Privacy-Aware Remote Mobile Health and Fitness Monitoring

Yu Xiao
Caitlin Gruis
Kaidi Du

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-100
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-100.html

May 12, 2017

Acknowledgement

Acknowledgement

Professor Ruzena Bajcsy, University of California, Berkeley

Professor Ali Javey, University of California, Berkeley

Daniel Aranki, University of California, Berkeley

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2017

Department: EECS

Concentration: Robotics and Embedded Software

TITLE OF PAPER: Privacy-Aware Remote Mobile Health and Fitness Monitoring

STUDENT FULL NAME: Yu Xiao

This **Masters Project Paper** fulfills the Master of Engineering degree requirement.

Approved by:

1. Capstone Project Advisor:

Signature: Ruzena Bajcsy Date 5/1/2017

Print Name/Department: RUZENA BAJCSY/EECS

2. Faculty Committee Member #2:

Signature: Ali Javey Date 5/1/2017

Print Name/Department: ALI JAVEY/EECS

BERKELEY
TELE-MONITORING

Privacy-Aware Remote Mobile Health and Fitness Monitoring: Extending The Functionality of The Berkeley Telemonitoring Framework

Master of Engineering Capstone Final Report

Yu Xiao
with Kaidi Du, Caitlin Gruis

Acknowledgement

Professor Ruzena Bajcsy, University of California, Berkeley

Professor Ali Javey, University of California, Berkeley

Daniel Aranki, University of California, Berkeley

Contents

Introduction

1 Motivation and Introduction for the Berkeley Telemonitoring Project

Health problem is always one of the most primitive problems people concern, but it is very difficult to take care of people's health all the time. One of the main difficulties of dealing with the health problem is that it needs problem solvers to be the expertise in the health area, such as doctors. The very first motivation of Berkeley Telemonitoring Project came from this point. When the doctors want to monitor the recovery status of patients through the smartphone and issue the medical advice remotely, because the lack of time and the software engineering expertise, doctors cannot build a robust and accurate application by themselves. However, if the doctors hire some software engineers to help them, the engineers have limitations about the data type that is important to the doctors because the lack of the advanced knowledge of medical and it is not easy to build a close collaboration between doctors and engineers (Aranki et al., 2016).

Besides the dilemma between the doctors and engineers, another motivation for the Berkeley Telemonitoring project is the high cost of the health care. According to the ObamaCare Fact report, one of the main ideas that Patient Protection and Affordable Care Act (PPACA) can reduce the cost is to reduce the readmission rate in chronic health condition (ObamaCare Fact, 2016). The high readmission rate means people have to go back to the hospitals to consult with their doctors more times for the same problem, which will increase the total cost of health care. If such unnecessary readmission can be significantly reduced, which means people will spend less money to visit doctors because they do not need to visit that often, people will afford better health care plans[1].

Berkeley Telemonitoring framework is an easy Android based development framework for the people without software engineering background, such as doctors and personal trainers, to build their own

[1] How Does ObamaCare Control Cost.
http://obamacarefacts.com/obamacare-control-costs/

application based on their different demands. By using the Berkeley Telemonitoring framework, doctors can collect data from patients and deliver the data to the cloud, and then the data will be automatically analyzed within the cloud and will be viewable by the doctors who can give medical feedback to patients in the real time. Figure 1 shows the basic work flow of Berkeley Telemonitoring framework.

