Fitbit iOS Fitness Application

by

Connor Mashburn

Advisor
Daniel Plante

A senior research proposal submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in the Department of Mathematics and Computer Science in the College of Arts and Science at Stetson University DeLand, Florida

Spring Term
2016
# TABLE OF CONTENTS

Abstract.................................................................................................................. page 3.

2. Introduction..................................................................................................... page 4.
3. Background..................................................................................................... page 5.
5. iOS Objective C using Xcode................................................................. page 8.
8. Fitbit.............................................................................................................. page 12.
9. Conclusion.................................................................................................. page 16.
10. Future Work............................................................................................... page 17.
Abstract

Smartphones have become an easy and affordable commodity, has opened the door to easily and constantly recording and computing sensor data. Bluetooth enabled health sensors, such as fitbit and Shimmer modules, have allowed the extrapolation and storage of pertinent medical and health data. Using social media along with fitbit would allow for people to compare and be ranked in their performance during exercise and running, with friends who also use the application validated through social media. The application would allow for social integration, and competition, without having to expose their scores to all of their friends or other users who aren’t Facebook friends.
PREFACE

Beginning the semester this project was gearing towards using shimmer modules to read and store ECG data. If any minor variances occur the application would alert a preset contact via email, and if it was more serious like a heart attack it would notify EMT or another predesignated contact. The idea was to make a less invasive pre-emptive device to manage their congenital or hereditary heart risks, rather than getting a pacemaker with similar capabilities. The problem is due to outdated software and hardware which incurred an unknown number of technical errors and defaults, it had to be changed. The next thought was to use a Fitbit Charge HR to gather the data needed for the application, but due to a lack of ultra precise accuracy which was needed, the project changed to shortly after midterms.

INTRODUCTION

Technological discovery has been accelerating immensely, allowing for most people access to smartphones and other portable computers. This has allowed for the advent of personal sensors to allow people to track and manage their exercise and health. This has lead to wearable sensors for exercise to become a large growing market with thirty-three million units shipping this year alone[13].

This project will take a look into the new coupling of these wearable sensors with an iOS mobile device, and a Parse backend. To be exact this project will create an application that is able to take in the data collected and store it where it can be accessed later, extrapolated, and then able compared to results of other users to the user’s own. As well as being able to make a geopoint driven running path system that is able to allow the user to make or use paths friends have made, and have it track your progress and stats.
The point of the running paths is to give a direct method of competition and comparison between the user and friends. So while the user is running the Fitbit will take the heart rate data, as the phone record the time elapsed, and will be self terminating once all of the checkpoints are made. This will allow for a hands free way of keeping track of where you have been and how well the user has been doing during the duration of it.

There are a few applications out there today that do similar functions but not that do both, they mainly do one or the other not both. The couple that do both tend to do one aspect well and the other poorly or neither very well. The closest that I have seen is the Fitbit application but that is to be expected due to the fact that they made the Fitbit itself.

BACKGROUND

Exercise monitors have been around since the medical technology became affordable enough for professional sports to use the equipment to enhance player performance. Heart monitors were used with treadmills and other indoor equipment, it was large bulky had a lot of wires and delicate parts. The energy consumption for these devices was immense also adding . Now roughly fifty years later sensors have become inexpensive and compact enough to be a convenience that any amateur athlete, or exercise hobbyist can own[21].

Wearable Technology has been moving towards smart watches ever since the Pebble in 2012[1]. The Pebble was a crowd funded smartwatch device which raised $10.2 million to make their smartwatch[1]. This was a major tipping point for the wearable sensor market, and displayed the consumers demand for an affordable, low hassle, wearable sensors.

Due to the rise in demand that has made the production of these sensors so cheap, they have also become accessible for use in physical rehabilitation[4]. It allows for physical therapists to monitor patients both more directly as well as when patients are supposed to work
independently. Though a lot of physical therapy uses clothing with built in sensors over in an area able to connect a sensor network, which makes the devices more proprietary and limited in their range of effectiveness[2].

The access to these cheap and simple sensors could also be used to help the elderly, with things like keeping active, detecting injury without any proprietary service (like Life Alert), and more. Wearable sensors are a revolutionary devices that can give people autonomy from things that can be done by these sensors and a smartphone application. The best example is probably that personal trainers are falling to the wayside in favor for personal activity tracking with a wearable sensor. Though personal trainers, aren’t totally done as of yet, but it is a foreseeable future due to both the uptick in demand for wearable sensors, and the decrease in purchase of personal trainer services since this new market started[22].

