INTEGRATION OF HEALTH DEVICES WITH HEALTHEREUM

PROJECT MEMBERS

Hitesh Kumar S (1MV15CS038)

Manohar R(1MV15CS058)

Dept. of CSE, Sir M. Visvesvaraya Institute of Technology, Bengaluru.

PROJECT MENTOR

Mr. Kiran

ACKNOWLEDGEMENT

We give our high, respectful gratitude to our Mentor **Mr. Kiran**, who has been our source of inspiration. He has been especially enthusiastic in giving his opinions and critical reviews. We have learnt a lot throughout this internship with many challenges yet valuable experience to complete this task. We will remember his contribution forever.

We sincerely thank **Mr. BadiyuZama Mohammed and Mr. Harish Daga**, who have been the constant driving force behind the completion of project. We also thank **Mr. Prasad Kothapalli** for his constant help and support throughout.

Our thanks and appreciations also go to our friends who have willingly helped us with their abilities.

-Hitesh Kumar S

-Manohar R

TABLE OF CONTENTS

1. ACTION PLAN

Goal #1:	Action Step Descriptions	Start Date	End Date
To send data to health devices from sensors . Resources: Sensors , Health devices.	Transferring data from human body to health devices(Beddit,Runtastic,Daily Burn, etc) using sensors.	09-Jul-18 Monday	11-Jul-18 Wednesday
Goal #2:	**Action Step Descriptions**	**Start Date**	**End Date**
To transfer data from health devices to Healthereum application via Google Fit. Resources: Health Devices, Google Fit, Healthereum Application.	Transferring data from health devices to Google Fit using different APIs and then retrieving the data from Google Fit to Healthereum App by linking the two.	12-Jul-18 Thursday	15-Jul-18 Sunday
Goal #3:	**Action Step Descriptions**	**Start Date**	**End Date**
Architecture and Programming	High level architecture of the data transfer between the devices and healthereum application.	16-Jul-18 Monday	22-Jul-18 Sunday
Resouces: Android Studio and Java backed API	Understanding a simple java program to connect the devices to Google Fit and retriving the health data		
Goal #4:	**Action Step Descriptions**	**Start Date**	**End Date**
Limitations of Google Fit APIs Resouces: History API(.enableServerQueries()), Custom Data Fields.	By adding .enableServerQueries() method, the server data can be combined with local data to overcome the limitations of history API. Backups and deletion of data are the responsibilies of the connected app and not of google.	23-Jul-18 Monday	25-Jul-18 Wednesday

2. INTRODUCTION

The main aim of HEALTHEREUM is to provide a single point of reference for all your aggregated health and fitness data. Sure, a good amount of the info is captured via the motion co-processor nestled inside your smart phone and the sensors in your health devices.

This project gives a brief knowledge about how the health related data can be transferred from human body to health apps like healthereum using the intermediary components like sensors, Google Fit API etc.

2.1 Purpose: Purpose of this document is to provide a set of pointers and guidelines on why and how to implement the retrieval of data from health devices to healthereum application.

2.2 Overview: This document can be used to understand goals, challenges, working of sensors and google fit with their design and srchitecture and steps to connect the applications to google fit to retrieve the data.

3. HEALTH DEVICES AND APPLICATIONS

It collects data from third-party devices and apps, and the more you plug into it the more information you receive. The more information you receive, the better idea you have of your overall health.

Here are some of the health devices and apps which keeps track of the heart rate, blood pressure, distance travelled, quality sleep etc., through which the healthereum application can retrieve the data and serve the cause of the users.

3.1 Wahoo X Heart Rate Monitor

A chest strap monitor is one of the best ways to measure your heart rate, even if it's pricey, and this premium model from Wahoo has the benefit of syncing seamlessly with health applications. 16 hours' worth of data can be stored between syncs too!

3.2 Amazfit Bip

The smartwatch with 45 days of battery life not only looks like an Apple Watch, but its data can be fed right into the health applications via the Mi Fit app.

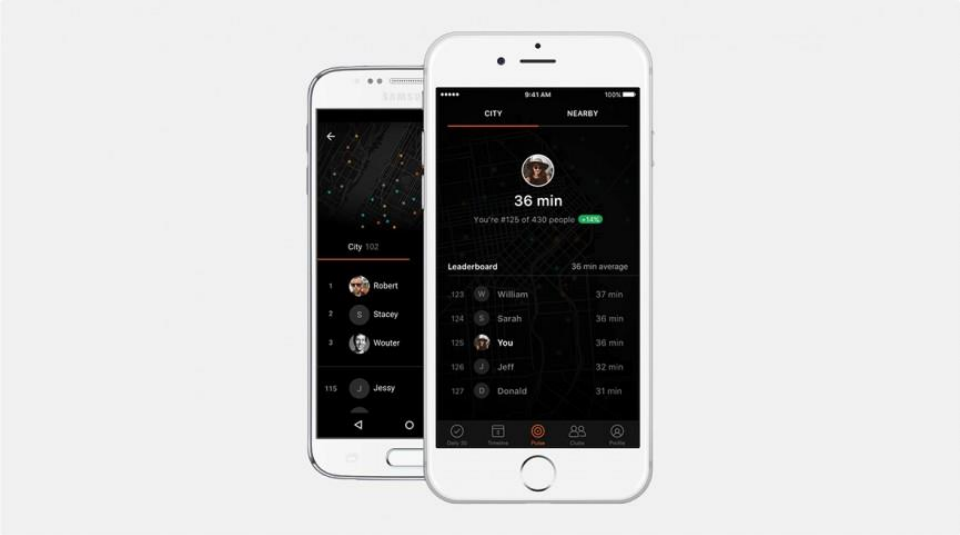
3.3 Beddit 3 Sleep Monitor

The Beddit 3 Sleep Monitor is a favorite at Wearable, and it's easily one of the best sleep trackers you can buy. You just slip it under your sheets and sleep away, using the companion app to handle all of that data.

3.4 Xiaomi Mi Band 2

Xiaomi's latest wearable comes with its own stats app but you can just as easily sync it with heath application as well.