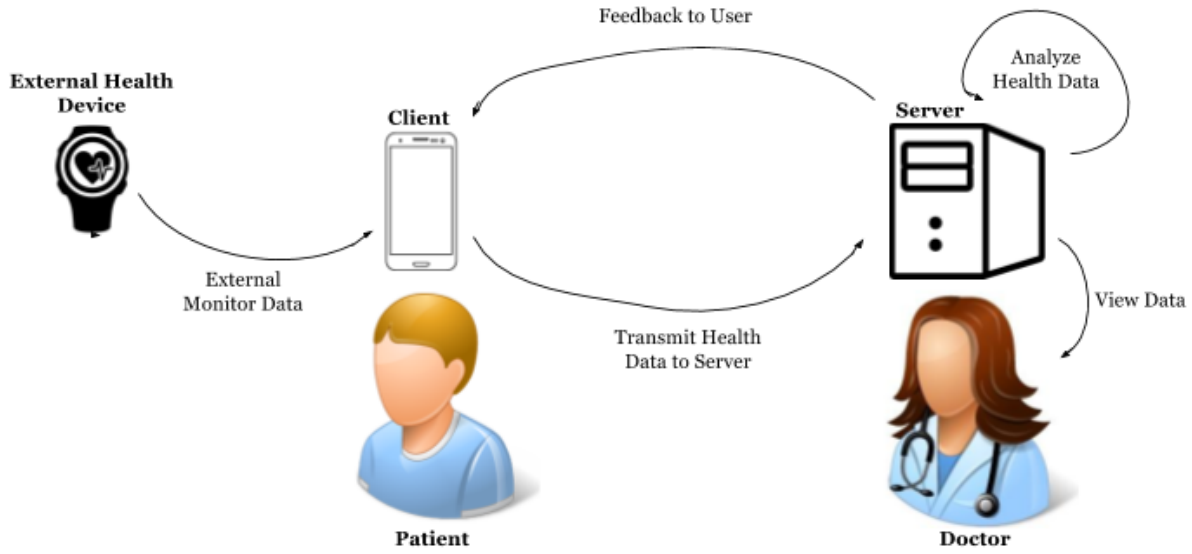

Figure 1. The Berkeley Telemonitoring Work Flow

The Berkeley Telemonitoring framework provides several important features. The most important feature is the simple APIs to develop the telemonitoring process, which is easily enough to be used by anyone. With the APIs, people can collect data on the client side, can build server especially for the telemonitoring purpose, and can build their own applications based on the examples framework provided. The framework also provides validated algorithm proved by the biomedical literature to analyze the patients' data on cloud. Finally, the framework provides the privacy and security measure to prevent patients' data being stolen by other people (Aranki et al., 2016).

The Berkeley Telemonitoring project began in 2013 and previous MEng students has finished some features with Ph.D. advisor Daniel Aranki right now. To provide an application that can real-time monitor and coach the runners and to demonstrate the Berkeley Telemonitoring Project, a sample application,

Running Coach, was built last year and this year (Aranki, et al., 2017). All the work for our group started from this point and the following section will talk about the specific job of each team member.

2 Overview of Current Team

There are eight people currently working on Berkeley Telemonitoring project: professor Ruzena Bajcsy and Ph.D. advisor Daniel Aranki; three MEng students, Caitlin Gruis, Kaidi (Kate) Du, and Yu (Sean) Xiao; and two undergraduate students Gao Xian Peh and Sachin Kesiraju. Since before joining the Berkeley Telemonitoring group, Caitlin, Kate and I were not familiar with Java programming language and Android Studio IDE, our Ph.D advisor Daniel help us learn Java and Git version control tool through the whole fall 2016 semester. Besides learning Java, we were also required to read and understand the codes developed by previous people, which is very important for us to build further features to the framework.

For the spring 2017 semester, there are mainly four parts of jobs need to be done by us. The first one is that we need to test the functionality of the Running Coach application and find possible bugs of our APIs. All the people in group are responsible for this job and we recruited more people to help us test the Running Coach application. Before starting the testing, Caitlin, Kate, Gao and I first created 12 tables inside of our MySQL database to store the 12 different data collected from the Android mobile phones.

The second job is to expand the Bluetooth Low-Energy (BLE) capabilities of our framework for which Caitlin is responsible for. The third job is to sanitize of health-sensitive data to make it private and secure when being transmitted between the client and server, for which Kate is responsible for. The final job for this semester is to expand the functionality of the server development library, which I am responsible for. To learn more about the functionalities of Bluetooth Low-Energy (BLE) capabilities and data sanitization, please refer to Caitlin's paper and Kate's paper. Figure 2 shows the division of work.

