CS 4720 - F17 - Final Project Documentation

Device Name: Yanma Platform: Android

Name: Timothy Davison Computing ID: tfd2xq

Name: Seven Starosta Computing ID: sbs3bx

App Name: CacheMapper

(Project proposal section)
Project Description:

Our app, CachMapper, will provide a connecting service through which geocaching hobbyists can digitally connect and participate. Geocaching is a popular outdoor recreational activity involving the hiding of and visiting of cache sites around the world, where a logbook/ any items are stored in a waterproof container and hidden in an unmarked outdoors location. Our app will facilitate that hobby by providing local cache sites through GPS, allowing cachers to add their own sites (including pictures and descriptions of their caches).

What we propose to do is create an app that will do the following:

- The system shall allow a user to create a profile using a username and password.
- The system shall allow a user to access a GPS generated map of cache sites worldwide;
- The system shall allow a user to log their visits. This will be shown on a view cache page, where the visited state of each cache is stored and displayed.
- The system shall allow a user to create their own cache site, complete with a picture and a description, and register that cache site with the app. They shall be able to store caches they make in the cloud.
- The system will allow users to send emails to each other from the cache viewing page

We plan to incorporate the following features:

- GPS/ Location awareness: the application will give the user access to a google generated map indicating their position relative to registered caches. GPS will also be used when creating a cache at the current location.
- Camera: the application allows users to access their camera when registering a cache site for taking pictures to be associated with that cache's page
- Build and consume your own web service using a third-party platform - cache information will be stored via the web service Firebase.
- Data storage using key/value pair storage - We will store the user's account information along with other basic information on the device.
- Open shared activity / features - We will provide the user with the ability to email the creators of other caches from the view cache screen.

Wireframe Explanation:

(see wireframe.pdf in the root of the project directory).
The first page is a login screen, where the user inputs an email and a password. If the username has not been used before, they are registered now, otherwise it is checked with the stored username/password combination.
The second page is the main screen with a map, which will show caches, and has an add button which leads to the add cache page. It also leads to individual caches by tapping.
The add cache page allows for a name, a username, and a picture.
The view page shows the name, username of the creator, and the picture.

Platform Justification
Mobile/android offers the clear advantage of being portable, which is essential to the task of creating geocaches dynamically out in the field. Mobile also has built in GPS which can function without internet (unlike a laptop), which furthers its suitability for this application. Using a phone/tablet allows the user to easily take their device off road, as well as to take pictures to include in their cache descriptions using a phone/tablet's outward facing camera.

Main Screens:

Login: here the user can log in using a gmail account and a password chosen specifically for the app (i.e. not their actual gmail password). The username and password pairs are stored into system preferences, and the currently logged in username is passed onto the main activity screen if login is successful. We chose to use systemprefs for username/password because it could function without any internet access.

Main Screen / Map Screen: Here caches automatically downloaded from firebase are displayed as red X's on the google map. The user's current location is also shown by a blue marker. The map fragment is set to not be destroyed upon rotation to prevent the time and data cost of reloading the map on rotation. The geocaches are also cached (haha) by firebase locally here, meaning that if the user loses connection to the internet they can still access the caches on the view screen after entering from the main activity. Note that the images associated with each cache are **not** cached in order to prevent excessive space usage by the app. The main screen has a logout button which takes the user back to the login screen, which wipes the password from the text field. The main screen also has a blue add cache button which takes the user to the add cache screen.

View Cache Screen: this screen shows the information for the cache that the user has tapped on. This information is as such: the username of the creator of the cache, the name of the cache, the description of the cache, the number of users who have visited the cache, whether the current user has visited the cache displayed as a checkbox, and an optional image associated with the cache. The visited checkbox will be preset according to whether or not the user has been recorded as having visited the cache on firebase. If the user taps the checkbox,

firebase and the number of visitors to the cache are appropriately updated. On the bottom of the screen there is an email button, which allows the user to email the creator of the cache in order to inquire about hints etc. The email activity is opened with the to: and subject: lines prepopulated specific to the geocache. The view cache also has a back button to the main screen. While this screen successfully rotates, it works best vertical in order to let the cache image display biggest.

Add Cache Screen: on this screen the user can give a name, multiline description, and choose to take a picture or choose an already taken one. A picture is optional and the cache can upload to firebase without a picture. The username will be automatically included, having been retrieved through systemprefs. The cache will also be automatically marked as visited by the creator, and will thus have an initial visitor count of 1. Upon creating the cache, a dialog prompts the user to return to the add cache screen.

