinations. The challenge destinations that are currently in the game database are public areas of campus. Still, some areas should not be travelled to alone or at night. Also, users of the game should verify that they are fit for physical exercise before attempting any strenuous activity

Testing the project

Our project unfortunately did not undergo as strenuous of testing as our group would have liked. This is for several reasons. First of all, we lacked resources to perform large-scale testing. Few of our friends that we could call in for testing have Android phones, and our group only had access to one phone with GPS. This made it difficult to perform much testing of the Android application on the field. Also, we did not envision and set up a comprehensive test approach for testing workout data or challenge goals. Our application allows users to set calorie goals, but we were unsure if any of our test subjects would really be playing with the intention of burning calories – we needed real users who were trying to meet goals. Because we feel like our application works with very few flaws, we still feel good overall about the amount of testing done. If we could go back, we would just test the concept of a health-promoting game such as this in more detail by analyzing results.

8 Budgets

This project was very inexpensive to develop and maintain. The main hardware components for this project were Android phones that were donated to the school through Verizon. The software API’s that were used were provided free of charge and the programming IDE’s were free as well. For Android development, all code was written in Eclipse which is a free java IDE and the Facebook Application was written in various free text-editors and then transferred to the CampuSeek server using free FTP clients (Cyberduck and WinSCP). Initially, our team tried to use a web hosting service with free database access that a team member already had, but this limited the amount of PHP functioning. To accommodate the necessary scripts that were needed for uploading and downloaded information from the database, a paid web host subscription service was required. This was fairly inexpensive and cost a total of $11.95 for 3

months of use. The itemized breakdown of project costs is shown below:

Item	Estimated Cost
Android Phone (x3)	$0 (donated)
Web Host Subscription	$11.95
Total	**$11.95**

This is actually cheaper than our proposed budget. Initially, we were under the assumption we would be getting more development phones and were going to have peers review our product. To make it easier on them while they played the game we would have gotten phone holsters so they would not have to physically carry the phone in their hands. However, since only three phones were given, the majority of testing was conducted within the team and phone holsters were no longer needed. This saved an estimated value of $150.