3.5 Human

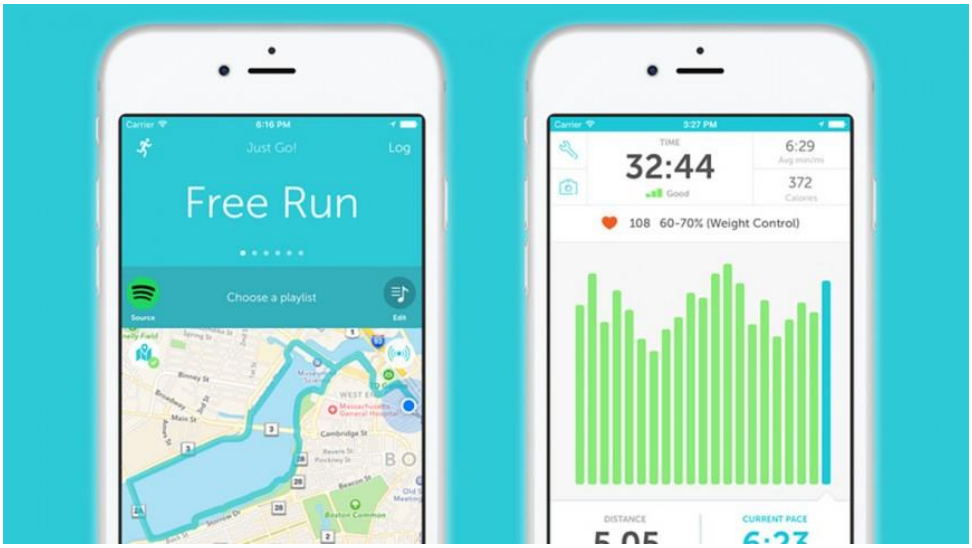


A simpler route to fitness, Human — it eschews complex workouts to focus on getting you active for 30 minutes every day, and you can then build up from there. Tracking is automatic on compatible iPhones and the Apple Watch.

3.6 Dance Party

Party like it's 2004 with this dance-off app that pairs your iPhone with an iMac or Apple TV. Mirror the moves on screen, and your device acts as a motion sensor to give you points. Yes, it's a fairly niche way of getting fit, and yes, your data gets synced to Health app.

3.7 Runkeeper

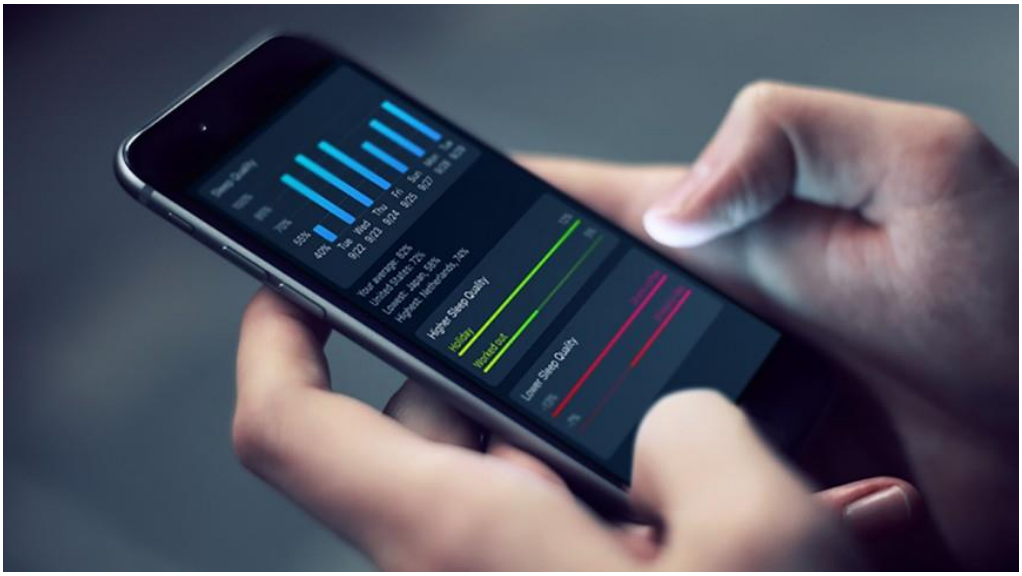


Runkeeper has been one of the most popular running trackers on the iPhone for many years now, and it syncs neatly with Apple Health if you want to keep all your runs in one place.

3.8 Weight Watchers

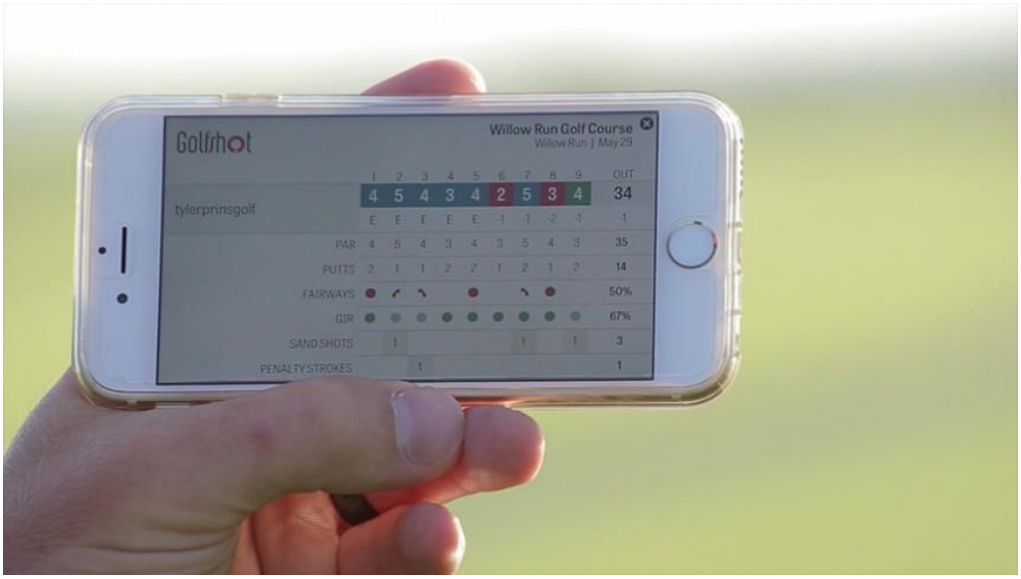
If you're a Weight Watchers member using the app for its food tracking database, Apple Health can display fat, carbs, fiber and protein information through one central dashboard. What's more, the same dashboard will show your points scores.

