UV Guardian B Final Report

Introduction

The purpose of this software design document is to provide a low-level description of the UV Guardian application, providing insight into the design and structure of its components. The overall purpose of this of this document is to provide the reader with adequate knowledge of how the application works and how it was designed.

Topics covered include:

- Design considerations
- Architectural strategies
- System architecture
- System policies
- System design

This document is intended for readers with an interest in the low level design of the UV Guardian application as well as readers who need an understanding of the design and architecture of the application in order to work with its code.

For knowledge of the requirements of the UV Guardian project, please refer to the System Requirements Specification.

System Overview

The UV Guardian application was designed modularly to allow for multiple teams to work on different aspects of the project. The application utilizes UV sensor technology paired with database management systems and Facebook integration in order to provide the service it was designed for. As this document pertains only to the contributions of UV Guardian team B, it will focus on the implementation of the Facebook integration and Parse database system.

Facebook Overview

- UV Guardian uses the Facebook API in order to provide the user with the ability to log in to Facebook. The app will then search for all of the user's friends who are UV Guardian users and will add them to the user's UV Guardian friends list.
- Planned future implementation includes the inclusion of friend profile photos to represent the user's friends on his/her UV graph
- For more information see (Facebook Servers under **System Architecture**)

Parse Overview

- The Parse database system allows the application to store and request data on UV Guardian users. This allows the application to search for a user's existing Facebook friends in the database and to also view each user's UV readings.
- The application does not allow users to create an account separate from Facebook.
    - Currently, the user must have a Facebook account to use UV Guardian and the application will use the user's Facebook information to populate the parse entry
- The application gets data for the database from the UV sensor (for more information on the UV sensor see UV Guardian A report)

Design Considerations

Approaching the assignment of creating an UV Exposure Graph, there were many different design considerations. The initial requirements wanted a scatter plot of sorts with user Facebook avatars as the data points. The scatter graph would be plotted according to the relative percentage of the UV Exposure. However, there was another problem found with this method of plotting the data points. If the data was plotted according to the relative percentage, one user would have to serve as a maximum user, meaning his data point would always be at the top of the graph compared all the users. Additionally, if one user had a lot more of UV Exposure compared to other users, then the relative percentage would be useless in comparison since all the other users would be plotted very low in the graph. This would cause actual comparison between other friends to be impossible on a small screen such as an Android mobile device. In the end, we decided on implementing a bar graph and limiting the amount of friends you can compare to 4. This allows a simple, easily recognizable comparison, between multiple friends. The data used for the bar graph is the total accumulated UV Exposure over a certain period of time.

Another design consideration we had to make was that of how to set up the fragments responsible for choosing friends and the fragment for viewing the graph. Initially we had a Menu screen of sorts that allows you to go to another fragment to choose friends. The user would then have to go back to the menu fragment and then choose "Show Graph". We decided that this implementation was overly complex and didn't require the in-between menu fragment. We opted to remove the Menu fragment in favor of just directly redirecting to the fragment containing the Facebook Friends. A button, "Show Graph" was placed at the bottom of the graph. This allowed the user to easily switch between both the contact list and the graph fragment. This also simplified our code a great amount, and made it more readable and understandable. With the previous menu fragment, the code was complex due to a lot of data needing to be passed between multiple classes/fragments.

Architectural Strategies

Programming Languages

UVGuardian is built on top of MyRuns5, a pre-existing application that monitors exercise activities. For this reason, the programming languages used such as Java and XML are kept the

same as the original application. No other programming languages were considered, as it would be impractical because the original application being built upon was already done in Java.

Database

The database management used is Parse, the third party database system that is used, which runs on NoSQL. Parse was used due to its convenience and how it seamlessly fit into our design, since it already gave us simple ways to do what we needed from a database system. It manages all user data from Facebook, as well as personal information that about users such as UV Index. This was an easy choice for us to make regarding the usage of Parse, because it saved us a lot of time by not having to implement our own database that may or may not have worked as well as Parse.

