FINAL SUBMISSION
Fitness Tracking Software

Dream Team

Michael Gravino	27088140	michael.gravino@gmail.com
Matthew Masi	27039956	masimatthew@gmail.com
Rabih El-Bitar	27013647	rabih.el.bitar@hotmail.com
Lucia Racine	26462111	lucia_racine@outlook.com

Concordia University
Engineering and Computer Science

December 5th 2016
Dream Team
Coen/Elec 390

Table of Contents

Table of Tables

Table of Figures

Abstract

Technology has developed into changing the way people live their lives. The application created for this project has helped a number of people with the way that they workout and view their progression through improvement. The application created is the Fitness Tracking Software (FTS) application. The manifesto for this application is to motivate people to document their workouts in order to see growth within their own success. The application has multiple functions that include timing the run and calculating the distance traveled, average speed, and calories burned during the workout. A database then collects this information and makes a convenient layout for the user to view their results and statistics from the previous five runs that have been done.

The application has also implemented a trophy system where the user is rewarded for their efforts as a way to motivate and encourage the user to continue their progression forwards to continue seeing the desired results without internet. The motivation behind the creation of this application was to track the progress being made for a typical runner who has difficulty motivating themselves.

Abbreviations & Definitions

Term	Definition
User	Owner of Mobile Device who uses the software
GPS	Global Positioning System
FTS	Fitness Tracking Software acronym
API	Application Programming Interface
SQLite	Client-server database engine
App	Application
PO	Product Owner
App Store	Application Store, used to download applications onto a device
GPS-Navigator	Provided software on mobile device used to provide connection to the GPS and show locations, give data
Stand-Alone	An object, person or software that doesn't require anything to function other than itself.
Download and Use	Personal expression, describes an application so simple to use that the user must only download the application and upon viewing the interface know exactly how to use the software with ease without needing a tutorial or manual
Home Page	First interface that the user will see upon opening the application
MB	Megabytes
GB	Gigabytes
RAM	Random-Access Memory
OAuth Library	Library within the Google Client API that will optimize the coding aspect inside Android
DESC	Description
RAT	Rational
DEP	Dependency
Class Diagram	A diagram that shows the classes defined in a software system, their methods, their interrelationships, and any other information related to the organisation of the software into classes and objects
Sequence Diagram or SD	A diagram that shows the classes and the sequence of method calls involved in performing a specific action in a specific scenario. (Acronym = SD)
UML	Unified Model Language, a modelling language used in software engineering. It defines various types of models and diagrams, including component, class, use case, sequence and communication diagrams.
MVC	Model-View-Controller is a software design pattern for implementing user interfaces on computers.

Table 1: Abbreviations & Definitions

I. SOFTWARE REQUIREMENTS SPECIFICATION

1. Introduction

This particular section of the document provides a broader scope of the SRS document and what you should expect in a general sense of the software.It will also provide more detail on key information to properly read this document and development teams vision of the Software that awaits the approval of the client.

1.1 Purpose

This document represents and describes a basic overview of the software requirements specifications. In other words, this document will provide details on what the Fitness Tracking Software must do in order to comply to the specifications given by clients and Product Owner. With this, the designers will be able to build the application in question in the vision of the client. The intended audience of this document is the client, which ordered this software to be built, for his approval of the first version of the software. The development team can then build the first version of said software.

1.2 Scope

The "Fitness Tracking Software" application is a GPS localization mobile app that will essentially track the users movement during runs. In other words, this application will track the users distinct route choosen when on a run and will display the entire course from start to finish on a map. This application will be free to download and will be available on the Play store.

This application should have a profile page where the user inputs their height, weight, age and sexe, from which the "Fitness Tracking Software" application can calculate an approximation of calories burnt per run. The application should also display the users best runs based on either time or distance ran.

This software is meant to help the user to properly see their improvement over the course of their training and also give them exactly the right information about the distance of their runs as well as the time to complete their runs. The Fitness Tracking Software's goal is to continue to motivate the users to live a happy and healthy life style and make sure they are improving after every single run. It also provides information that they might feel useful when regarding their runs.

1.3 Overview

The rest of this document will have 5 more sections. The next section will describe in detail the overall description of the FTS, the way it interconnects itself with different systems while in use as well as it’s functionality. It will also include the characteristics and the constraints of the application.

Section 3 will specify the functional and non functional requirements and will show the use cases.

The fourth section will the change management process of the SRS for the “Fitness Tracking Software.

The fifth section of this document will provide the names of the people that will approve this document.

The final and sixth section of this document provide extra information for the software and the SRS document for the development team as well as the client.

2. The Overall Description

For the “Fitness Tracking Software: application, it will feature a database to store information acquired, and will utilize the GPS of the users phone to help track movements in order to map out there training. Also, this section will describe what kind of users will interact with this software and will define in more lamens terms the functionality of the system. In addition it will present the user characteristics and the constraints.

2.1 Product Perspective

Fitness tracking is an independent and totally self-contained mobile application. The application will need to use Google Play Services API Client to use the mobile device’s embedded GPS to get the location of the user. The GPS will periodically provide the mobile application with the location of the user. It will also provide the maps and functionality to display the data on the map.

In order to keep track of the data and properly display the users improvement, we must use a database to store it. In this case we will be using SQLite, which is a client-server database engine utilized with Android Studio, to communicate with the FTS. This part of the system is crucial to the clients goal of encouraging the user to keep working hard to see improvement. As the block diagram in figure 1 points out, the Google API Client is in direct communication with the “Fitness Tracking Software” application and in turn communicates with the GPS device embedded in the mobile device as describe in the former paragraph.

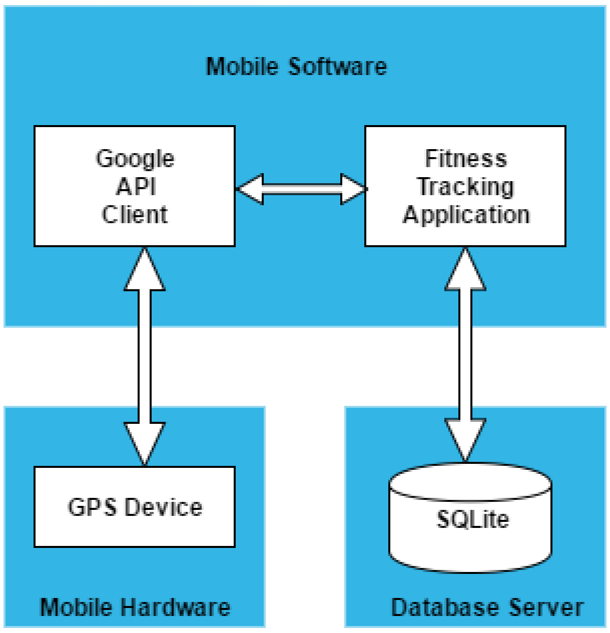


Figure 1: Block Diagram of System

The FTS application will be using SQLite to store the data and the GPS to acquire the current location of the user. The data stored will have a limit of a max of 10 mb or top runs stored, whereas the GPS of the device will utilize the location tools within the mobile device. In other words, this application will not severely impact the storage of the device nor battery usage to an extreme extent due to the simplicity of the application.

2.1.1 System Interfaces

This application doesn't have system interfaces since it doesn't interact with an existing web application. In other words this application is stand-alone and doesn't require any outside applications to function correctly.

2.1.2 Interfaces

The logical characteristics of the interface between the software product and its users is not complicated at all. The user will be greeted by the "home page" if you will of the "Fitness Tracking Software" application, once the user is ready to begin his run, he will then press a button indicating that he has started. Upon activation, the software will communicate with the Google API Client and in turn utilize the mobile devices GPS to obtain the users location periodically during his training. Once the user has completed his or hers run, he clicks on the button that will stop the communication between the Fitness Tracking Application and begin its communication to the SQLite database in order to store all the data just obtained. It will then display the course taken from the user on a google map interface and display the users distance ran and time completed, as well as a couple of other features.

The logical interface will utilize a GUI described in section 3 as well as a optimized interface for the user to interact with in a fairly simple way in order to invoke the "Download and Use" mentality for the user. The "Home Page" of the application will contain three to six buttons, one will invoke the Start of the Exercise (once started the same button will be used to stop the applications communication with the Google API Client, thus the end of the users exercise). The second button will open an optional profile page where the user can add their characteristics such as height and weight to obtain a few more features after every run. As well as a third button that will display the users top runs using the software. We would have a fourth button that will allow the user to pause during the course of their run to see how he or she is doing, it will turn into an unpause button once activated so that the user can continue his or her run afterwards. Two other buttons will be available to view the map and to view the statistics of the user.

2.1.3 Hardware Interfaces

The interface for the "Fitness Tracking Software" application contains 4 crucial external systems that much interact themselves to have a proper functioning app. Like explained in the previous subsection we have in order from left to right as seen on figure 1 of this document; Google API Client, Fitness Tracking Application, GPS Device and SQLite.