3.9 Sleep Cycle Alarm Clock



Sleep Cycle Alarm Clock is smart enough to track your sleep patterns through the noises you make. It can log your sleep time in Health app, and wake you up in the morning at the most natural time to fit your circadian rhythms.

3.10 GolfShot Plus



A more unusual addition to the Apple Health family, the popular golf app sits quietly in the background monitoring steps, calories, and the pace of your play, while you find new ways to ruin a nice walk.

4. SENSORS

The term body sensor refers to a variety of biomedical devices operating within, on or at close proximity to the human body. Some examples are pacemakers, implantable defibrillators, bionic eye, bionic ear, swallowaSensors like wireless capsule, neural recording and stimulation devices and wearable sensor devices in telemedicine for physiological signal sensing and monitoring. More body sensor devices will be developed in the future to undertake various body tasks and to diagnose chronic diseases. These devices are made wireless in order to increase patient comfort, to enable patient mobility, and to make the device robust. A wireless system is necessary for information transfer from these devices placed on or inside the body for purpose of controlling and monitoring. A wireless link designed for body sensor devices should form an intelligent communication network such as a body area network in order to work efficiently and safely in the same environment. This article presents recent sensor nodes developed for such medical networks.

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device. For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, such as tilt, shake, rotation, or swing. Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dewpoint, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing.

The Android platform supports three broad categories of sensors:

- **Motion sensors**

  These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

- **Environmental sensors**

  These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

- **Position sensors**

  These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

  You can access sensors available on the device and acquire raw sensor data by using the Android sensor framework. The sensor framework provides several classes and interfaces that help you perform a wide variety of sensor-related tasks. For example, you can use the sensor framework to do the following:

- Determine which sensors are available on a device.
- Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution.
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data.
- Register and unregister sensor event listeners that monitor sensor changes.

  This topic provides an overview of the sensors that are available on the Android platform. It also provides an introduction to the sensor framework.

4.1 Introduction to sensors:

The Android sensor framework lets you access many types of sensors. Some of these sensors are hardware-based and some are software-based. Hardware-based sensors are physical components built into a handset or tablet device. They derive their data by directly measuring specific environmental properties, such as acceleration, geomagnetic field strength, or angular change. Software-based sensors are not physical devices, although they mimic hardware-based sensors. Software-based sensors derive their data from one or more of the hardware-based sensors and are sometimes called virtual sensors or synthetic sensors. The linear acceleration sensor and the gravity sensor are examples of software-based sensors. Table 1 summarizes the sensors that are supported by the Android platform.

Few Android-powered devices have every type of sensor. For example, most handset devices and tablets have an accelerometer and a magnetometer, but fewer devices have barometers or thermometers. Also, a device can have more than one sensor of a given type. For example, a device can have two gravity sensors, each one having a different range.

Table 1. Sensor types supported by the Android platform.

Sensor	Type	Description	Common Uses
TYPE_ACCELEROMETER	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
TYPE_AMBIENT_ TEMPERATURE	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperature s.
TYPE_GRAVITY	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_LINEAR_ ACCELERATION	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
TYPE_MAGNETIC_FIELD	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.

TYPE_ORIENTATION	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the getRotationMatrix() method.	Determining device position.
TYPE_PRESSURE	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
TYPE_RELATIVE_ HUMIDITY	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_ VECTOR	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Measures the temperature of the device in degrees Celsius (°C). This sensor Implementation varies across devices and this sensor was replaced with the TYPE_AMBIENT_TEMP ERATURE sensor in API Level 14	Monitoring temperature .

4.2 Sensor Framework

You can access these sensors and acquire raw sensor data by using the Android sensor framework. The sensor framework is part of the android.hardware package and includes the following classes and interfaces:

4.2.1 Sensor Manager

You can use this class to create an instance of the sensor service. This class provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information. This class also provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

4.2.2 Sensor

You can use this class to create an instance of a specific sensor. This class provides various methods that let you determine a sensor's capabilities.

4.2.3 Sensor Event

The system uses this class to create a sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.

4.2.4 Sensor Event Listener

You can use this interface to create two call back methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

In a typical application you use these sensor-related APIs to perform two basic tasks:

- **Identifying sensors and sensor capabilities**

Identifying sensors and sensor capabilities at runtime is useful if your application has features that rely on specific sensor types or capabilities. For example, you may want to identify all of the sensors that are present on a device and disable any application features that rely on sensors that are not present. Likewise, you may want to identify all of the sensors of a given type so you can choose the sensor implementation that has the optimum performance for your application.

- **Monitor sensor events**

  Monitoring sensor events is how you acquire raw sensor data. A sensor event occurs every time a sensor detects a change in the parameters it is measuring. A sensor event provides you with four pieces of information: the name of the sensor that triggered the event, the timestamp for the event, the accuracy of the event, and the raw sensor data that triggered the event.

  Sensors are moving from external devices to the body. We've gone from the Six Million Dollar Man to the common folk with a smart watch, fitness tracker, smart wristband, GPS on his wrist, barometer in his pocket, heart monitor on his sleeve, and this is only the beginning. Researchers are working on a continuous blood pressure monitor, and a Lab-on-a-Chip that will enable anyone to carry a continuous glucose monitoring sensor, among many other things. The human body is the New Continent and we are all pilgrims. However, for the data collected to be usable, devices must be able to work together.

  For instance, in the case of body motion tracking, the synchronicity of the sampling and streaming activities is essential to ensure the quality and usability of the data. In other words, to determine if a body is running, jumping, falling, etc. all the sensors must pull data at precise times and intervals to track all the aspects of the body simultaneously and appropriately. If sensors on the leg capture data at one point in time and those on the arms pull data at a completely different moment, models will no longer work and the system won't have an accurate picture of what is happening.