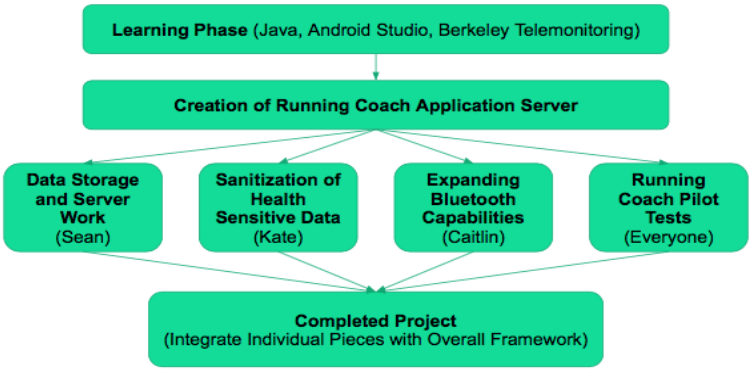

Figure 2. Division of Work

Chapter 1

Technical Contribution

1.1 Server Library Development

1.1.1 Important Data Structures

To develop the framework in more proper way, there are several self-defined data structures used by the Server Library. There are twelve built-in data categories that can used directly by the developers, such Cadence, Heart Rate and Blood Pressure. Each of these data categories can have multiple data value fields, for example, the Cadence category will measure the distance, the time elapsed and the average cadence. So, we defined a data structure called *Container* to represent each category of data. There are twelve corresponding data containers classes, such *Cadence Container*, *HeartRate Cotainer* and *BloodPressure Container*. Each container contains the multiple data value fields corresponding to the specific data category. All the data value inside of one *Container* is collected simultaneously and a Java *TimeStamp* object will be created at the same time, which indicates the time at which this *Container* is created. Combining the *Container* object and the *TimeStamp* object together, we defined another data structure called *TimeZonedTimeStamp*.

1.1.2 Introduction of Server Library

The main purpose of the Server Library is to help developers easily build a server especially for the telemonitoring purpose, which will collect data from client side and then distributed the data to different tables of the database. There are three packages of this library right now:

1. *analysistools*
2. *clientservice*

3. *database*

The *analysistools* package is mainly used to predict the potential results using the user's' data as input and using linear regressor to do the prediction.

The *clientservice* package contains the main functionality of the Server Library and contains the following units:

1. *Exceptions*
2. *Jobs*
3. *TI Protocol*
4. *Client Handler*

The *Exception* unit concludes the exceptions the program will throw if anything goes wrong. The *Jobs* unit contains the job classes that the server is capable of handling. The *TI Protocol* unit is the communication unit that makes sure the server and client side communicate properly. The *Client Handler* is used to set up the communication between client and server. The *analysistools* package and *clientservices* package have been built by the previous students and I am responsible for the *database* package.

The *database* package is one of the main packages that the server library has, which is the parallel package to the *analysistools* package and *clientservices* package. The main purpose of the database unit is to provide some classes corresponding to the data type containers for the users to make query and change to the records in the database without building everything from scratch, such as make connection between server and database and write SQL language.

There are two main categories of classes. The first category is used to build the connection between the server and the database and the second category is used to make query and change to the records of a specific table.

1.1.3 Java Database Connectivity Programming Interface (JDBC)

Java Database Connectivity Programming Interface (JDBC) is the one of the most important APIs used in the *database* package, so understanding how JDBC works helps understand how the *database* package works. Figure 3 shows the basic architecture of JDBC[2].

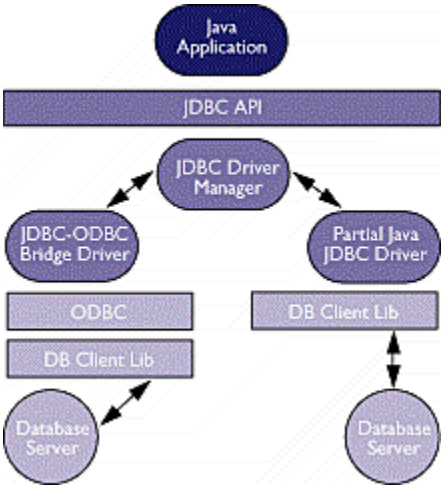

Figure 3. JDBC Architecture[2]

The JDBC API is the industry standard developed by Oracle for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API provides a call-level API for SQL-based database access (Oracle). There are mainly three functionalities of JDBC API:

1. Set up a connection with the database
2. Send query statements
3. Process the results

JDBC is used to construct the database connector class named *MySQLConnector* to connect the server with selected database and make SQL query to the the specific tables or record.

1.1.4 Connection to The Database

[2] Oracle. JDBC Overview. http://www.oracle.com/technetwork/java/overview-141217.html

The first category of classes is to establish the connection between the server and the database. There are two classes served for this purpose: *MySQLConnector* and *DatabaseInfo*.