The Google API Client simply interacts with both the FTS and the GPS device embedded in the system. It will communicate with the GPS Device from the users mobile phone after being authorized to get data periodically and properly track the users movements when they are exercising. It then relays the information to the Fitness Tracking Application that stores it within SQLite's database for further organization and flexibility.

The Fitness Tracking Application, like explained above and the previous subsection, contains the user interface, the exterior shell of the app. It will ease the stakeholders use of the product and be the main thing the user sees. It interacts with the SQLite database and the Google API Client hand in hand to properly deliver the users data such as route taken, distance accomplished, time per course, etc.

The GPS Device is simply a piece of hardware already available on the mobile device. The application will ask the user upon the applications start and will provide through the Google API Client the proper tracking to the app.

Last but not least the SQLite is simply a database server to store all the data that is fed to the Fitness Tracking Application through its constant communication with the Google API Client. It will also be rearrange an given back to the Fitness Tracking application in the form of a scoreboard for best runs or longest distance ran.

2.1.4 Software Interfaces

With the FTS in this case we interact with 2 different software interfaces. First we need to use the Google API Client version 1.22.0 for java to properly manage the data received from the GPS Device hardware embedded in the mobile device. The Google API Client also utilizes the OAuth library 2.0 to optimize the lines of code in a more manageable way.

The second and last software interface that the FTS application communicates with is SQLite. Android studio utilizes SQLite3 version 3.9 in the most recent android API Version 24. This software interface will keep all data inputted within the FTS application to minimize the overall memory usage from the FTS onto the phone as well as help organize the data in a way that the client demanded. It will communicate within the FTS like described in section 2.1.2.

2.1.5 Communications Interfaces

For the FTS application, it will not utilize any custom protocol to communicate between systems, instead it will utilize the Google API Client to effectively to that for them. Once upon activating the app, the user will be prompted by the phone to authorize access to the GPS device embedded within itself and the Google API will do the rest in terms of communicating within the interfaces. Same goes for the SQLite interface with the FTS application.

2.1.6 Memory Constraints

Focus groups have determined that the target market has between 512MB to 1 GB of RAM, therefore the design footprint should be between 32 and 128 MB of Random-Access Memory to ensure a fluid functionality for all features with the FTS application Software and other applications on the users mobile device.

2.1.7 Operations

The Operations of this system have 2 different modes of operation. One would be once the user starts the exercise, it would then communicated with the Google API client. Since this application is stand-alone and requires access to the localization tool of the mobile device hence the internet, if the the mobile device cannot access the localization tool within the GPS the tracking will not work for that particular moment.

2.1.8 Site Adaptation Requirements

There is absolutely no site adaptation requirements for the use of the "Fitness Tracking Software" application. Like explained in a previous subsection of this SRS document, if the GPS' localization tool has access to the internet then the application will function as planned.

2.2 Product Functions

With the application, the user will be able to get a feedback after going for a run. The main function of the application is tracking the route of the run. In other words, at the end of the run, the application will display the route taken by the runner.

FTS will calculate the time to complete a run and show its total distance. The user will get this information at the end of the run on a scoreboard. This scoreboard will also include the number of calories burnt during the run.

The application will save the information collected so it can compare it to other information collected from other runs. This information will posted at the end of a run showing the best run took so far.
The functions of FTS are:

- ➢ Map out the route of a run
- ➢ Calculate total time to complete the run
- ➢ Calculate the distance
- ➢ Calculate the number of calories burnt
- ➢ Store the data collected for several of the best runs
- ➢ Make a history of best runs

2.3 User Characteristics

It is not complicated to use the “Fitness Tracking Software”. The user does not need a certain level of education or expertise to be able to get the benefits from the application. Having the minimum experience in using an android is sufficient to use FTS.

Therefore, there is not only one type of target user, anyone can simply use the application. More precisely, the user must be able to open the application, run it before starting, and stop it when the run is done. Finally, read the recorded information.

Special skills and requirement are not needed to be able to work with FTS. If interested anyone who is willing to go for a run can easily run the application and get use of its functionality. This application will be only used for this purpose.

2.4 Constraints

FTS will use the GPS device within the phone to track the runner and record the route taken. There are many different systems that provide this service which will not be used in the same way, which make the use of GPS a constraint.

Another constraint for the application is the internet connection. The application will track and follow the runner. The user will need database service to be able to have an internet connection all the time. To be able to function, the app needs internet connection

2.5 Assumptions and Dependencies

As stated before, the application will be using the GPS, and one assumption is that all GPS devices within the phone work is the same way. Which means that the development of the application consider all GPS similar. The application will need adjustment if the GPS components differ from a device to another.

Another assumption made is that all phones have internet connections, which means that anyone can use the application. In this assumption the number of users is limited to the ones that always have access to internet connection.

For the final assumption, the mobile device that the user will be utilizing the FTS application as a minmum android version of Android 4.2 (Jelly Bean).

2.6 Apportioning of Requirements.

Future version of the application will be able to calculate the heart rate and display it in the scoreboard with all the other information recorded, listed in Product Function section. To implement such an idea much more time is needed as well as other components.

To be able to record the heart rate of the user and display it on the phone, the application should use sensors that count the heartbeat to make a rate, for example: number of heartbeats in 10 seconds.

This requirement is not a priority to the application but it is a feature that could developed latter and added to the existing application.

3. Specific Requirements

This section contain all the software requirement of the system and provide a detailed description of the of all the system's features. This section will help the design team properly assess and quantify the requirements within the "Fitness Tracking Software" application and will be the baseline to designing the app.

3.1 External Interfaces

This section describes all inputs and outputs from the software system. It will give in greater detail the outlook of the application and give the designers a basis of what the client expects in a more technical termed way.

3.1.1 User Interfaces

When a user opens the application, a welcome page is displayed. The welcome page will have seven buttons that will lead to other pages or other functions.

This page will have button that the user will press to be able to run the application (Start). After completing a run, the user should stop the collecting information, and to do so he/she should press a stop button (Stop).

The home page of the application will contain a button (Profile) that once it pressed it will redirect the user to another page to add the weight and the height to be able to calculate the number of calories burnt.

Another button is on the home page to display the user's top runs on a scoreboard (Scoreboard). Also, a button is used to pause in the middle of the run (Pause). This button is pressed again to continue the exercise.

As explained in section 2 of this document, the homepage will have to other buttons that are used to display the map and the characteristics of the user (View map and View Stats).

The timer will be given in seconds, the distance will be given in kilometers and calories burnt will be given in calories. In the profile page, the Gender will have the choice of M for male or F for female, height will be given in centimeters and weight will be given in kilograms.

3.1.2 Hardware Interfaces

The "Fitness Tracking Software" application will only use the GPS embedded within the mobile device as its only hardware.

3.1.3 Software Interfaces

The FTS application will utilize the Google API Client to communicate with the GPS hardware within the device to properly track the users location after the press of the start button as seen in figure 2. It will localize the user every 5 to 10 seconds to properly track his or hers movements and once the user is finished he can then stop the localization, effectively ending his run and it will then display the exact route the user has taken. It will also display other metrics such as distance traveled, calories burned (if profile has been filled out) and duration of activity. The "Fitness Tracking Software" application can then communicate with the SQLite database through the internet to keep the load of memory used onto the mobile device to a minimum.

3.2 Functions

This section includes all the functional requirements of the system that describe all the specific actions that take place in the system.

3.2.1 Functional Requirement 1

ID:	**FR1**
TITLE:	Download mobile application
DESC:	The user should be able to download the app through an app marketplace or from any other source.
RAT:	In order for the user to download the mobile application.
DEP:	-

3.2.2 Functional Requirement 2

ID:	**FR2**
TITLE:	Mobile application – Start timer
DESC:	The user should be able to start the timer when pressing the start button.
RAT:	In order for the user to start the timer.
DEP:	FR1

3.2.3 Functional Requirement 3

ID:	**FR3**
TITLE:	Mobile application – Stop timer
DESC:	The user should be able to stop the timer when pressing the stop button.
RAT:	In order for the user to stop the timer.
DEP:	FR2

3.2.4 Functional Requirement 4

ID:	**FR4**
TITLE:	Mobile application – Pause timer
DESC:	The user should be able to pause the timer when pressing the pause button.
RAT:	In order for the user to pause the timer.
DEP:	FR1

3.2.5 Functional Requirement 5

ID:	**FR5**
TITLE:	Mobile application – Unpause timer
DESC:	The user should be able to unpause the timer when pressing the unpause button.
RAT:	In order for the user to unpause the timer.
DEP:	FR4

3.2.6 Functional Requirement 6

ID:	**FR6**
TITLE:	Mobile application – Start GPS tracking
DESC:	The GPS should start tracking the location of the user when the start button is pressed.
RAT:	To start tracking the location of the user.
DEP:	FR1

3.2.7 Functional Requirement 7

ID:	**FR7**
TITLE:	Mobile application – Stop GPS tracking
DESC:	The GPS should stop tracking the location of the user when the stop button is pressed.
RAT:	To stop tracking the location of the user.
DEP:	FR6