Problems of the Body Area Network

Hence comes Andrea Vitali to the rescue. He's an Advanced Research Manager at ST's Internet of Things Excellence Center. As a system architect, he and his team design prototypes based on clients' requirements and specification, collaborating with them to deliver the result they expect and more. This has led him to work on SensorTile, the development kit that packs a tremendous number of sensors in a 13.5 mm x 13.5 mm board. As a result, he's at the forefront of the revolution happening in the field of Body Area Network.

As the name implies, the Body Area Network (BAN) refers to the intercommunication of the plethora of sensors worn, attached on or even inserted in the body. From simple accelerometers, gyroscopes, barometers, and digital microphones, to much more complex sensors, this network of wearaSensors faces very difficult technological challenges. Indeed, BAN clients often come from different manufacturers and function very differently, yet they must all gather data at the same time or use accurate timestamps to enable the alignment of the information. It's the only way for the system to know exactly what the entire body was doing at an exact moment in time. Today, Bluetooth Low Energy is the network of choice for these devices, but the standard was never designed for this.

Solutions for Your Body Area Network

To overcome this challenge, Andrea and his team have developed different solutions that he will present in his talk entitled *Body Area Network and sensor synchronization.* With the help of demos, executables, and documentation available to the attendees, he will show how developers can master the complexities of these devices to quickly and safely achieve project goals.

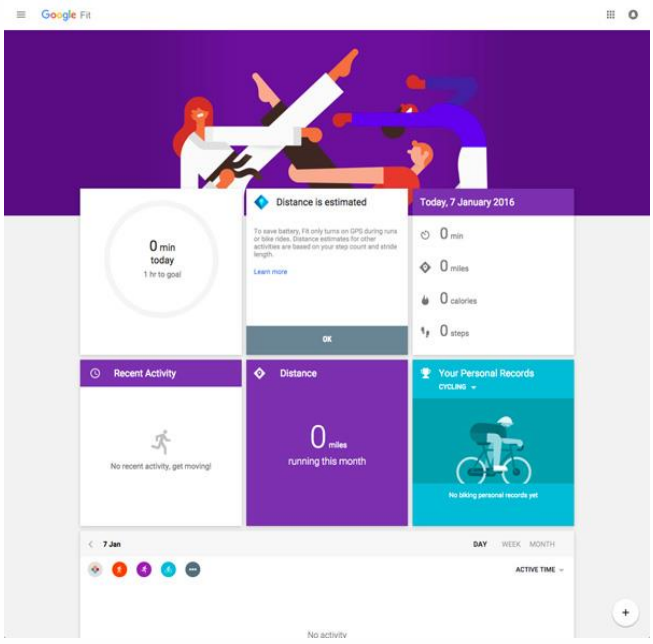
Andrea will show how one can use Bluetooth Low Energy features, like broadcasting or receiving a trigger signal, to ensure the gathering and streaming of data in unison. Hence, anyone involved in designing a system using those types of sensors will have hands-on practical examples to immediately see the result of certain practices or codes. Whether you are aiming to mass produce a device in the millions of units or simply launch your crowdfunding campaign, mastering the Body Area Network is the key to transforming a design from great to truly remarkable by ensuring fast and reliable communication with other sensors and the host system.

5. GOOGLE FIT

Google Fit is included in Google Play Services and allows developers to store fitness data for users to help users keep track of their exercise habits. All data stored through Google Fit is stored online and is accessible by the user from multiple devices. The user never has to worry about losing their information if they upgrade or change devices.

Connect to the Google Fit platform to read and store a user's fitness data across devices, collect activity data, and record sensor data. The Google Fit platform drives retention, higher ratings, and faster development for health and fitness applications.

Google Fit gives you a single set of APIs to access — with the user's permission — phone sensor data, wearable data, and other apps' fitness data. Once you integrate using the Google Fit APIs, any new data source that becomes available will work, without the need for additional integration.



5.1 Key Features

- **Low power, passive recording of activities, steps, and distances**. After an app subscribes using the Recording API, Fit turns on sensors at appropriate times throughout the day and records activity data that an app can query later.
- **Computes distance and calories by transforming sensor data**. If your application only writes steps and activity data to Fit, the platform will calculate corresponding distances and calorie estimates. This is one less computation for your fitness app to perform.
- **Write to and read from a centralized fitness store**. Fit acts as a hub of fitness data from connected apps and devices. Fit keeps the user's full fitness history, making it available to apps as raw, aggregated, or merged data. These include fitness data points for nutrition, workout sessions, and steps from connected apps, wearables, smart scales, and more.
- **React to changes**. Fit allows apps to register listeners for data changes in the fitness store. For example, if a user logs a new meal or a new weight entry, your app can be notified and deliver a timely, customized notification to the user to re-engage them.

5.2 Getting Started

Before users can use the Google Fit features of your application, you need to register your application through the Google Developers Console. You can then enable the Google Fit API in the developer's console and create an OAuth client ID that can be used for your app to access fitness data stored through Google.

Once you have a client ID that can be used in your application, you need to connect a Google API Client in your application and request the necessary features for your application.

Once your application is set up and the user has granted your application permission to access their information, you are ready to go.

5.3 APIs

Google Fit for Android is made up of various APIs that allow developers to perform specific tasks.

Here we have a general idea of what each API does.

- **Sensors**

  This API allows you to access the raw information from sensors on the device. Not only can you access standard hardware on a mobile phone, but you can also access sensor data from accessories, such as an Android Wear device.

- **Recording**

  The Recording API allows your application to subscribe to specific pieces of data and automatically store them. This lets the user or your application access this information from any device that the user has granted permission to.

- **History**

  This API lets you access an archival database. You can perform standard operations, such as inserting, deleting, and querying fitness data that has been previously stored.