Before establish a connection, the user should first use the DatabaseInfo class to create an object, which contains every information needed to establish the connection and is an immutable class. By separating the information container object from the connector object, the user only need to modify the information container object and will not mess up the connector object. To make a connection successfully, the user should know four basic information about the database based on the JDBC standard: JDBC URL of the database system, username of the database, password of the database and the database name of which user want to select. The JDBC URL is the address which points to the selected database system, which could be locally or on the cloud. The standard formats of different database system are different. Table 1 shows the standards for four different databases[3].

RDBMS	JDBC driver name	URL format
MySQL	com.mysql.jdbc.Driver	**jdbc:mysql://**hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:**@hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

Table 1. URL Standard
(All the highlighted part in URL format is static)

There are could be multiple databases inside of one database system, so the users can select the specific database name they want to use. Our design forced the users of our API to type in the username

[3] Tutorial Point. JDBC-Database Connections. https://www.tutorialspoint.com/jdbc/jdbc-db-connections.htm

and password of the database because all the data collected from the clients will be some very sensitive data and cannot be exposed to other people. Considering such security and privacy problem, we request the users to set the database to be password accessed only, otherwise the program will throw an exception to the programmer.

After the database information is created, users can use the *MySQLConnector* to get a connector singleton instance. The connector object is used to connect the server to one specific database. Once the *MySQLConnector* object is initialized, it will not be allowed to create second time. The first reason to make it singleton is that we do not want to developers mess up multiple connectors while debugging. The second reason is that such method can provide a persistent connection to the server but it is not the only way to make a persistent connection.

After the connector instance is created, user can also use this instance to check whether a table already existed and get a statement to make query. To prevent the potential query injection vunerbility, the statement returned from the connector instance is the *preparedStatement,* which is a statement that only accepts certain values as input. For example, some attackers can use the combination of the data and query statements to dump the information inside of the database, which should be only accessed by certain authorized people. The *prepareStatement* method will prevent this problem by only executing certain query types with certain types of data input, which will filter out unnecessary information or throw exception to the programmer.

1.1.5 Database Table Modifier

The database table modifier category is built to further help developers to build the server in a more convenient way. Each modifier is corresponding to data container and one container is corresponding to one table in the database. With the database table modifiers, developers can make multiple changes to the tables directly by using the methods of the modifiers, so they do not need to build everything from scratch. In the database table modifier category, the factory design pattern is used. There is a singleton class named

getDatabseModifier, which will return an instance for the users to have a specific modifier. There is an interface named *DatabaseTableModifier*, which will be inherited by all other data container table modifier. There are five methods in the *DatabaseTableModifier* interface:

1. *createTable()*: this method will create a table corresponding to the data container given by user. For different data container, the implementation will also be different.
2. *insertRecord()*: this method will insert a specific record to the corresponding table.
3. updateRecord: this method will update a record or a group from records of the corresponding table.
4. *deleteRecord()*: this method will delete a record or a group of records from the corresponding table.
5. *selectRecord()*: this method will return a record or a group of records from the corresponding table.

For different data containers, the implementation of the modifiers will be slightly different, but they will all implement the *DatabaseTableModifier* interface. Such generic design pattern will be beneficial to create new modifiers when new data containers added to the core library. Also by using the factory design pattern, the users only need to think about how to use the *getDatabseModifier* and do not need to worry about what the specific implementation of each modifier. In such way, the signature of using the modifiers will not change even when new modifiers are added, so our API will keep consistency when it is updated in future.

1.1.6 Future Work

There are totally twelve containers in the core library, which means there should be at least twelve table modifiers in the *database* package to help users reduce the work to create and modify the tables in the database. Right now, only the *BatteryUsageTableModifier* is built and other modifiers should be built in the similar way.

Also, right now, only the unit tests are implemented, but further tests are needed to test the functionality and the reliability of each modifier.

1.2 Test of Running Coach

The Running Coach application is built by implementing the Berkeley Telemonitoring framework and the purpose of this application is to demonstrate the functionalities of the framework and this application also aims to reduce injury and increase performance of runners. Figure 5 shows the screenshot of the Running Coach application.