Libraries

AChartEngine is an open-source Android chart library that was chosen by us to use because it fit nicely with our needs in the relative exposure graph. Many other chart/graph libraries were considered and tried, such as ChartView, but in the end AChartEngine was the better choice, because its features fit our needs better than the others. These features included an easier interface to use to create bar graphs in AChartEngine vs. ChartView.

System Architecture

UVGuardian app

The main UVGuardian app allows the user to compare their UV exposure levels to their friends', in order to get a better idea of how much exposure they are receiving. It does this by getting a list of the user' Facebook friends who also use the app, then querying the Parse database to find their UV exposure levels. The user can then select up to four friends to compare exposure levels with on a bar chart.

Facebook servers

Facebook is a social networking system and is the platform by which our app connects users. We use the Facebook Android API to allow the user to log into their Facebook account and add Facebook friends who also use the app. The user logs into Facebook upon first starting the app, and the app pulls the list of Facebook friends who also have UVGuardian and displays it on the screen.

Parse database

Parse is a free platform for data storage, especially designed for mobile applications. Data is stored on the servers in the form of a JSON-compatible ParseObject. The UVGuardian app uses Parse to store a user's name, Facebook id number, and a list of Facebook friends. UVGuardian can pull data from the server whenever it is needed, and can use it to find other UVGuardian users on the friend list. This is done by searching for other ParseObjects with Facebook usernames and ids that match those in the user's friend list.

UV sensor
The UV sensor is a separate module that calculates UV exposure levels and sends the information to the app. This module was developed by Team A.

Policies and Tactics

As a group we tried to implement multiple processes and coding practices that we learned in CS 130. One such practice was the use of external tools such as Pivotal Tracker and Github to manage workload and have a version control program. Pivotal Tracker was used by the group to create jobs and assign workload to different members of the group. Such workload was usually attempted to be completed the same week it was assigned, however that was not always the case. Additionally, Github was used extensively as our code repo. We had multiple problems with Github mostly because most of us did not know how to use it. Due to our lack of familiarity with Github, we ran into numerous problems with merging code and just using the tool in general. These problems were very costly in the time we spent in trying to resolve issues and correct our code.

Another practice that we tried to attempt was pair programming. We found that it was very useful in creating functional code. Since none of the group had programmed in Android before, we lacked basic understanding of the language and the structure of Android software. However, with pair programming, it was noticeable that more functional code was created in a shorter span of time compared to people working alone. This was usually due to one person researching concepts such as Fragments and Views and explaining and guiding the actual person typing the code. This sped up the process of learning how to program in Android and also allowed for a faster debugging process due to two sets of eyes on the code.

Detailed System Design

1. LoginActivity
    a. Classification
       Android activity with the use of Facebook libraries. This counts as a module in that it is one of the stages of the application.
    b. Definition
       This activity logs the user in using Facebook.
    c. Responsibilities
       The code is found in an Activity called LoginActivity which uses Facebook's API to display a login page, allow the user to sign in, and request permission for user information.
    d. Constraints
       The user must log in with Facebook to make it past this activity.

e. Composition
   This module is does not contain any subcomponents as it only has the purpose of asking the user to log in using Facebook.
f. Uses/Interactions
   The Facebook Login feature comprises of a splash page which the user sees if he is not logged in using facebook, and this page is removed once the user has logged in. This immediately calls User Details by switching intents, once the user has successfully logged in.
g. Resources
   Uses the Facebook SDK extensively.
h. Processing
   If the user opens the application when already signed in, then the application will remember this saved state and skip the login splash page.
i. Interface/Exports
   The login page centerpiece is a Facebook login button which for many users takes the place of any login text editor, as the user likely already has Facebook installed.