RELATED WORK

Since the wearable sensors have become smartphone compatible there have been fitness and activity tracking applications. Even Fitbit has had an application for iOS and Android since their pedometer came out in 2008[11]. In recent years more and more sensors are being added to the wearable products, specifically heart rate sensors. The thing is though many them tend to either not offer connectivity to social media and if it does then it tends to not have many other features. There are also plenty jogging trackers, some offer fitness tracking as well as the jogging path, but most of these also don’t offer any social media connection.

Fitbit’s application comes close to what this project is looking to accomplish, in the display of analytics. The Fitbit application connects to the device, communicates with the server, is able to display all the data, and has some form of social media interface[10]. Though from looking at the social media aspect of it it seems to be mostly for posting and viewing of general
activity not a breakdown of friend’s statistics[10]. Fitbit also doesn’t allow a user to make a running path and then be ranked based on performance on each path[9].

Runtastic is a jogging application, where you are able to record and analyse your results from the path that you ran. This application even has a color coordinated running line which shows where you activity was at its peak and where it was to low[17]. Runtastic also tracks the amount of time and distance the run was. The data can also be shown over averages for the whole of the run. Where it lacks is the activity tracking when not in an active run, and no social media whatsoever[17]. This application also seems to have many in application purchases to gain access to the full array of the features that are offered.

The last one to compare is the Digifit iCardio, this application is also an activity tracker for array of possible exercise methods, like running, biking, hiking and so on[6]. Digifit iCardio allows users to track activity, record running paths, track performance on those runs, and has an easy access dashboard[6]. Where this one falls short of what this project looks to accomplish is there is no social media connection, and doesn’t seem to allow you to view activity besides the over all. Where this project would be able to break down by the day, week, or month, Digifit seems to look mainly at individual workouts and an overall averages.

**iOS Objective C using XCODE**

For iOS application development there are a few things, first being a Mac OS computer, and the xCode programing environment. The most recent release of xCode and iOS there were large updates, one of the biggest being the replacement and update of the condensed lib files[20]. Xcode also rolled out the latest versions of swift and objective C, swift being their new language that requires little programing. This project uses objective C due to the level of control as well as previous experience. Objective C also has newly updated and improved MKMapkit,
UITableView, CLLocation, and the new delegates that are needed to implement them with
them[20].

The key feature is making and run jogging paths, using the MKMapkit and the
CLLocation[20]. The user enters the create a new path or run one that was previously created,
either of their own creation or a friends. When entering the new path scene it segways to the
MKMapView in the NewPathViewController, once started the user’s CLLocation is used to start
collecting locations when the distance between the last and current position is one meter or
greater. The array saves the coordinates until the finish button is hit and proceeds to push the
array to parse, but only if there is more than two points in the array. While the application is
creating the running path, it uses the same array to create an overlay to display the path that is
being created until the finish or cancel button is clicked.

The next major use of the MKMapkit is when the user is viewing the paths of friends or
their own. Once a path is selected either of their own or their friend’s, the MKMapView is used
to display the path that was previously made using the same overlay calls from the MKMapkit,
as when displaying creating a new one. Once the overlay is displayed the user then can then
choose to run the path that had been previously created. Running the previous path can only be
done when the user’s current location based on their CLLocation is within a meter of the first
coordinate form the array retrieved from parse. Once started, the array is used with an array of
boolean variables, while running when you come within a one to two meter radius each of the
points in the array the corresponding boolean in the array is switched. The run can only be
completed once all of the checkpoints are passed, otherwise it won’t allow for the run to be
completed and have to cancel it instead.
The application has a need to be able to list things that it queries from the database such as friend, paths, and the fitbit data for each different run. Objective C in recent years has created the UITableView to dynamically display a list of items or objects that when pressed can lead the user to another ViewController. Specifically in the FriendsViewController uses UITableView to display the users who are also friends with the current user on Facebook. The table is populated with the queried data, which when a friend is selected it brings the user to the ProfileView where the friend’s fitbit performance, paths, and other data can be displayed. It is then used again, to display the selected friend’s paths that when selected the user brings up the MKMapView where the path is then run as previously mentioned. The UITableView is again used to display the user’s paths in the MyPathsViewController, and when selected does the same as FriendPathViewController.

Apple has been working to move towards swift in recent years especially after the most recent update Apple rolled out this last fall. While they improved xcode, objective C, and swift, in the months following the release there have been an abundance of tutorials and examples of swift have been created. Whereas with objective C the tutorials for the new features and replaced functions and frameworks are few and far between. This fact has made it incredibly difficult to troubleshoot the issues that arise while programing the application.