Optional Features:

Build and consume your own web service using a third-party platform: firebase (database): We used firebase database and firebase storage to store information on the caches. Note: if you consider firebase storage (for images) and firebase database (for strings and other more primitive objects) to be separate web services, please evaluate just our implementation of the database. We used the database to store the cache name, cache creator username, latitude, longitude, description, number of visitors, as well as the caches visited by each user. We implemented the database such that it will continue to work without internet access, provided that the data was already downloaded beforehand. The data should also sync back up to firebase once the app regains internet access.

We used firebase storage to store the images associated with each cache. Note that images are considered optional, and we have created one cache already showing this functionality. Additionally, in order to save space on the user's device, images are not cached. They will be preserved upon rotating a screen, but killing the view activity when there is no internet access will cause the image to be wiped.

For grading this, you should consider the quality of uploading, statelessness (such as going to the map screen with wifi, turning off wifi, then going to the view cache screen, as well as correct syncing when getting back to wifi), the appropriateness of using firebase for this project, and downloading (both onto the main map screen and onto the view cache screen).

Data storage using key/value pair storage (SharedPreferences): We used this to store the username and password combinations, as well as the current logged in username. The current username is required to be a gmail, but is just stored as the first segment (i.e. everything but @gmail.com). We did this in order to be able to name nodes in the firebase database by username, since firebase cannot have periods in node titles. We use the username pulled from system prefs for uploading new caches as well as for checking whether or not the current user has visited a cache on the view cache page.

For grading, consider successful putting and retrieving the above info from system prefs, as well as the quality of the login screen (i.e. you can only log in if username and password

match, and username and password are checked to see if they are valid to prevent problems). Also note the choice of using systemprefs allows the app to function without any internet, provided that some firebase info has be downloaded by a previous use and cached.

Camera: we use the camera and/or the image selector to choose an optional image to associate with each cache upon cache creation. Grading: does the camera work? Does the use of a picture as a hint for a geocache make sense?

GPS/google maps: We use a google map as the main screen of the app, where existent caches are shown around the user. The map is centered on the user's current location upon initialization, but does not snap back automatically in order to allow the user to roam. GPS is also used on the cache creation screen, as caches are always created at the current location. Grading: note that the map is preserved (i.e. not recreated) on rotation. Also note the use of custom markers, as well as the successful saving/upload of current location from the add cache page. Note that the current users location is updated between every 1 and 10 seconds, and the respective marker location is updated. The long interval of 10 seconds was chosen to save battery life, as the gps only shows an approximate closeness of the user to nearby geocaches.

Open shared activity: email. On the view cache page, there is a button that allows the user to send an email to the creator of the cache. We added this feature in order for the user to be able to ask the creator for clarification or hints on finding the cache. The email will be prepopulated with the cache maker's email in the mailto: field and with the current cache's name as the subject line.

Testing methodologies: Our testing was primarily by using the app itself on the tablet in person, trying out various corner cases as well as the main functionality. We turned wifi on and off, left the app and came back, and rotated on all of the screens and in various states in order to ensure correct functionality. We made several users and tested the functionality of storing who has visited each cache, as well as counting the number of visitors. We created caches with and without images, doing this with and without wifi. We checked to see correct syncing after reconnecting to wifi. We attempted to log in with invalid credentials, and then separately using the wrong username. We actually sent emails using the email send button.

Usage: usernames: sevenstarosta@gmail.com, password: seven
Username: tim@gmail.com, password: timtim
You are free to make your own new username/password combination.
The app is designed to work with GPS turned on.
Rotation is functional on all screens but the app looks best and is designed to be held upright on the add cache and view cache pages.

Lessons Learned: Though choosing a premade platform for a web service can save lots of resources, you should definitely be sure about how it will function and what its limitations are. We did not have any access to server side code for firebase, which ultimately put restrictions on what our app would be able to do in the real world if thousands of people were making caches.

We had little choice but to require each app to download all of the data (though not the images) from the database because we could not easily sort the data on the server end by latitude and longitude. We thus couldn't create a feature of, say, downloading only the caches within a 30 mile radius of the user's current position. Firebase could also make code for correctly allowing incrementation/decrementing variables on the server end tricky, especially when there is no internet access. This came up when we were making the code to keep a count of the number of users who have visited each cache. Firebase also would not allow us to use certain characters in the creation of usernames, which could cause problems when attempting to account for which caches each user has visited. We also learned that updating the locations of markers on google maps requires storing references to them somewhere, which could require a bit of overhead code. We thus decided to not let users update the location of caches in order to avoid substantial synchronization and updating code for the marker locations.

Overall, we learned the importance of planning in mobile development, as well as the use of relatively modular code that could be swapped out at later stages in development.