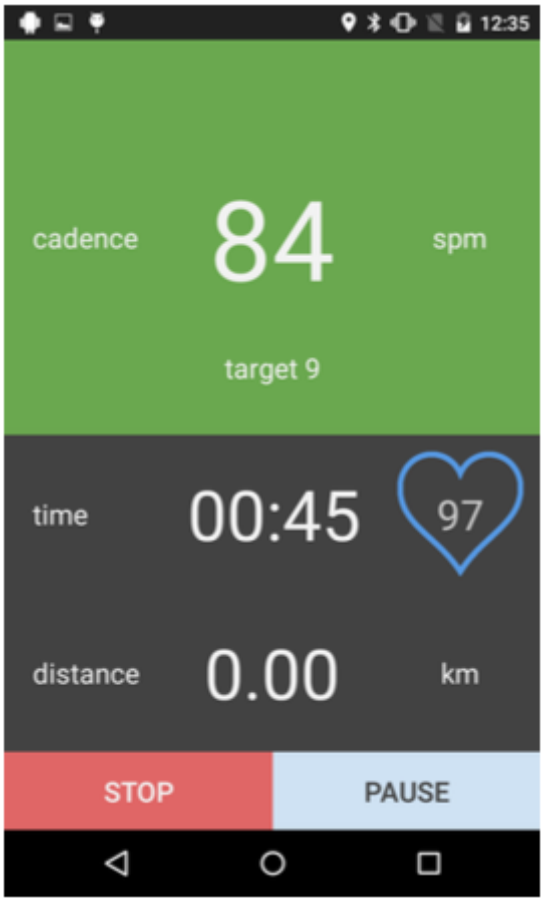

Figure 5. Screenshot of Running Coach Application

Continuing from where the previous MEng students left, we first create a MySQL database to store the data collected from the Android phones. There are 12 different types of data needed to be collected, so we created 12 tables inside of the database (Figure 6):

1. User Parameters - Records information about the user's age, height, weight, gender, leg length, beginning and target running cadence, start and end date of their training program, and the steepness of the training regimen (alpha) (Caitlin Gruis, 2017).
2. Run - Records information about the user's run, including current and target running cadence, average and target speed, average heart rate, and run start and end time (Caitlin Gruis, 2017).
3. Distance - Records the start and end time of the user's run as well as the distance during certain period (Caitlin Gruis, 2017).
4. GPS - Records the latitude and longitude of the user while running (Caitlin Gruis, 2017).
5. Cadence - Records the cadence of a user while running (Caitlin Gruis, 2017).
6. Speed - Records the speed of a user while running (Caitlin Gruis, 2017).
7. Heart Rate - Records the heart rate of a user before and after the run, as well as whether the heart rate was taken by an external heart rate monitor or through the phone application (Caitlin Gruis, 2017).
8. Battery - Records the battery percentage of the user's phone and whether or not the phone is currently charging (Caitlin Gruis, 2017).
9. Screenlight- Records whether the user's screen is turned on or off, and can be used to tell if the user is looking at their screen or not while running (Caitlin Gruis, 2017).
10. Energy Expenditure - Records how much energy the app is currently using (Caitlin Gruis, 2017).
11. Survey - Records information from a post-run survey that all users are colud take. The survey asks questions about how accurate they thought the application was and how tired they were after the run (Caitlin Gruis, 2017).
12. Report - Records generic information about the application such as when someone opens or closes the app and when they start or stop a run (Caitlin Gruis, 2017).