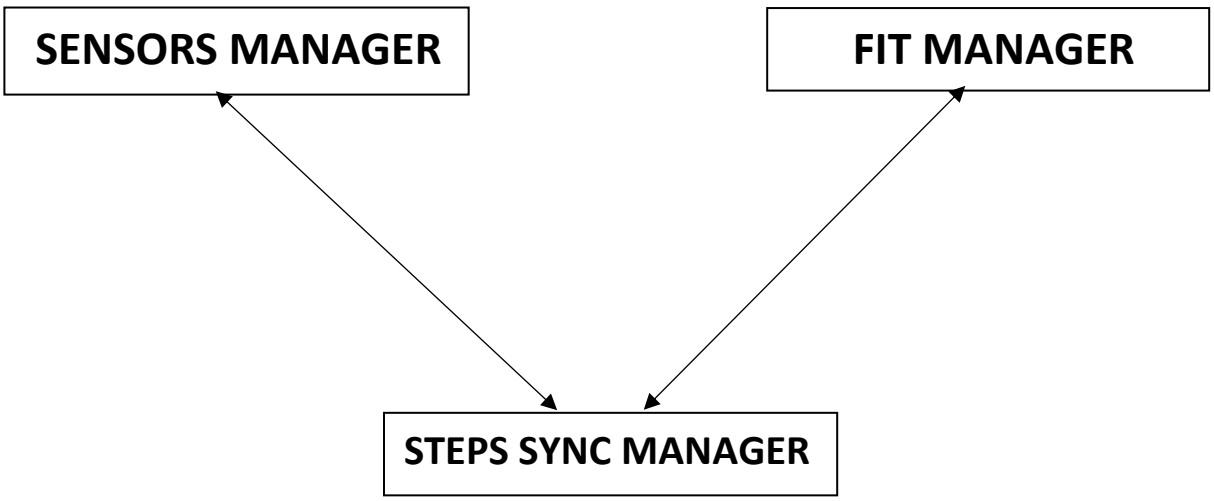
- **Sessions**

  The Sessions API allows your application to record data over a period of time in order to aggregate information. This can be useful for situations where data without a timeframe or an associated activity isn't as useful as data with a context, such as the user's heart rate during a run.

- **Bluetooth Low Energy(BLE)**

  While the Sensors API allows your application to access sensor data from wearables that support Google Fit, the Bluetooth Low Energy API allows your application direct access to Bluetooth devices in order to store data from them.

5.4 Components



The simple idea behind the architecture shown in the diagram above was to have the following three main components:

- **Sensor Manager**: The Sensors manager is responsible for all the Sensors communication with the wearable and to provide a simple Java backed API to send and receive data from the device.
- **Fit Manager**: The fit manager is responsible for handling all the communication with the Google Fit API, it also provides a simple Java backed APIs to read and write steps data to the Fit API.
- **Steps Sync Manager**: The steps sync manager is responsible for connecting the **Sensor Manager** and the **Fit Manager** together by retrieving all the steps data from the **Sensor Manager** and passing them to the **Fit Manager** to be stored using the Google Fit History API.

5.5 Implementation

5.5.1 Fit Manager

When implementing the Google Fit API, we wanted to use Java to simplify some of the complicated APIs that would be commonly used. This led to the creation of **Fit Manager**.

5.4.2 Sensor Manager

The **Sensor Manager** is responsible for providing the ability to easily connect to the Sensor, send commands, and retrieve data from the device.
To build the API client we have to specify a couple of values.

- The first value is the API we want to use, if we want to use the Google Fit history API it can be specified using the Fitness. SENSORS_API constant.
- We also have to specify the access scope we want to that particular API, if we need both read and write access to the history API to be able to save new steps data and read them again later. This is specified using the constant value Scopes.FITNESS_ACTIVITY_READ_WRITE.

Finally, we need to provide a connection Callbacks listener and connection failed listener.

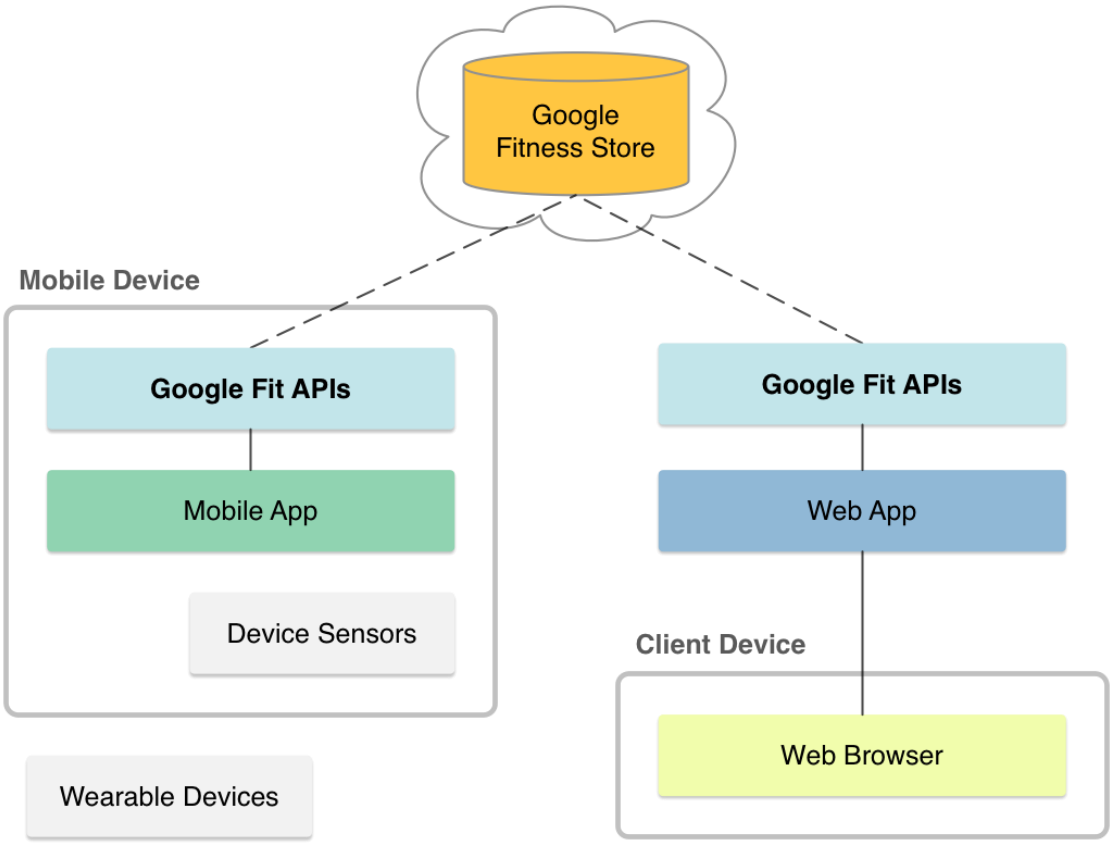
5.4.3 Steps Sync Manager

Finally, the **Steps Sync Manager** is responsible for connecting the **Sensor Manager** and **Fit Manager** together by retrieving steps data from our **Sensor Manager** and pushing them to the **Fit Manager**.