2. UserDetails
   a. Classification
      The User Details feature is an Activity class.
   b. Definition
      This class saves the user's details on the Parse database.
   c. Responsibilities
      Using the facebook user information retrieved from LoginActivity, this class saves the information about name, facebook id, and a selection of friends.
   d. Composition
      This class also contains the function getFacebookFriends which is implemented by making a list of the user's facebook friends and comparing their facebook id's with those stored in the parse database. The intersections are stored as a List of strings.
   e. Uses/Interactions
      This class creates an ParseObject on theParse server of the following form:

      ParseObject user
      {
       string name
       string fb_id
       List <String> fb_friends

}

If the user has not logged in with Facebook, then this class will call Login Activity. Upon completion of this activity, this will call an intent to go back to MainActivity.

f. Resources
In addition to using the Facebook SDK, this also uses the Parse API.
g. Processing
h. Interface/Exports
There is no user interface for this class as this is all done "behind the scenes," but this does make changes on the Parse servers online.

3. FriendsFragment
    a. Classification
    FriendsFragment extends a ListFragment (an Android Fragment object).
    b. Definition
    This Fragment class enables users to view Facebook friends' UV info on a graph,
    c. Responsibilities
    The primary responsibility of FriendsFragment is to list all of the user's Facebook friends that also happen to use UV Guardian. The app user can simultaneously view the UV exposure details of up to a certain number of friends and compare their levels of radiation with his or her own.
    d. Constraints
    Notice that there exists a maximum number of friends that can be selected for comparison (default of four). The smartphone must also be connected to a consistent internet connection at the start of the app (where this class is created and populated by Parse) and when selecting and adding friends (where the friends' names are fetched from Parse). Otherwise, the app may not function correctly until it is restarted under required conditions.
    e. Composition
    FriendsFragment makes use of a ParseUser object and Facebook login credentials for obtaining UV information on friends that also have the app installed.
    f. Uses/Interactions
    By clicking-and-holding a Facebook friend on the ListFragment, the user can toggle (add/remove) that friend from showing on the graph. A toast message is shown indicating which action was taken. A "Show Graph" button at the bottom of this Fragment will render the resulting graph for the user.
    g. Resources
    Both Facebook SDK and Parse API are used to fetch and process friend UV info.
    h. Processing

This component for adding friends fetches user data from Parse and filters it to obtain an ArrayList of relevant friends, which in turn populates the ListFragment itself. Toggling a friend will either add or remove their ID and array index to another ArrayList.

i. Interface/Exports
The ArrayList above, which contains an added friend's ID and position index, is essentially passed on by Intents to the BarGraph for display purposes.

4. BarGraph
    a. Classification
    BarGraph is an Android Activity based on AChartEngine, an open-source Android chart library.
    b. Definition
    This Activity plots the UV exposure levels of selected friends on a bar chart.
    c. Responsibilities
    BarGraph's purpose is to fetch Facebook friends (within maximum specified) included in the ArrayList that is passed in, then graph their UV exposure levels on a simple bar chart for the user to compare with his or her own. The user's own UV information is always displayed on graph, regardless of how many friends are selected.
    d. Constraints
    BarGraph depends on initializing a ParseUser for grabbing and displaying user information from the Parse database. Internet connectivity is required for obtaining and displaying this UV information.
    e. Composition
    This Activity uses AChartEngine's BarChart class and it requires a ParseUser.
    f. Uses/Interactions
    The BarGraph is constructed when the user clicks the "Show Graph" button from the FriendsFragment before. Movement of the graph and pinch-zoom functionalities have been disabled on the bar chart in order to preserve an ideal view for the app user. To return to the friends list or main screen, use the phone's Back button.
    g. Resources
    The AChartEngine library and Parse API are utilized heavily.
    h. Processing
    On creation, the overall layout of the bar chart is established and user UV information is updated depending on the number of friends selected. Some layout details, such as the positioning of axis labels, are implemented in order to fix known shortcomings in the AChartEngine bar chart library. The series color of

each individual user's bar is drawn from an array meant for storing constant hexadecimal color codes.

i. Interface/Exports

The ArrayList passed in contains the index positions of selected friends as well as the ParseUser data. Using the index, we can determine which ParseUsers must be queried for UV information for displaying on the graph.