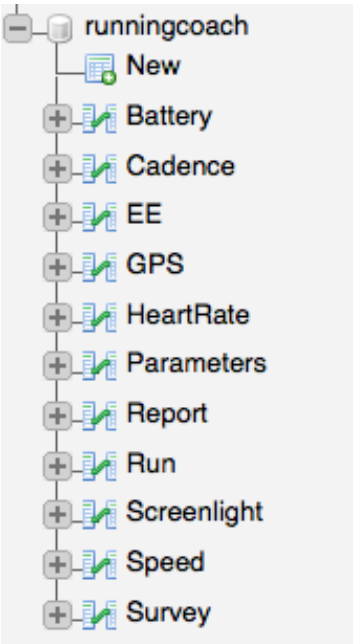

Figure 6. Overall Tables in Database

DATAID ▾ 1	SID	dataPointTimeStamp	submissionTimeStamp	heartRate	source
815	yuxiaoTest	2017-02-14 15:59:31	2017-02-14 15:59:47	83.056640625	FACE
814	kate Test	2017-02-14 15:55:00	2017-02-14 15:55:18	67.4560546875	FINGER
813	kate Test	2017-02-14 15:54:58	2017-02-14 15:55:18	67.67578125	FINGER
812	kate Test	2017-02-14 15:54:58	2017-02-14 15:55:06	67.67578125	FINGER
811	kate Test	2017-02-14 15:54:56	2017-02-14 15:55:06	67.4560546875	FINGER
810	kate Test	2017-02-14 15:54:54	2017-02-14 15:55:06	67.67578125	FINGER
809	kate Test	2017-02-14 15:54:52	2017-02-14 15:55:05	67.4560546875	FINGER
808	kate Test	2017-02-14 15:54:50	2017-02-14 15:55:00	67.67578125	FINGER

Figure 7. Sample HeartRate Table

After creating these tables, I started to use a LG Nexus Android phone [4]with Android version 4.4 (KitKat[5]) and a Huawei P9 Android phone with Android version 6.0 (Marshmallow[6]) to conduct the first period testing. The testing process included:

1. User information accuracy testing - Compare the user information on the phone with the data of user parameter table inside of the database.
2. Application functionalities testing - Try to use the built-in functionalities of the application in different orders and different manners to see whether the application crashes.
3. Data Synchronization testing - Try to use phone to collect data with internet connection and without internet connection and synchronize the data with database to see whether there are data missings or abnormal data points.
4. Different Android version testing - Do the same testing on different Android versions to see whether an application crush happens.

After repeating this testing process about two weeks by Caitlin, Kate and me, we found there was crash on Android version 6.0 and this bug was fixed by our Ph.D advisor Daniel. Right now, we recruit six more subjects to the longer period testing in order to find more potential bugs of our framework.

[4] Android. Android Phones. https://www.android.com/phones/
[5] Android. Android KitKat 4.4. https://www.android.com/versions/kit-kat-4-4/
[6] Android. Android Marshmallow 6.0. https://www.android.com/versions/marshmallow-6-0/

Chapter 2

Engineering Leadership

2.1 Introduction

We now turn from technical contributions to a new topic of engineering leadership, with the goal of viewing our project from a business perspective. For this portion of the report, we consider bringing the Running Coach application to market. With this application comes several questions that need to be answered regarding where our product fits into industry, how we would market it, and social implications.

2.2 Industry Analysis

Today's world is one that is increasingly health conscious. Over 50% of Americans exercise on a regular basis, and this number is on the rise (Mintel, 2014). In addition to this, over 3.3 million fitness bands and exercise trackers were sold between March 2013 and April 2014 (Kovar, 2016). Based on these statistics it is clear that a change in global health patterns is on the horizon, and Berkeley Tele-monitoring believes that it can help by entering into the sports coaching industry.

The sports coaching industry is defined as "consisting of establishments that offer instruction for athletic activities to groups or individuals" (Masterson, 2016, p. 5). This industry tends to target selling towards adolescents and young adults from ages 20-29. Recently, it has been doing quite well, with an average annual revenue increase of over 3.0%. This is believed to be due to an increase in sports participation. (Masterson, 2016).

There are many competitors in the sports coaching industry, and we will take a look at a few of them. The biggest subset consists of human personal trainers and fitness coaches at gyms and athletic

facilities. However, a fast-growing portion of this industry is becoming sports coaching software technology, and this is where our product has the most competition. These platforms are known to give teams and individuals that adopt them a competitive advantage in the form of athletic improvement. Major companies in this industry include Coach.me, AthleticLogic, and Coach's Eye (Tiwari, 2015). Upon reviewing these products, they appear to gather a lot of data, whether through filming an athlete doing a sport or collecting sensor data. Many of them involve recording a training session and allowing an athlete to go back and view their workout later. Where these products fall short is that they fail to provide real-time feedback to the athletes, and they use limited sensors to gather their information. Our product seeks to improve upon these downfalls by providing real-time suggestions to athletes during exercising and using a wide variety of sensors such as accelerometers, GPS, and cameras.