During this semester, to manage all the files, SDKs, and the NXOAuth Library, the use of a dependency manager alleviated the stress and work of keeping all files updated. CocoaPods was the one suggested by both Parse and the NXOauth2 library from Github. All that is needed is the creation of a pod file listing the required SDKs and libraries, in the folder of the project you want it to access. Then run through the terminal CocoaPods downloads the files and creates a project workplace to be used instead of the project to work[23].
FACEBOOK API

Facebook has created an extensive SDK and API for communicating with Facebook’s servers. Facebook for mobile development has been an integral key to online and communication between users through the application. Major functions of Facebook for the application need access to user’s friends, and Facebook login. The Facebook login button, protocols then ask the users for access to their Facebook, and only the parts that the application needs access to[8]. The Facebook Login button is a improved feature to the SDK which allows for developers to create the iconic button without having to find their own images. This button is also linked to the permissions that you set as what was needed to access. The user is then prompted to allow the application to access to certain items the permissions outlined. Once access is obtained the application creates a user in the database using the Facebook identification and other pertinent data.

The Facebook API requires the generation of a developers account which then in turn grants access to the SDK needed for access to the servers. Facebook servers have adopted the Oauth 2.0 method of client and user validation[12]. The project designates the server resources that it wishes to gain access to through the setup done by the login function. It will request the client ID and secret ID that are needed for Oauth 2.0 to work, from the authentication server[5]. The application will then ask for the user’s permission for the resources the server wishes to access[5]. The authentication server will then send an access token that will allow the application to access the resources it requested and the actions it requested to use for these. To be exact, if the application wants to be able to look at your posts and only has permission for that it won’t be
able to make posts itself and vice versa.

The next step is allowing the application to gaining access to the Facebook identifications of all of the user’s “friends”. There are a couple options on how this could be handled. First option is to have the application use the SDK to call to the Facebook servers and then call to the backend and check if the identification numbers from the friends list match the IDs in the backend. The other option, is to take the list one time when the user is logged in for the first time or everytime the application is opened and add it as an array of the backend. Finally there is also the chance that once opened the application could get the list once from Facebook and save it as an array in the application, making any queries from the backend faster but could bog down the memory depending on the size of the array. For the the project it may be wise to try a each of
these respectively. Though, in all likeliness the best method will probably be the second or third method because at the least it would cut down on the amount of queries to an high traffic server.

**PARSE BACKEND**

Parse is a free online backend hosting service, it can be use to for iOS, android, web, and desktop applications. The service is free for users with a certain number of requests to the server. The server runs in a MongoDB format which has an open flow input for rows allowing for the creation of columns as they are required. This is done because the transfer of requests is sent and requested in json format[14]. Json is a named array list of attributes and their data, Parse then takes the list of data and breaks it into columns named by the attribute, unless it already exists.

One of the main advantages of parse is how it handles the json data. As mentioned it can create columns based on the new incoming json attributes, this means that if there's a change to the code to add or change a feature, the changes are handled in the application instead of both the application and as well as dropping the tables and recreating them. Though this could also have its pitfalls, for example if there is multiple post requests within the application. If there are any inconsistencies in the attribute names it could cause multiple columns of similar names, get requests would miss any columns that were mistakenly named differently.

Parse SDK is mainly used for pushing data to custom objects or to query data that is needed for display. The user creates a profile that gets pushes the user information using the PFUser signup function, this is also where the user logs into Facebook to connect the user’s Facebook information[14]. The PFUser calls are then used to logout and login to the application. Once created and the data is pushed to parse, the data is cached so that it can be used later without asking for it again unless logged out and attempting to log back in.
The remainder of the push requests are made mainly through the PFObject functions push a json array to populate the custom tables[14]. The tables can be set up prior through the parse website, or once created the tables columns can be created and populated by the push call made in the application. This is used in the NewPathViewController to push the new path and all other subsequent data linked to that. The next place PFObject is called is to push the collected time and fitbit data to Parse when successfully completing a run, the user or friends have made prior, in the RunPathViewController.

The application does need to be able to get data from parse just as often as pushing data to it. As stated the UITableViews display the friends data on the FriendsTableViewController, and the run paths tables that are under each friend as well as the user’s own paths. The FriendsTableViewController first uses the PFQuery call to pull the user’s FacebookID array of their friends from Facebook. The array of FacebookIDs is queried against users’ FacebookID of the users, fetching the names and userIDs of each. The user can then select a user to view, when selected the userID is used to PFQuery their friends’ profile, performance, and paths. The paths are then listed where each running performance can be viewed or the user can proceed to try to run the path that they are looking at. Which the pathID is used to PFQuery the array of CLLocationCoordinatecoordinates that are used in the MKMapView to produce the path overlay.

Parse SDK has an analytics system that with only a couple calls in tracks crashes and all of the traffic to and from the server[14]. The analytics is able to analyze the rate of the incoming data as well as how often or frequent the application calls the servers. This looks to be the best method of tracking the use and speed of the applications communication with the servers[15]. Analytics handles everything from requests, crashes or bugs, performance and even slow queries.
The analytics system is also a possible choice for how to display the data of the user performance in a table or graph form[15].