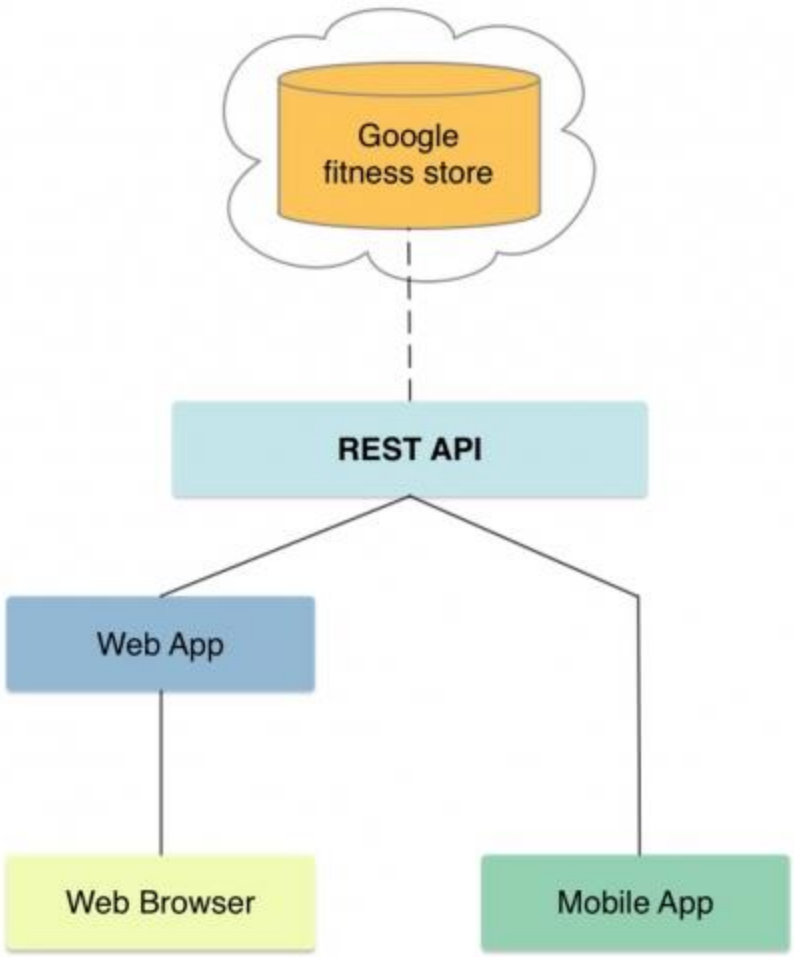
The **Steps Sync Manager** is started through a sticky service and kept running in the background all the time. This insures steps data are kept up to data with the Google Fit API, and if the users were to open the Google Fit App at any moment they would be represented with an accurate count of their steps for the day.

Steps syncing is divided into two parts: the first is retrieving the steps data from our Sensor and the second pushing that data to the Google Fit API.