Overall, our final product would fit well into the sports coaching industry. Within this industry, there are several major competitors, some of which are well established corporations. However, our team believes that we have a unique value proposition in creating a framework instead of focusing on a physical wearable device, as well as providing real-time feedback to our users for how to improve their workouts. These advantages described above play a large role into how we will choose to market our product.

2.3 Marketing Strategy

Our primary clients are long-distance marathon runners. In recent years, marathon events have continuously attracted a lot of professional participants and public attention. According to the Running USA Annual Marathon Report, there were 1,100 U.S. Marathon events and more than 500,000 finishers each year from 2014 to 2015 ("Running USA," 2015). This means that there is a large number of clients in the long-distance running market. A Marathon runner's performance heavily relies on their training, and therefore, a comprehensive training plan plays an important role in his/her good competition outcome. However, current personal one-on-one training programs are expensive and without many flexible schedules. For example, an in-person coach training program in COACHUP is about $100/session

("Running Coaches near San Francisco, CA," 2016). Although these human coaching programs can customize the training plans, they are not able to report real-time feedback regarding performance metrics if the runner does outdoor training. However, our smartphone coaching application can quickly generate a personalized training plan for a runner for a low price. This gives our product a competitive advantage in the market.

As mentioned in the industry analysis, there are many sports coaching competitors in this market. However, we would market our tele-monitoring Android platform framework by showing how it is distinguished from the rest. The IBISWorld Industry Report 61162 showed that there were about 123,094 businesses in the sports coaching field (Masterson, 2016, p. 4). Many of these businesses are one-on-one sports training camps and schools that offer instruction in athletic activities to groups or individuals (Masterson, 2016, p. 2). Additionally, the current health and fitness apps are designed for general sports or fitness purpose instead of specializing in long-distance running training, so the measurement of performance may not be as accurate. For example, the Garmin FR620 was an advanced running GPS watch (Rainmaker, 2013). But, it could not support getting any dynamic running information (Rainmaker, 2013). Our framework design allows implementing user interfaces, data acquisition, data storage, and proper security and privacy mechanisms using APIs (application programming interfaces) through the sensors in the Android based smartphones (Aranki et al., 2016). To be specific, our design will customize the training plan by monitoring the user's heart rate based on the cadence (steps per minute) using the smartphone. This method is specifically designed for long-distance runner training and is easily implemented without compromising human coaches' schedules. It would bring more convenience to the users than traditional human coaching.

In summary, our team's marketing strategy involves analyzing the current holes in the sports coaching industry and identifying how our product can fill those needs. In considering these needs, however, we must also consider their overall social implications.

2.4 Social Context

One of the biggest social impacts we see our project having is in the world of health care. While many countries have regulations for the health care of their people, it still remains a growing concern. Particularly in the United States, the increasing cost of health care has lasted for many years. As indicated in the report of the 2016 Milliman Medical Index, "The rate of increase is still well above growth in the consumer price index (CPI) for medical services, and far surpasses the average 2% annual increase in median household income between 2004 and 2014." (Girod et. al, 2016, p. 1). One of the main ideas given to reduce this cost is to reduce the readmission rate for chronic health conditions (ObamaCare Fact, 2016). The high readmission rate means people have to go back to medical facilities to consult with their doctors several times, increasing the cost of their health care.

Since our project provides the possibility that doctors and patients can communicate remotely without going to medical facilities multiple times, it could significantly decrease the readmission rate, which may reduce health care costs. Our project could potentially change the health care structure over time. Instead of going to medical facilities multiple times, doctors could receive health data from their patients via our Android framework, analyze it, and give feedback if necessary. This would save patients unnecessary trips to the doctor's office.

Outside of healthcare, our Android platform based project also follows a technological trend. According to a research paper, the Android platform market share of the smartphone was 45.4% worldwide in 2015, while secondary platform IOS only had 15.3% of the market share - three times less than Android (Burguera, 2011). With such a huge market share, our Android platform based project could have many more potential customers than any other platform. Also, the market share by Android could increase in the future with the new release of Google Android features later this year (Anirudh, 2016). Furthermore, the trend of increasing use of smartphones has continued for many years. The sales growth rate of the smartphone in China for 2015 is 52% more as compared to 2014, and this rapid growth is projected to

continue in following years (Baoling, 2014). This means that not only will the market share of Android increase, but the total number of smartphone users will go up as well.