3.2.8 Functional Requirement 8

ID:	**FR8**
TITLE:	Mobile application – Pause GPS tracking
DESC:	The GPS should pause tracking the location of the user when the pause button is pressed.
RAT:	To pause tracking the location of the user.
DEP:	FR6

3.2.9 Functional Requirement 9

ID:	**FR9**
TITLE:	Mobile application – Unpause GPS tracking
DESC:	The GPS should unpause tracking the location of the user when the unpause button is pressed.
RAT:	To unpause tracking the location of the user.
DEP:	FR8

3.2.10 Functional Requirement 10

ID:	**FR10**
TITLE:	Mobile application – Draw route
DESC:	A route should start being drawn as the GPS periodically tracks the user’s location.
RAT:	To draw out the route taken by the user.
DEP:	FR6

3.2.11 Functional Requirement 11

ID:	**FR11**
TITLE:	Mobile application – Navigate to map page
DESC:	The user should be able to navigate to the map page by pressing the view map button.
RAT:	To navigate to the map page.
DEP:	FR1

3.2.12 Functional Requirement 12

ID:	**FR12**
TITLE:	Mobile application – Map display
DESC:	The user should be able to see the map on the page. The user should be able to see the route taken.
RAT:	To view map page.
DEP:	FR11, FR6

3.2.13 Functional Requirement 13

ID:	**FR13**
TITLE:	Mobile application – Navigate to profile page
DESC:	The user should be able to navigate to the profile page by pressing the profile button.
RAT:	To navigate to the profile page.
DEP:	FR1

3.2.14 Functional Requirement 14

ID:	**FR14**
TITLE:	Mobile application – Edit profile
DESC:	The user should be able to edit all of fields on the profile page. The fields include name, gender, age, weight, and height.
RAT:	In order for the user to edit the profile.
DEP:	FR13

3.2.15 Functional Requirement 15

ID:	**FR15**
TITLE:	Mobile application – Save profile
DESC:	The user should be able to save their profile.
RAT:	In order for the user to save the profile.
DEP:	FR13, FR14

3.2.16 Functional Requirement 16

ID:	**FR16**
TITLE:	Mobile application – Navigate to statistics page
DESC:	The user should be able to navigate to the statistics page by pressing the statistics button.
RAT:	To navigate to the statistics page.
DEP:	FR1

3.2.17 Functional Requirement 17

ID:	**FR17**
TITLE:	Mobile application – View statistics
DESC:	The user should be able to see all of the statistics on the page. These statistics include duration, distance, and calories burnt.
RAT:	To view statistics on statistics page.
DEP:	FR16

3.2.18 Functional Requirement 18

ID:	**FR18**
TITLE:	Mobile application – Navigate to scoreboard page
DESC:	The user should be able to navigate to the scoreboard page by pressing the scoreboard button.
RAT:	To navigate to the scoreboard page.
DEP:	FR1

3.2.19 Functional Requirement 19

ID:	**FR19**
TITLE:	Mobile application – View entries in scoreboard menu
DESC:	The user should be able to view all entries in the scoreboard menu.
RAT:	To view scoreboard entries.
DEP:	FR18

3.2.20 Functional Requirement 20

ID:	**FR20**
TITLE:	Mobile application – Select entries in scoreboard menu.
DESC:	The user should be able to select any entry in the scoreboard menu. Upon selecting an entry, the duration, distance and calories burnt statistics of said entry should be displayed.
RAT:	To select and view entries in scoreboard menu.
DEP:	FR18, FR19

3.3 Performance Requirements

This application is useful for the personal gain of someone who is attempting to improve their workout regime. By using this application, the user can keep track of their routes made while a workout is being done. One user can use the application at a time since the GPS will be used to track a single phone. The application functions through the tracking system updating a server every 5-10 seconds in order to keep track of the route being taken by the individual. The input by the individual takes as much time as it will take to input the information, but a number of the functions that the application possesses will execute in less than 1 second. One example of this is the amount of calories burned during the duration of the workout will be calculated once the finish button has been pressed. Since the amount of calories burned is evidently a simple algorithm based on the information the user inputs the calculation can be made with all the information once the application collects it.

The fastest GPS tracking unit functions at 10Hz, since our tracking equipment will be less advanced as those units, the tracking through GPS will be less than 10Hz. The most information that the application will have to process will be during the duration of the workout, whether the user is on a bicycle or on foot, since it will have to update every 5-10 seconds and store that information in a server database that will save to route the user's path. This information is then displayed on the user's phone screen, if desired, to track their destination. The user will have to be connected to the internet during the duration of their workout in order to ensure that the information captured by the GPS is stored and organized. Offline, the application will be able to display the best runs that have been documented, the past workouts that have been executed, and the progress that has been made by the user.

The dynamic workload of the application will only be present during the workout whereas the application will have a static workload when offline since the functions of the static application will be displaying or calculating. These functions are fundamental to most applications and require very little processing capacity

3.4 Logical Database Requirements

The use of a database is fundamental to the application being produced. The database being used will be from the SQLite database server that has been discussed during lecture and tutorial. By using this database, information will be able to be stored and displayed through the android application since the two programs are compatible.

The database will be accessed in multiple situations. Before the first workout is implemented, the database will be empty and will not be able to display any of the previous workouts since there have not been any. Once the user performs a workout that involves making distance, that distance and time taken will be stored in the database through a GPS tracking device.

The information that was stored, and to be stored, will then be available to the user to reflect on once multiple workouts have been produced. The use of a database creates the ease and precision for the user to keep track of their progress and reflect on their previous efforts. Once the workout duration, distance, and path are stored in the database, that information can be used later on for the user's convenience and for the application to perform any calculations or tasks that the user will find useful. Since there is a limited amount of information that can be stored, the user will be limited to 5 entries. The sixth entry to be processed by the application will replace the earliest entry and each entry afterwards will cause each of the other runs to be incremented forward while it replaces the first entry and so on. Integrity constraints are therefore present in the application which may be inconvenient for the user to lose the information that is unnecessary or outdated.

3.5 Design Constraints

Since the sensors are being provided to the class, the software being used is limited. There are other programs that are updated to today's convenience to ensure the quickest tracking through GPS, but this application is limited by the hardware and software provided.

3.5.1 Standards Compliance

Once the user downloads the application from the android app store, a specific requirement that must be met is the allowance of the application to access the location of the device being used to track the user. These traces are needed for the user to meet the regulatory terms of the GPS tracker being used. Without the permission of the location services from the phone, the application will not be able to process the route being tracked by the user and thus the application will be useless to the user.

3.6 Software System Attributes

In this section, the attributes of the software being used will be discussed. Android Studio has a number of functions that will be of great use when designing and creating this application. The application should be reliable, available, and portable.

3.6.1 Reliability

The reliability of the "Fitness Tracking Software" application is as reliable as it will get. Since the database is internal the only threat to reliability would be if the phone has a proper connection to the internet and has proper signal so that the GPS device embedded within the users mobile device is strong enough that it will not lose the signal. Since the app relies a lot on localization, the only threat to reliability would be signal loss for the gps.

3.6.2 Availability

This section refers to a defined availability level for the entire application. As mentioned previously, the application will be updated every 5-10 seconds with the route that is being tracked through the GPS system. Since the information that is created through the user while online will be stored in a database that is relatable to Android Studio, that information can be accessed by the user once offline. Once the system fails, the Android application will have to restart. This application will also be able to be used offline, using only the GPS, though less accurate, will be completely functionable by allowing location services on the device. This application was designed in such a way that it will not crash nor have any bugs upon releases to ensure reliability for the user.

3.6.3 Security

Since the FTS application utilizes and internal database with SQLite and an embedded piece of hardware such as the GPS within the mobile device there is no threat to security. The profile setting within the application contains no crucial information to the well being of the user, no credit card information and no address as well. Thus the need for security is at a pure minimum for this particular application.

3.6.4 Maintainability

Eventually the software of this application will be outdated by a new software and the application being produced will have to be updated. There are also a number of updates made by the hardware that may not be compatible with the software design of the application. These conflicts will disrupt the functionality of the application over time and the application software must be updated along with the hardware updates.

3.6.5 Portability

The portability of the FTS application at this point in time will only be available on Adroid devices, but for the future it can easily be ported over other app stores simply, since the effectiveness of this application and translation to other devices code wouldn't be too much of a hassle. For the time being, the client specifically asked for this software to be implemented only in Android.

Correctness, Efficiency, Maintainability, Reliability, Usability and Availability all rate High on the priority list for this application. In an effort to produce the best product possible, the development team for the FTS application have deemed those 6 characteristics as the most important going forward. Correctness for the clients vision for the app, efficiency for the effectiveness for the user, maintainability so that the app can stay viable even through updates to the hardware and software of the hosted mobile device. Reliability so that it can also be available when the user wants to use the app, Usability for the simpleness in usage and availability so that every user with an android device can use it.