5.6 Architectural Overview

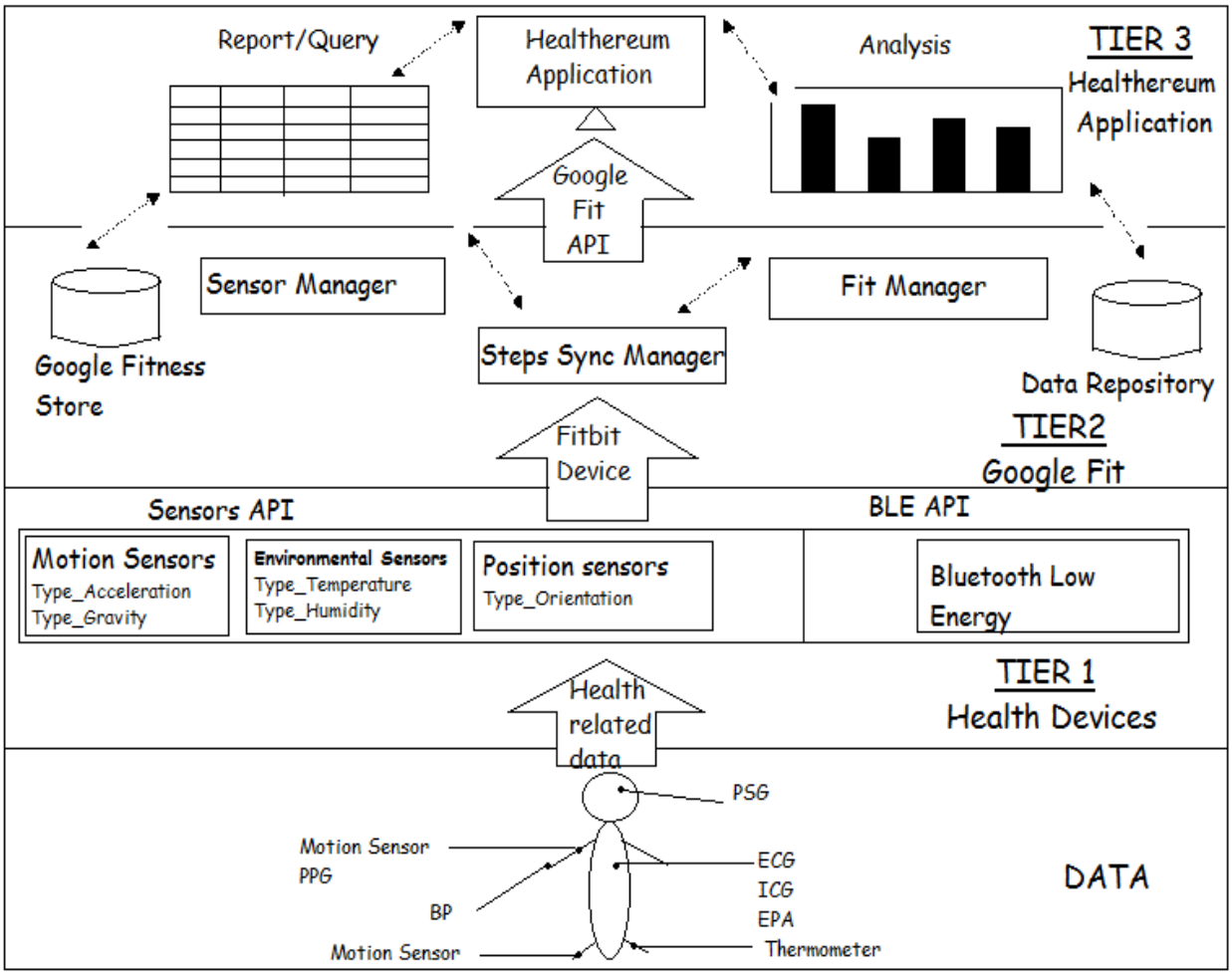


Google Fit consists of the Fitness Store, the persistence data cloud layer that stores data from disparate devices and sources; Sensor Framework, which is an abstract representation of sensors and fitness types (such as step count and heart rate) with the ability to query and interact with the database; and Permissions/User Controls, which manages user permissions and consent.



An example architectural diagram using google fit's REST API

6. HIGH LEVEL ARCHITECTURE



A Three Tier Architecture showing the dataflow from the health devices to Healthereum application.

6.1 Tier 1 or Bottom Tier is concerned with the health device like Fitbit which retrieves the data from human body using the Sensors APIs.

The Android platform supports three broad categories of sensors:

- **Motion sensors**

  These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

- **Environmental sensors**

  These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

- **Position sensors**

  These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

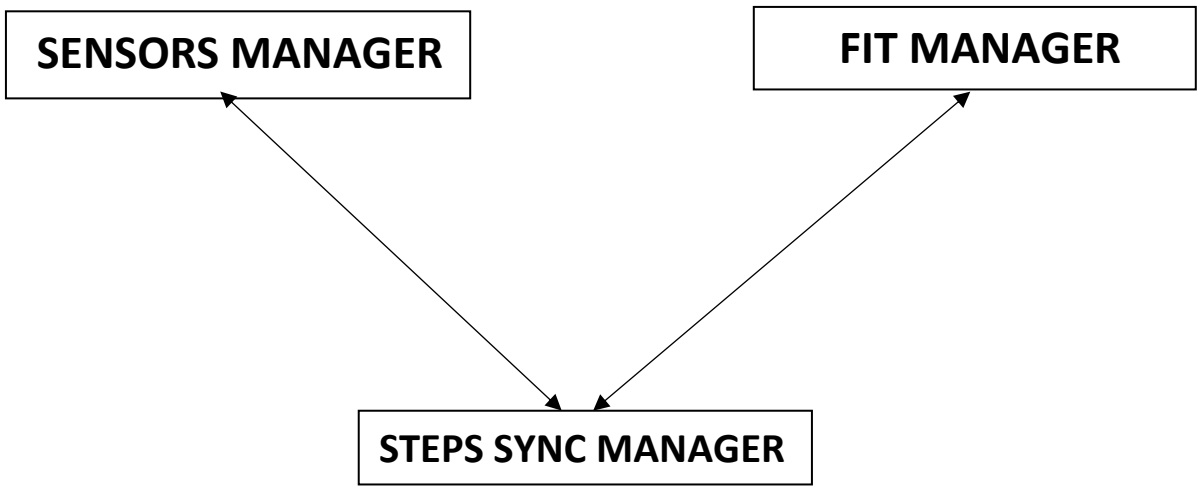
You can access sensors available on the device and acquire raw sensor data by using the Android sensor framework. The sensor framework provides several classes and interfaces that help you perform a wide variety of sensor-related tasks. For example, you can use the sensor framework to do the following:

- Determine which sensors are available on a device.
- Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution.
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data.
- Register and unregister sensor event listeners that monitor sensor changes.

6.2 Tier 2 or Middle Tier is concerned with the Google Fit platform which manages the flow of data between the device and the health application.

Google Fit gives you a single set of APIs to access — with the user's permission — phone sensor data, wearable data, and other app's fitness data. Once you integrate using the Google Fit APIs, any new data source that becomes available will work, without the need for additional integration.

6.2.1 Components



The simple idea behind the architecture shown in the diagram above was to have the following three main components:

- **Sensors Manager**: The Sensors Manager is responsible for all the BLE communication with the wearable and to provide a simple Java backed API to send and receive data from the device.
- **Fit Manager**: The fit manager is responsible for handling all the communication with the Google Fit API, it also provides a simple Java backed APIs to read and write steps data to the Fit API.
- **Steps Sync Manager**: The steps sync manager is responsible for connecting the **Sensors Manager** and the **Fit Manager** together by retrieving all the steps data from the **Sensors Manager** and passing them to the **Fit Manager** to be stored using the Google Fit History API.

It also uses Google Fitness store and Data Repository for managing and organizing the data between the device and the app.

The Google Fit API provides great value at a very low cost for both the developer and user. It removes all the complications that come with implementing your own backend to store all the steps data, while providing the user with a tremendous set of tools to interact and better analyze the collected steps data using the Google Fit App.