By following the current social trends, we can conclude that there is potential for wide use of our product. This will help more patients have a better healthcare experience with a lower cost. By taking advantage of Android's large market share of smartphones, our project could potentially change the traditional health care methods for the many people in the future.

2.5 Engineering Leadership Conclusion

In brief, our Extended Platform for Android Tele-monitoring product should perform well in the sports coaching industry due to its unique value proposition of providing real-time feedback through an Android framework. Our major buyers are long-distance runners, and marketing patterns show this group increasing every year. As a good competition outcome highly relates to a good training plan, long-distance runners will be willing to spend less money buying this smartphone based application to achieve the same high-level competition results with coaching by human trainers. Finally, in a social context, we believe our product can take advantage of the large Android market share and potentially create a positive impact on healthcare worldwide.

Reference

Baoling Li, Haiyan Fu (2014, June). Research on the developing trend and strategies for mobile marketing. Retrieved April 14, 2017, from http://ieeexplore.ieee.org/document/6874110/

Burguera et al (2011, Oct 17) . Crowdaroid: Behavior-Based Malware Detection System for Android. Retrieved April 14, 2017, from http://delivery.acm.org/10.1145/2050000/2046619/p15-burguera.pdf?ip=136.152.208.249&id=2046619&acc=ACTIVE%20SERVICE&key=CA367851C7E3CE772E3158474DDFAA3F102E4D4702B0C3E38B352E4D4702B0C3E38B35&CFID=680488789&CFTOKEN=48860795&__acm__=1476255122_c589c038cdedbe7fc0d9761e1015a684

Caitlin Gruis. Technical Contribution Paper (2017). Retrieved April 14, 2017.

Dhebar, Anirudh (2016, Aug). Bringing new high-technology products to market: Six perils awaiting marketers. Retrieved April 14, 2017, from http://www.sciencedirect.com/science/article/pii/S0007681316300830

Daniel Aranki, Gregorij Kurillo, Adarsh Mani, Phillip Azar , Jochem van Gaalen, Quan Peng , Priyanka Nigam , Maya P. Reddy, Sneha Sankavaram , Qiyin Wu and Ruzena Bajcsy. A Telemonitoring Framework for Android Devices (2016). Retrieved April 14, 2017, from http://ieeexplore.ieee.org/document/7545843/

Daniel Aranki, Uma Balakrishnan, Hannah Sarver, Lucas Serven, Eugene Song, Carlos Asuncion, Kaidi Du, Caitlin Gruis, Gao Xian Peh, Yu Xiao and Ruzena Bajcsy. RunningCoach – Cadence Training System for Long-Distance Runners (2017). Retrieved April 22, 2017.

F. (n.d.). ObamaCare Facts: Facts on the Affordable Care Act. Retrieved April 14, 2017, from http://obamacarefacts.com/obamacare-facts/

Girod et. al(2106, May 24). 2016 Milliman Medical Index. Retrieved April 14, 2017, from http://www.milliman.com/uploadedFiles/insight/Periodicals/mmi/2016-milliman-medical-index.pdf

How Does ObamaCare Control Cost. Retrieved April 14, 2017, from http://obamacarefacts.com/obamacare-control-costs/

Masterson, R. (2016, June). IBISworld Industry Report 61162: Sports Coaching in the US. *IBISworld*. Retrieved April 14, 2017, from http://clients1.ibisworld.com/reports/us/industry/default.aspx?entid=1542

Mintel. (2014, October). Exercise Trends Market Report.Retrieved April 14, 2017, from Mintel Academic database.

Rainmaker, D. (2013, November 4). Garmin Forerunner 620 In-Depth Review. *DC RAINMAKER*. Retrieved April 14, 2017, from http://www.dcrainmaker.com/2013/11/garmin-forerunner-review.html

Running USA. (2015). 2015 Running USA Annual Marathon Report. Retrieved April 14, 2017, from

http://www.runningusa.org/marathon-report-2016?returnTo=main

Tiwari, R. (2015). Sports Coaching Platform Technology Market. Retrieved April 14, 2017, from http://www.prnewswire.com/news-releases/sports-coaching-platform-technology-market-worth-864m-by-2021-561558701.html

Wearable Technology - US - December 2015. (2015, December). *Mintel*. Retrieved April 14, 2017, from http://academic.mintel.com/display/757616/?highlight