For the medium priority we have Reusability and Testability, the client expressed that Reusability is important to a certain degree for the users, those who want to see improvement and want to get better will reuse this app with ease and not be fearful of a major change in interfaces. As for testability, the application will be tested to its full extent from the development team working on the FTS app and it will not be overlooked.

For the lower priority characteristics, the "Fitness Tracking Software" application will give the least attention to the Flexibility, Integrity/Security, Interoperability and Portability. Flexibility, because for the time being this application will only be used for users that are going to be doing a type of exercise involving movement in great distances (minimum 100 ft) in anyway they want (I.E: walking, jogging, roller blading, biking, etc.). Integrity/Security is low priority since the user will not be at risk from anything since the application will not require input of crucial information such as address, credit card number or anything of that nature. Interoperability and Portability because the client specifically asked for the application to be available on Android devices for the time being.

ID	**Characteristic**	**H/M/L**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	Correctness	H	▨											
2	Efficiency	H	▨	▨										
3	Flexibility	L	▨	▨	▨									
4	Integrity/Security	L	▨	▨	▨	▨								
5	Interoperability	L	▨	▨	▨	▨	▨							
6	Maintainability	H	▨	▨	▨	▨	▨	▨						
7	Portability	L	▨	▨	▨	▨	▨	▨	▨					
8	Reliability	H	▨	▨	▨	▨	▨	▨	▨	▨				
9	Reusability	M	▨	▨	▨	▨	▨	▨	▨	▨	▨			
10	Testability	M	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
11	Usability	H	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	
12	Availability	H	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Table 2: Portability Priorities

4. Change Management Process

In every project lifespan there are changes that must be made due to client preference or problem occurrences. The outcome of the success of the project is dependent on the adaptiveness of the team members. Regulations of the rate of change are in order to protect the development team from disappointing the client and to have strict guidelines that satisfy both the team and the client. In the event of a change of preference from the client, that change will be analyzed by the team and placed into the timeline of the development of the application when it is most convenient for the development team. The product owner will ultimately be in charge of this change in schedule, since they are the ones who have an overall plan of what each team member is to achieve in the upcoming weeks. Once a change is requested by whomever, that change will be analyzed by the team and each member of the team will be allowed an input suggestion as well as a chance to discuss time conflicts in order to maximize efficiency of the team. After each team member contributes their problems, the product owner will decide when the change should occur and who should provide the services of making that change. These changes can be submitted regardless of formality in hopes to satisfy the client, although in return the client should be respectful of the restrictions of a team of university students.

When a change is requested, it will be sent through the initial schedule of the upcoming weeks. The team will then rank the priority of this change in terms of the production of the application. If the change conflicts the outcome of the application then the priority will be higher than if the change is in style or schematic of the layout of the application. Communication is important in any team, therefore each change will be assessed by each team member of the development team in order to satisfy the workers of the product.

In the event that the change is unable to be produced, a new decision will be discussed with the client by the product owner. This is the worst case scenario that could occur due to time conflicts or unrealistic expectations by the client and will be attempted to be avoided at all costs. The new decision that is to be negotiated will either be for the change to occur at a later time, or a different change that can be brought out that also satisfies the client’s desire.

5. Document Approvals

The approval of this document is to be represented through a signature in the table below. Each team member must approve of the written document to ensure that a clear understanding of the task and product to be done is had by each member. It is also of fundamental importance that the client signs the SRS in order to ensure understanding of the project and the highlights and guidelines of the outcome.

Table 3: Table of Document Approval

Name	**Date**	**ID**	**Signature**
Michael Gravino *Expert Documenter & Programmer Aid*	**05/12/16**	**27088140**	Michael Gravino
Matthew Masi *Main Expert Programmer & Head of IT*	**05/12/16**	**27039956**	Matthew Masi
Rabih El-Bitar *Expert Documenter & Chief Communicator*	**05/12/16**	**27013647**	Rabih El-Bitar
Lucia Racine *Expert Documenter & Head Hardware Consultant*	**05/12/16**	**26462111**	Lucia Racine
Carmine Masi *Client of Dream Team*	**05/12/16**	**-------------**	Carmine Masi

*All have been signed electronically with the authorisation of every participant.

II. DESIGN DOCUMENT

6. Introduction

A sprint is a portion of the Scrum process that dictates the tasks and events that will occur in the upcoming weeks. In the most recent sprint, the development team and product owner achieved great lengths in the application being created.

The application has a scope of fitness enthusiasts interested in health and tracking their progress. This application has implemented many features through the last 6 weeks, the following has been implement. The home page has been created with two buttons, both activated through an on-click listener. One button transports you to a profile page where the user can input information; the information contained includes height, weight, age, and gender then by using this information given by the user, the program then calculates the amount of calories burned during the users jog after completion using the details provided through the profile page and the time and distance of run. The user can now see their statistics within the application such as calories burned (calculated from the data given in the Profile page set up by the user) as well as distance traveled in meters, duration in minutes: seconds: milliseconds and average speed. The user can also now save there runs within the maps activity to keep track of his or her best runs, they can save up to 5 items and it will automatically delete the oldest one when the user supersedes 5 saves. Also, the add of the foregrounding service gives more versatility for the user, they can now go in and out of the application to use other application and still maintain functionality. Trophies have also been added to add an extra motivating factor so that users can achieve their goals with a little more pleasure and self gratification.

As the final submission of the FTS application has come, every feature set out and every requirement given from the client has been achieved. All items on the product backlog with the highest priority have been implemented and completed. The information below will confirm all added features from the beginnng of Sprint 1 up until the end sprint 3.

7. Architecture

7.1 Overall Macroscopic System Structure

The application in progress uses multiple unattached modules. The FTS uses a SQLite Database to store and retrieve the information collected from the user tracking their route. It also has an API in order to contact the GPS tracking device in a phone to access its functionality. The "Fitness Tracking Software" application does not classify each method into a separate module, instead most methods are written under public classes. Since multiple variables that are input by the user are used by the program to calculate other variables, these elements must be shared between methods and activities. The FTS application contains 11 classes, MapsActivity, ProfileActivity, Trophy, ForegroundService, ReportActivity, TrophiesUnlocked, TrophyActivity, RunStats, DBHandler, DistanceCalculator and a Timer.

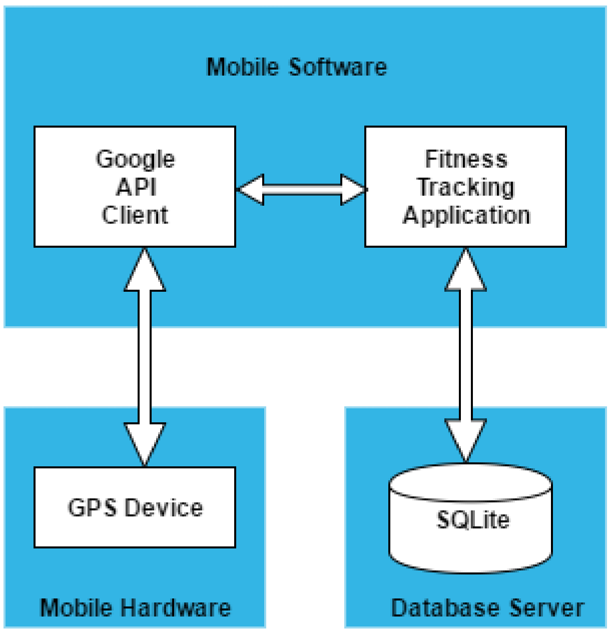


Figure 2: Abstract Diagram

7.2 Set Of Design Decisions

The FTS application was first designed in the vision of the client. Since this application is stand-alone it is important to note the simplicity in the design in order to maximize usability, performance and maintainability. In order to increase usability, the application will suffer in flexibility since the main focus will be on design strategy and interface optimization versus running the application on different platforms or devices.

Rabih, Michael and Lucia wanted the FTS application to feature a sensor that can detect the heart rate if the user plans to use a third party sensor that can be plugged into his device while running. Due to the timeline and necessity of said feature, the development team decided against it and place it on a lower priority has some of the main features.

The “Home Page” was discussed in detail on which buttons and how many buttons the team wanted to feature, after a long discussion we agreed upon a set number of between 3-5 buttons maximum, since we don’t want to clutter the application and don’t want to confuse any users.

As a team, we had to come up with a compromise when it came to the UI, deciding on colours and pictures for features was also an important issue. Through discussions at different meetings we came up with a consensus that pleased all of us and the client.

7.3 Difficult Changes

At this point in time, there are few details that would be difficult to change. The “Fitness Tracking Software” application has made tremendous progress during its creation, but most of the functions implemented into the program are fundamental to the essence of the application. Similarly, the functions that have been created will not be changed throughout the production. The Google API is directly connected through the utilization of the GPS so it will be difficult to change the way the FTS utilizes the maps within the app.

One difficult change was the adding of the trophy feature, in order for it to function as mentioned we had to input a lot of effort to get it working property to the end of the sprint goal, which was a success.