At the beginning of this semester, Parse had put up a notice that it would be shutting down its servers for good. This luckily isn’t taking place till mid-January of 2017, but it has had certain features beginning to be sidelined or shut down. Specifically the SDKs and other features are no longer going to be upkept, such as the FacebookGraph API. FacebookGraph is Parse’s own Facebook integration calls, which appear to no longer be upkept or in use due to their shutdown of the tutorials linked to it on their website. Though it seems that for now, or at least until the next major update of xcode or objective C.

**FITBIT**

Fitbit as well has its own Fitbit application which is the medium between the Fitbit, the smartphone, and the server. The Fitbit itself is just a collection of sensors with a display for easy display as well as basic watch functions. For the data to be stored it must be paired with the internet enabled device for it to be stored to the server. The server uses oauth 2.0 to validate the communications between the the server and the application as well as any other subsequent applications like this one[16].

The Fitbit has a bit more of a stringent process of validation for use of their data than Facebook does due to the lack of SDK. Fitbit has more strict requirements on what This makes using it in an application more difficult to implement for iOS applications. The NXOAuth2Client library is used for this validation of the fitbit Oauth 2.0 and the functions needed to request the data need. NXOAuth2Client is a very generalized product so it didn’t work directly downloaded using CocoaPods due to the separate set of requirements that fitbit calls for over other services. Luckily the library had been attempted to use this library for the same reasons so the creator
already worked on a workaround to make his library work with fitbit by adding a few lines of code.

When running the NXOAuth2.0Client it needed the work around to get the OAuth token from the fitbit servers. The NXOAuth2.0Client especially was difficult with the use of the workaround, because when it failed, it was uncertain if it was due to the workaround or the implementation. Though it is only is used to pull the data from the fitbit servers to be pushed to the Parse server.

Fitbit Charge HR, has a sleek and robust system with pedometers, accelerometers, heart rate, distance and elevation sensors, while remaining economical to a consumer[11]. Fitbit servers also manage the data taken in by their products, it is because of this that their application is needed to collect their data[9]. The sensors connect to the servers via bluetooth to the application then to the servers[10]. The diagram, Image 1, below shows how the sensors take in and extrapolate the data, and how the Fitbit processes the data and sends it to the application.
Specifically the optical light heart rate sensors, as shown in the image[2, 3], use photoplethysmography. The process uses infrared light shined through the skin the sensor then measures the dilation of capillaries with an optical sensor[7]. Photoplethysmography doesn’t track heart rate through the electromagnetic impulses of the heart, which means that it can be applied directly to skin anywhere on the body[7]. The trade off is that the algorithm that calculates the results for Photoplethysmography has to use assumed calibration and averages of
current or previous readings. Optical noise, or interference from other light sources or lack of contact with the skin, creates skewed, altered, or false results.
Figure 4A

Figure 4B

Protrusion Detail

Figure 4C
As the diagrams show in the diagram above these sensors use the reflected light off of veins or arteries to calculate the heart rate of the user’s fitbit[13]. The photodetector takes in this reflected light but as mentioned there are issues with this that is why for improved accuracy the device should be placed higher up the forearm and tight against the skin[11]. Accuracy of the devices will be key to the accuracy of the data, which will dictate the accuracy of the results.

**CONCLUSION.**

The project challenging, bringing together the Parse SDK, Facebook SDK and NXOAuth2.0Client, all incorporated to make this application. The key to the whole application became the integration of all the different frameworks, libraries, and SDKs. Bringing them together and setting them up was made easier through the use of CocoaPods, where any updates and changes to the SDKs is updated when the project is opened up without any hassle. Apple’s push to make swift their main new application development language made troubleshooting more difficult when working on Objective C.

This project took the relearning of certain Objective C features to get the application to work, features that were improved during Apple’s most recent major update. Features such as the mentioned MKMapView, UITableView, and MKMapkit’s overlay, and more specifically the overlay to date is running through deprecated functions. The delegates that were needed to run the MKMapView and the UITableView, had been updated as well but made them easier to use. As well since I targeted my application to run on iOS 9 or later made it so the project didn’t require redundancies in the code to run on the old OS.

The project still needs work before I would consider it good enough for consumers, but it has come leaps and bounds since the beginning. Though it still needs work and is rather plain, the focus was getting the communication functioning over the aesthetics. Future work would be
needed to streamline this product for user consumption. The focus of getting the integration of all of the different libraries and SDKs as the key focus of the project, encryption has been a secondary thought for all the communication between Parse, Facebook, and Fitbit.
REFERENCES


**IMAGE REFERENCES**