6.3 Tier 3 or Top Tier is concerned with the Health application Helthereum which uses the data stored in Google Fit by connecting to it in the following way:

Each app you connect will have a different process to connect it. But generally, you can:

- Open the app you want to connect, like Fitbit, Runkeeper, or MyFitnessPal.
- Look for the Settings menu.
- Look for the setting to connect other apps and devices. Depending on the app, this might be called 'Link other services,' 'apps & devices,' 'manage connections,' or something else.
- Follow the onscreen instructions to link the app to Google Fit Google Fit.

  The Healthereum app uses this data from Google Fit for **data analysis(dashboards)** and **Report**

Apps connected to Google Fit share and use your info in one of two ways:

- **Collaborative sharing:** Usually, apps let other connected apps view the fitness data they've stored in Google Fit.
- **Isolated storing:** Sometimes, app store data in Google Fit without letting other connect apps see it.

7. MANAGE GOOGLE FIT SETTINGS

Before you connect an app to Google Fit, Google will ask if you want to allow this app to store new data to Google Fit and view data you've already stored in Google Fit.

- **Store new data:** The app can save information to your Google Account on Google Fit. Only some apps store data and when they do, any other apps or devices that you've given permission to view data can use that information that they've stored in your Google Fit account.
- **View existing data:** The app can use information saved to Google Fit by other connected apps.

  After you give permission, a connected app can access information in your Google Fit account from any device where you have it installed. If you give an app permission to connect to Google Fit on one device, you can download that app on a second device and it'll automatically connect to your Google Fit account.

With Google Fit, you can keep all your fitness information tied to your Google Account and share it with the other apps and devices you use to stay healthy. When you want to check in or get insights about your activities, open Google Fit.

The fitness data in Google Fit belongs to you. You can grant apps access to use the data and add to it, and remove access whenever you want from the Google Settings app. You can also delete the data stored in Google Fit at any time.

7.1 Connect apps with Google Fit

You need to connect an app with Fit to share the data from that app with Fit. Each app you connect will have a different process to connect it. But generally, you can:

- Open the app you want to connect, like Runkeeper, or MyFitnessPal.
- Look for the Settings menu.
- Look for the setting to connect other apps and devices. Depending on the app, this might be called 'Link other services,' 'apps & devices,' 'manage connections,' or something else.
- Follow the onscreen instructions to link the app to Google Fit Google Fit.

7.2 See the connected apps

You can control which apps share data with Fit.

- Open the Google Fit app Google Fit.
- Tap Menu and then Settings.
- Under "Google Fit data," tap Connected apps.

7.3 To **disconnect** an app from Google Fit, tap on the app and then Disconnect.

If you disconnect an app from Google Fit, it will be disconnected on all devices where that app is installed. Disconnecting an app will not delete any data that the app stored in Google Fit.

7.4 Delete your Google Fit history

To delete the information your apps and device have stored in Google Fit, follow these steps:

- Open the Google Fit app Google Fit.
- Tap Menu and then Settings.
- Scroll down and tap Delete history and then Delete history.
- Check the box next to “I understand and want to delete.”
- Tap Delete.

Deleting your history will only remove saved information from your Google Account. Other apps or devices might have similar data stored in their own services, and you’ll need to use their settings if you want to delete that information.

8. LIMITATIONS OF GOOGLE FIT

8.1 History API

In the process of reading the google fit history using the android API, we come across an annoying limitation of only being able to read the past week or so's worth of history, even though there is vastly more data (according to the google fit app).

By default, your request will query only local data stored on the device. To include data from Google servers add .enableServerQueries() to the DataReadRequest.Builder:

```
DataReadRequest readRequest = new DataReadRequest.Builder()
        .aggregate(DataType.TYPE_STEP_COUNT_DELTA,
DataType.AGGREGATE_STEP_COUNT_DELTA)
        .bucketByTime(1, TimeUnit.DAYS)
        .setTimeRange(startTime, endTime, TimeUnit.MILLISECONDS)
        .enableServerQueries()
        .build();
```

enableServerQueries()

Enables querying the Google Fit server to fetch query results, in case the local store doesn't have data for the full requested time range. Server results will be combined with local results into one DataSet.

8.2 Storage

- We use the custom data fields to store content that cannot be appropriately stored in the com.google namespace.

- If at any time content collected by your API Client could be appropriately stored in the com.google namespace, then such content is stored in the com.google namespace and not the custom data fields.

- The content your API Client stores in the com.google namespace will be of at least the same level of granularity as the content accessed by your API Client. For example, if your API Client accesses step count from Google Fit on a per-day basis, then your API Client should also store step count on Google Fit on a per-day basis, rather than per week or per month.

8.3 Back ups

Backing up the content is another responsibility. Google reserves the right to set limits on the length of time, content may be accessible on Google Fit. Where reasonable, notice of those limits will be given within the Google Fit documentation before Google sets them.

8.4 Deletion of Data

Our App is responsible for complying with any request by a user to remove content. Google is not responsible for removing content that has been stored on third-party services or your own services.

9. CONCLUSION

The Google Fit API provides great value at a very low cost for both the developer and user. It removes all the complications that come with implementing your own backend to store all the steps data, while providing the user with a tremendous set of tools to interact and better analyse the collected steps data using the Google Fit App.

10. LINK TO SOURCE CODE

https://github.com/hitesh6473/Integration-of-health-devices.git

11. REFERENCES:

1. Global Status Report on Noncommunicable Diseases 2010, World Health Organization, 2011; www.who.int/ nmh/publications/ncd_report2010/en/.

2. J. Meyer and H. Hein, "Live Long and Prosper: Potentials of Low-Cost Consumer Devices for the Prevention of Cardiovascular Diseases," J. Medical Internet Research, vol. 2, no. 2, 2013.

3. C. Tudor-Locke et al., "How Many Steps/Day Are Enough? For Adults," Int'l J. Behavioral Nutrition and Physical Activity, vol. 8, no. 1, 2011, p. 79; doi:10.1186/1479-5868-8-79.

4. E. Hawley et al., "Movement Toward a Novel Activity Monitoring Device," Sleep and Breathing, vol. 16, no, 3, 2012, pp. 913–917.

5. https://youtu.be/hS8OLV-D1lw

6. https://youtu.be/jwaQaWGLlTo