7.4 Software Structure

As of the current sprint, this MVC (Model, View, Controller) diagram will demonstrate the appropriate design and will show the main parts with their relation and interaction with each other. In the View category, all activities will be present, this is what the user sees, utilizes for input and receives outputs from it. The controller is what is used to calculate or output to the activities. The model is the tools utilized by the controller to properly calculate the outputs for the view portion of the diagram.

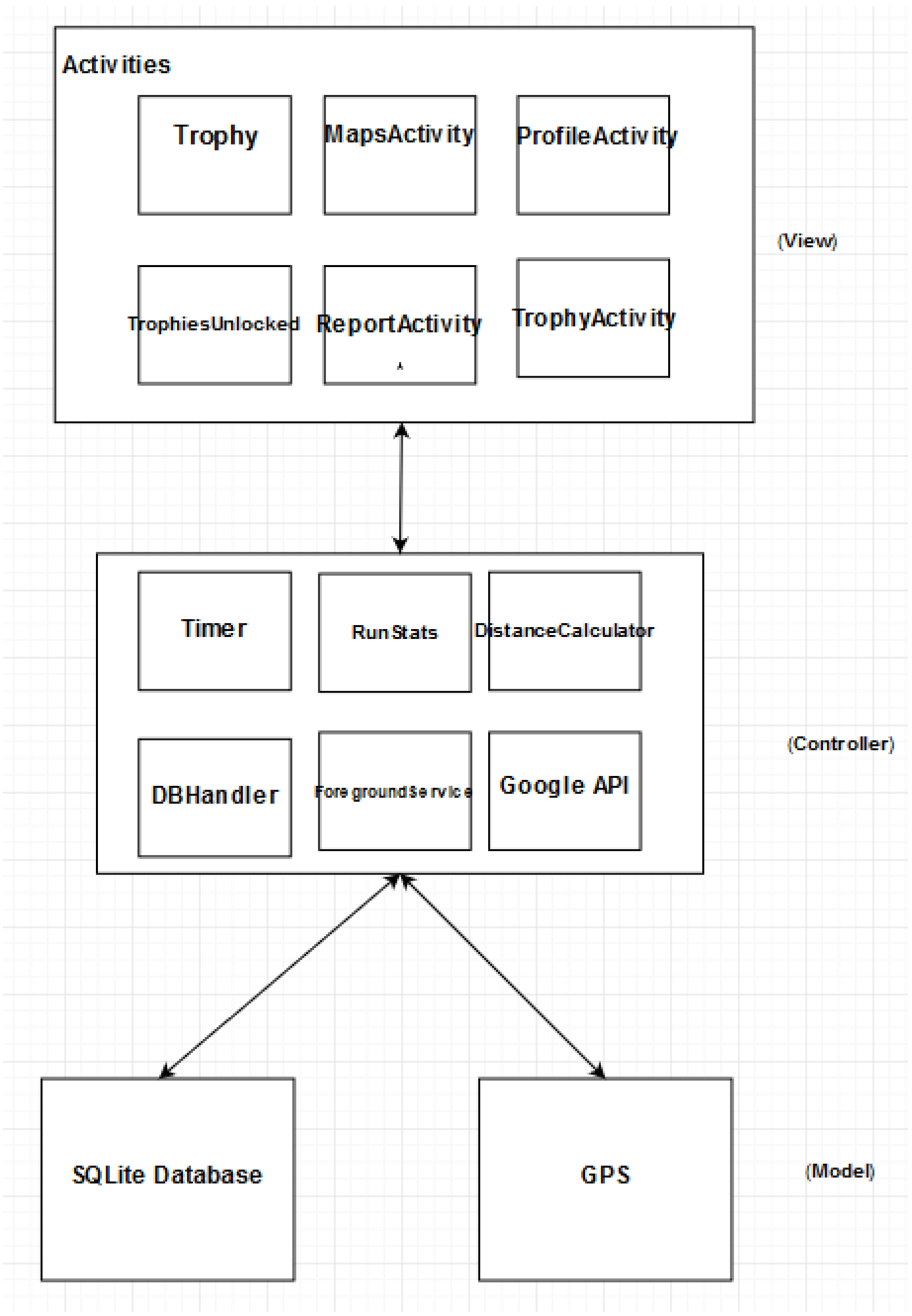


Figure 3: MVC for FTS

As can be seen from figure 3, the View portion of the MVC diagram has all classes (activities in android's case), where as the controller part of this diagram will feature the Google API that will give the FTS application the ability to utilize the embedded GPS within the mobile device as well as the maps function to display the route the user takes. It will also help by giving the current location and display it for the user as they go along there run to be mapped. The model portion is simply the SQLite database being used internally within Android as well as its GPS hardware. Anything that will handle the data from the Activities section or (View) will also be in the controller section such as DBHandler, Timer, DistanceCalculator, RunStats and ForegroundService.

7.5 Communication Interfaces

This system does not communicate with any other systems in the outside world. The FTS application is stand-alone and will utilize an internal database as well as an internal sensor within the mobile device to localize the user.

7.6 APIs

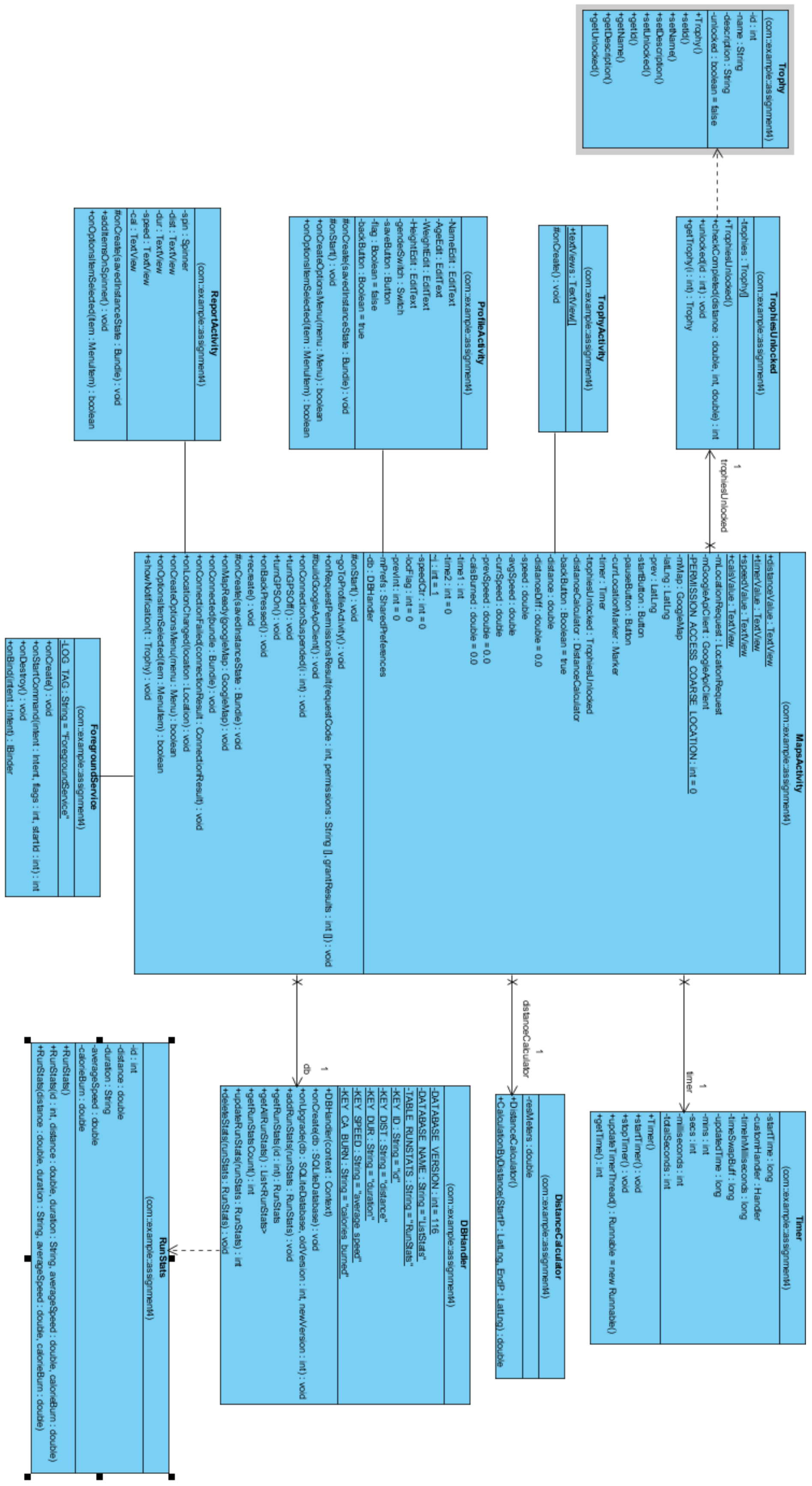
The application has a feature that tracks the route of a user while they are traveling during their workout. In order to map out the user's route and display the markers onto it the application utilizes the Google Maps API. After opening a Google Maps Project in Android Studio, a new activity is created that allows the programmer to create an API key. Once this step is finished, the map can then be manipulated into doing what the development team chooses, in this case to track the user and record the results. An xml file will appear once the API key is created, in this xml file the development team input a series of commands in order to achieve routing the user. Since this was the most extensive and intricate part in the process of designing the application, this step will remain unchanged.

The Google Maps API allows communication with Google Services, more specifically the Google Maps service. Since the "Fitness Tracking Software" application is a tracking software that will display periodically every 5-10 seconds and every 5-10 meters of the user's movements, there is no easy way around utilizing the benefits of this API for this application.

8. Static Model

For the “Fitness Tracking Software” application it is important to note that there are fundamental classes and activities that contribute to the foundation of the application which contain exactly 11 classes. These 11 classes are spread out into 3 modules, the data module, the map module and the trophy module. Within the trophy module, the application will have the Trophy class and TrophiesUnlocked within it. We implemented a database, the database utilizes objects from the RunStats class. The MapsActivity class is the parent class to all other classes and contains the ProfileActivity which is the “home page” of the application and every other module and class will be derived from there. The Data module has the ReportActivity and RunStats class and the map module has the Timer and DistanceCalculator class. The ReportActivity directly communicates with the DBHandler and the MainActivity. The DBHandler in this case is a controller that will control the data from the activities such as the MapsActivity, RunStats and ReportActivity like just mentioned. It will handle this data and will relay it to both the view and model portion of the application. The RunStats class will calculate all the statistic the FTS features such as distance, duration, calories burned and average speed. It will relay that information to the DBHandler. Last but not least is the DistanceCalculator class which will calculator the distance travelled from the map and the locally embedded GPS device within the mobile device of the user. With the foreground service, the user can now exit out of the application while it is running and use other applications without disrupting the FTS app. There is also a notification that will appear once the user unlocks a certain trophy, this can be seen by looking at the android overlay as well as within the Fitness Tracking Software application. All of this will be better understood by observing the following UML class diagram in figure 4:

Figure 4: UML Class Diagram for FTS



Trophy
(com::example::assignment4)
-id : int
-name : String
-description : String
-unlocked : boolean = false
+Trophy()
+setId()
+setName()
+setDescription()
+setUnlocked()
+getId()
+getName()
+getDescription()
+getUnlocked()
TrophiesUnlocked
(com::example::assignment4)
-trophies : Trophy[]
+TrophiesUnlocked()
+checkCompleted(distance : double, int, double) : int
+unlocked(id : int) : void
+getTrophy(i : int) : Trophy
TrophyActivity
(com::example::assignment4)
+textViews : TextView[]
#onCreate() : void
ProfileActivity
(com::example::assignment4)
-NameEdit : EditText
-AgeEdit : EditText
-WeightEdit : EditText
-HeightEdit : EditText
-genderSwitch : Switch
-saveButton : Button
-flag : Boolean = false
-backButton : Boolean = true
#onCreate(savedInstanceState : Bundle) : void
#onStart() : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onOptionsItemSelected(item : MenuItem) : boolean
ReportActivity
(com::example::assignment4)
-spin : Spinner
-dist : TextView
-dur : TextView
-speed : TextView
-cal : TextView
#onCreate(savedInstanceState : Bundle) : void
+addItemsOnSpinner() : void
+onOptionsItemSelected(item : MenuItem) : boolean
MapsActivity
(com::example::assignment4)
+distanceValue : TextView
+timerValue : TextView
+speedValue : TextView
+calsValue : TextView
-mLocationRequest : LocationRequest
-mGoogleApiClient : GoogleApiClient
-PERMISSION_ACCESS_COARSE_LOCATION : int = 0
-mMap : GoogleMap
-latLng : LatLng
-prev : LatLng
-startButton : Button
-pauseButton : Button
-currLocationMarker : Marker
-timer : Timer
-trophiesUnlocked : TrophiesUnlocked
-distanceCalculator : DistanceCalculator
-backButton : Boolean = true
-distance : double
-distanceDiff : double = 0.0
-speed : double
-avgSpeed : double
-currSpeed : double
-prevSpeed : double = 0.0
-calsBurned : double = 0.0
-time1 : int
-time2 : int = 0
-i : int = 1
-speedCtr : int = 0
-locFlag : int = 0
-prevInt : int = 0
-mPrefs : SharedPreferences
-db : DBHandler
#onStart() : void
-goToProfileActivity() : void
+onRequestPermissionsResult(requestCode : int, permissions : String [], grantResults : int []) : void
#buildGoogleApiClient() : void
+onConnectionSuspended(i : int) : void
+turnGPSOff() : void
+turnGPSOn() : void
+onBackPressed() : void
+recreate() : void
#onCreate(savedInstanceState : Bundle) : void
+onMapReady(googleMap : GoogleMap) : void
+onConnected(bundle : Bundle) : void
+onConnectionFailed(connectionResult : ConnectionResult) : void
+onLocationChanged(location : Location) : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onOptionsItemSelected(item : MenuItem) : boolean
+showNotification(t : Trophy) : void
ForegroundService
(com::example::assignment4)
-LOG_TAG : String = "ForegroundService"
+onCreate() : void
+onStartCommand(intent : Intent, flags : int, startId : int) : int
+onDestroy() : void
+onBind(intent : Intent) : IBinder
trophiesUnlocked
1
timer
1
distanceCalculator
1
db
1
Timer
(com::example::assignment4)
-startTime : long
-customHandler : Handler
-timeInMilliseconds : long
-timeSwapBuff : long
-updatedTime : long
-mins : int
-secs : int
-milliseconds : int
-totalSeconds : int
+Timer()
+startTimer() : void
+stopTimer() : void
+updateTimerThread() : Runnable = new Runnable()
+getTime() : int
DistanceCalculator
(com::example::assignment4)
-resMeters : double
+DistanceCalculator()
+CalculationByDistance(StartP : LatLng, EndP : LatLng) : double
DBHandler
(com::example::assignment4)
-DATABASE_VERSION : int = 116
-DATABASE_NAME : String = "ListStats"
-TABLE_RUNSTATS : String = "RunStats"
-KEY_ID : String = "id"
-KEY_DIST : String = "distance"
-KEY_DUR : String = "duration"
-KEY_SPEED : String = "average_speed"
-KEY_CA_BURN : String = "calories_burned"
+DBHandler(context : Context)
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
+addRunStats(runStats : RunStats) : void
+getRunStats(id : int) : RunStats
+getAllRunStats() : List<RunStats>
+getRunStatsCount() : int
+updateRunStats(runStats : RunStats) : int
+deleteStats(runStats : RunStats) : void
RunStats
(com::example::assignment4)
-id : int
-distance : double
-duration : String
-averageSpeed : double
-calorieBurn : double
+RunStats()
+RunStats(id : int, distance : double, duration : String, averageSpeed : double, calorieBurn : double)
+RunStats(distance : double, duration : String, averageSpeed : double, calorieBurn : double)

9. Dynamic Model

9.1 Sequence Diagrams

These sequence diagrams depict the user action and the sequence of method calls. In this subsection, we will depict most or all of the sequence diagrams from the classes of the Fitness Tracking Software application, they are separated in 7 classes; MapsActivity, ProfileActivity, Timer, TrophiesUnlocked, ForegroundService, ReportActivity and DistanceCalculator.

MapsActivity:

Figure 5: SD for MapsActivity.buildGoogleApiClient()

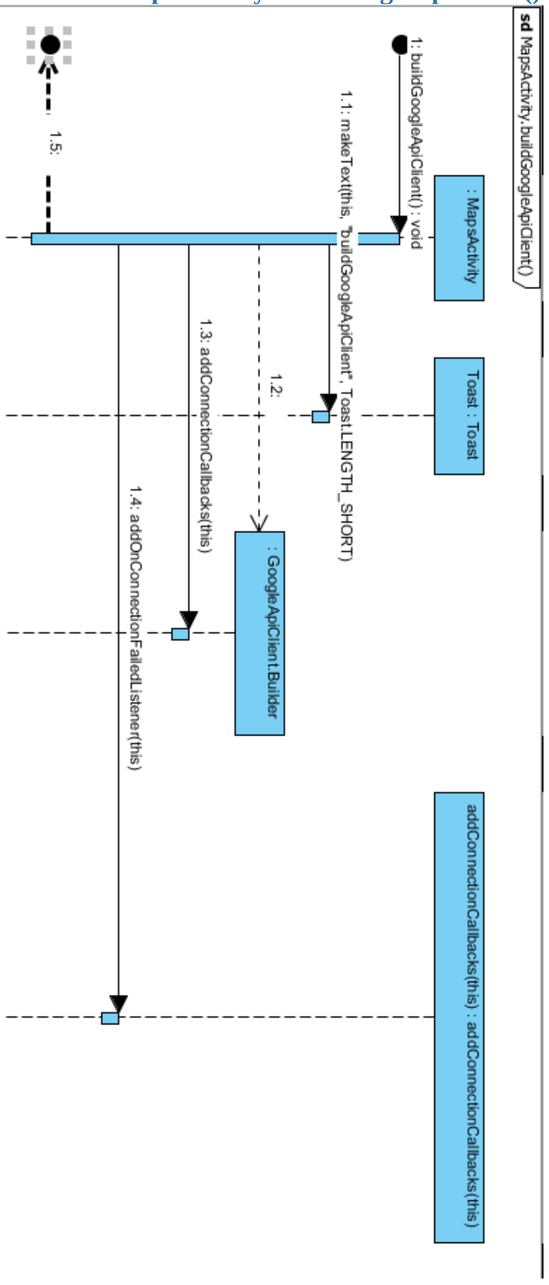

sd MapsActivity.buildGoogleApiClient()
: MapsActivity
Toast : Toast
: GoogleApiClient.Builder
addConnectionCallbacks(this) : addConnectionCallbacks(this)
1: buildGoogleApiClient() : void
1.1: makeText(this, "buildGoogleApiClient", Toast.LENGTH_SHORT)
1.2:
1.3: addConnectionCallbacks(this)
1.4: addOnConnectionFailedListener(this)
1.5:

Figure 6: SD for MapsActivity.onConnected(Bundle)

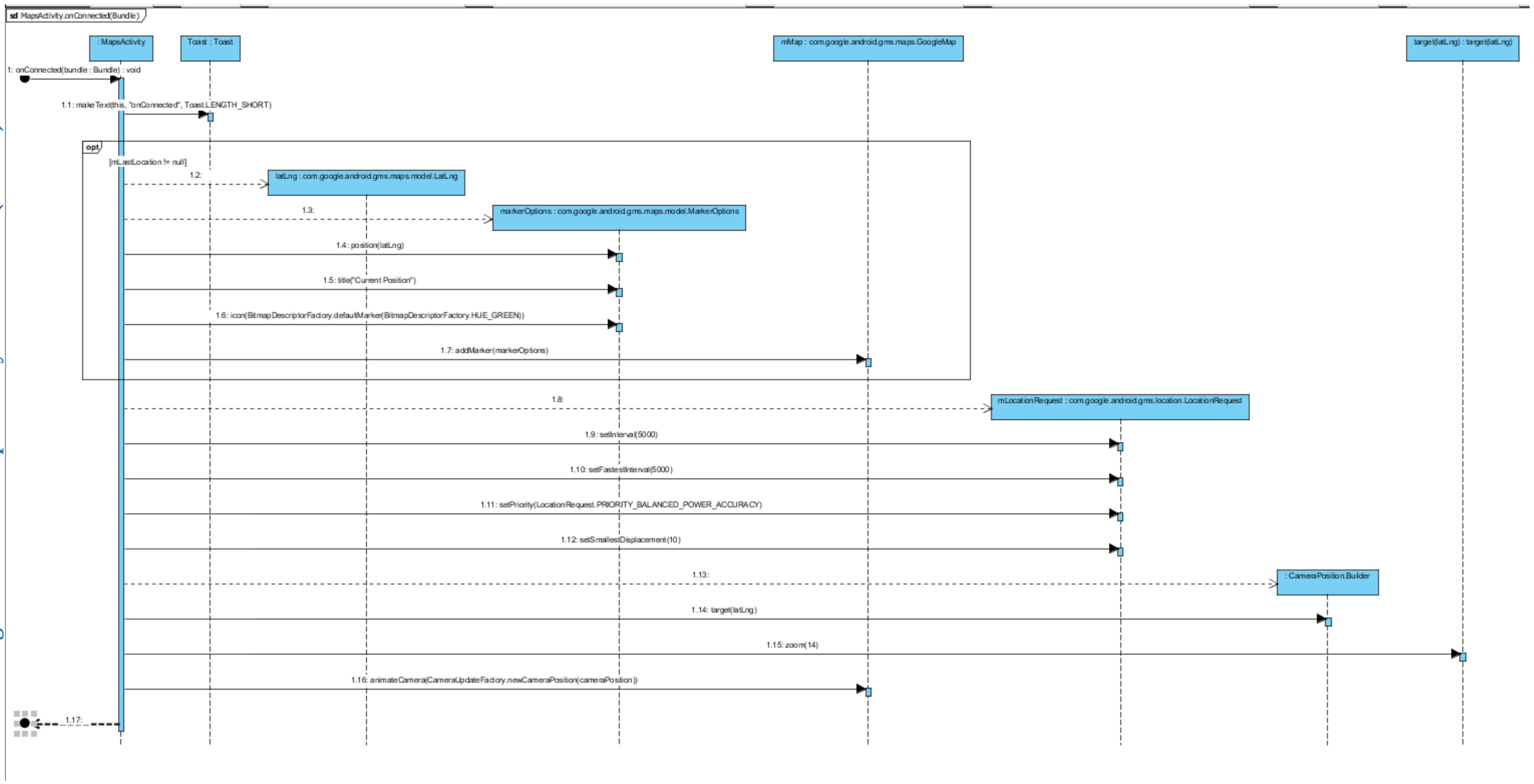
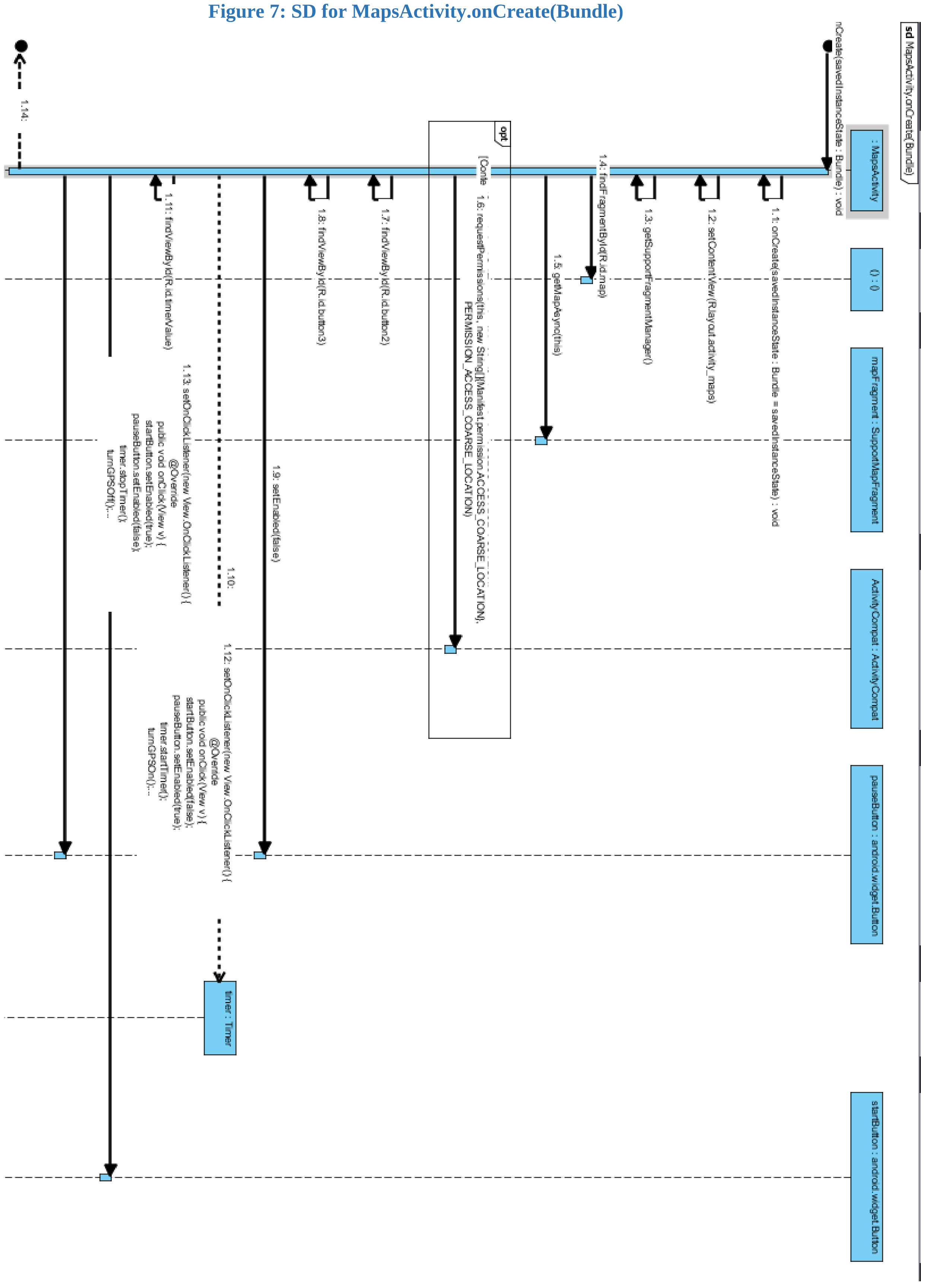

sd MapsActivity.onConnected(Bundle)
: MapsActivity
Toast : Toast
mMap : com.google.android.gms.maps.GoogleMap
target(latLng) : target(latLng)
1: onConnected(bundle : Bundle) : void
1.1: makeText(this, "onConnected", Toast.LENGTH_SHORT)
opt
[mLastLocation != null]
1.2:
latLng : com.google.android.gms.maps.model.LatLng
1.3:
markerOptions : com.google.android.gms.maps.model.MarkerOptions
1.4: position(latLng)
1.5: title("Current Position")
1.6: icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN))
1.7: addMarker(markerOptions)
1.8:
mLocationRequest : com.google.android.gms.location.LocationRequest
1.9: setInterval(5000)
1.10: setFastestInterval(5000)
1.11: setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY)
1.12: setSmallestDisplacement(10)
1.13:
: CameraPosition.Builder
1.14: target(latLng)
1.15: zoom(14)
1.16: animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition))
1.17:

Figure 7: SD for MapsActivity.onCreate(Bundle)



sd MapsActivity.onCreate(Bundle)
: MapsActivity
() : ()
mapFragment : SupportMapFragment
ActivityCompat : ActivityCompat
pauseButton : android.widget.Button
startButton : android.widget.Button
timer : Timer
onCreate(savedInstanceState : Bundle) : void
1.1: onCreate(savedInstanceState : Bundle = savedInstanceState) : void
1.2: setContentView(R.layout.activity_maps)
1.3: getSupportFragmentManager()
1.4: findFragmentById(R.id.map)
1.5: getMapAsync(this)
opt
[Conte
1.6: requestPermissions(this, new String[]{Manifest.permission.ACCESS_COARSE_LOCATION}, PERMISSION_ACCESS_COARSE_LOCATION)
1.7: findViewById(R.id.button2)
1.8: findViewById(R.id.button3)
1.9: setEnabled(false)
1.10:
1.11: findViewById(R.id.timeValue)
1.12: setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { startButton.setEnabled(false); pauseButton.setEnabled(true); timer.startTimer(); turnGPSOn();...
1.13: setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { startButton.setEnabled(true); pauseButton.setEnabled(false); timer.stopTimer(); turnGPSOff();...
1.14:

Figure 8: SD for MapsActivity.onLocationChanged(Location)

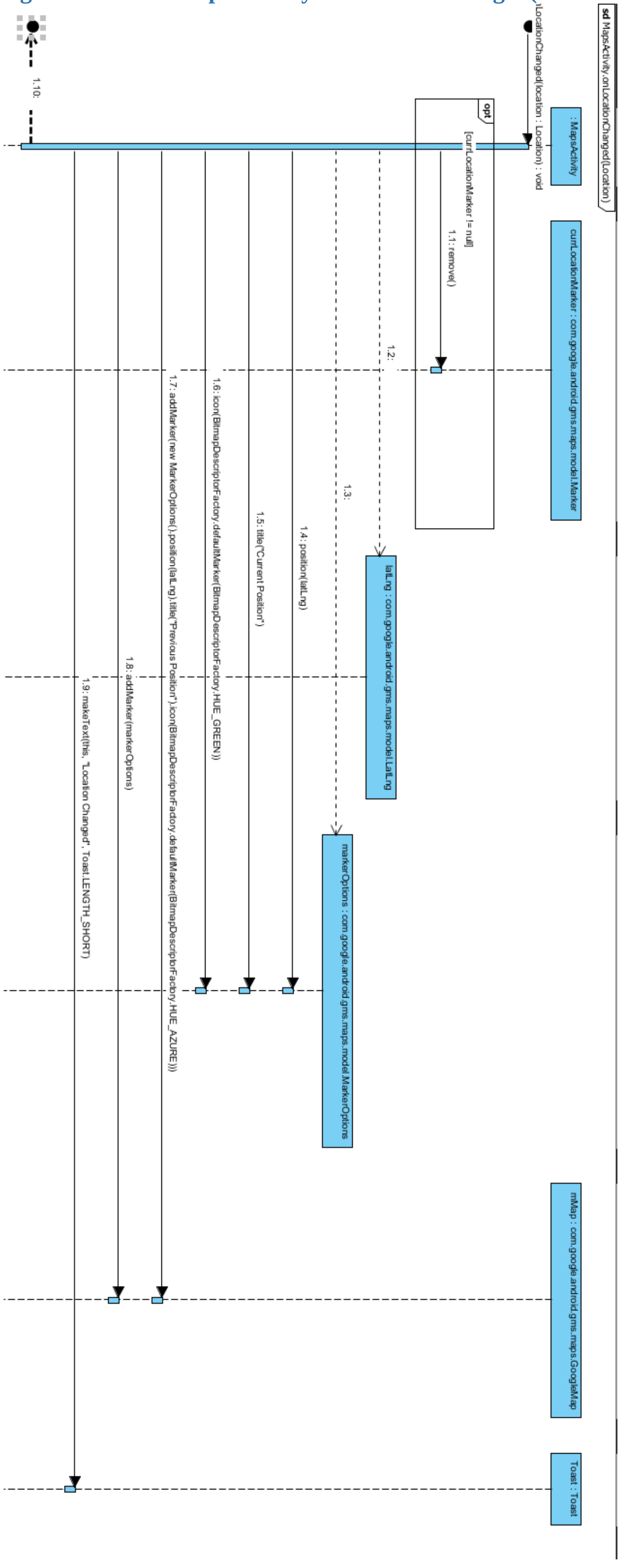

sd MapsActivity.onLocationChanged(Location)
: MapsActivity
curLocationMarker : com.google.android.gms.maps.model.Marker
latLng : com.google.android.gms.maps.model.LatLng
markerOptions : com.google.android.gms.maps.model.MarkerOptions
mMap : com.google.android.gms.maps.GoogleMap
Toast : Toast
onLocationChanged(location : Location) : void
opt
[curLocationMarker != null]
1.1: remove()
1.2:
1.3:
1.4: position(latLng)
1.5: title("Current Position")
1.6: icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN))
1.7: addMarker(new MarkerOptions().position(latLng).title("Previous Position").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)))
1.8: addMarker(markerOptions)
1.9: makeText(this, "Location Changed", Toast.LENGTH_SHORT)
1.10:

Figure 9: SD for MapsActivity.onMapReady(GoogleMap)

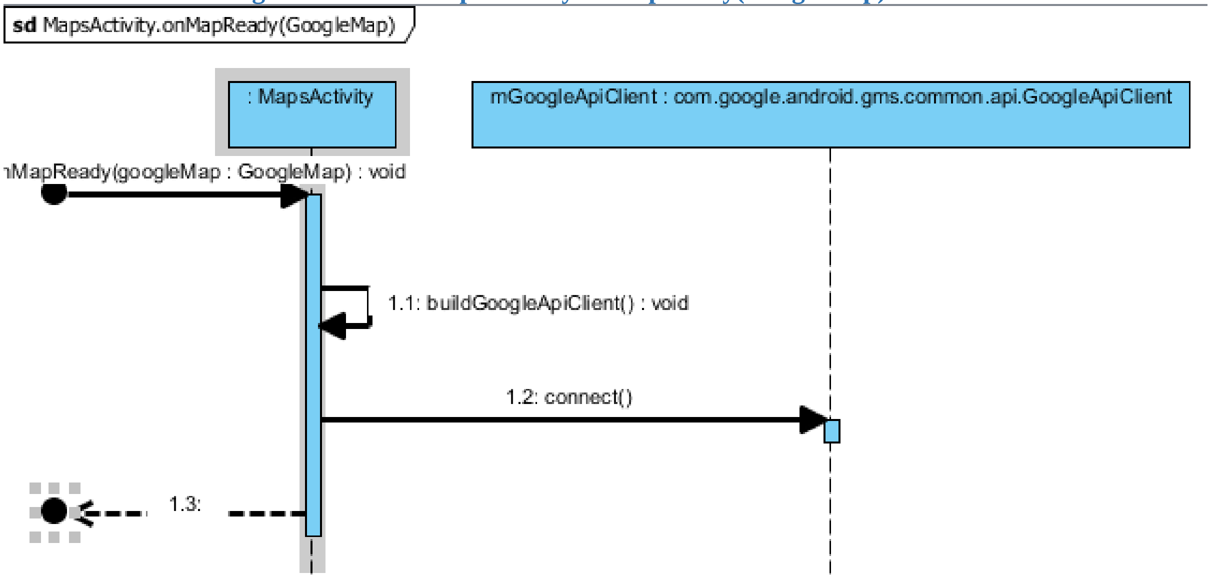

sd MapsActivity.onMapReady(GoogleMap)
: MapsActivity
mGoogleApiClient : com.google.android.gms.common.api.GoogleApiClient
ıMapReady(googleMap : GoogleMap) : void
1.1: buildGoogleApiClient() : void
1.2: connect()
1.3:

Figure 10: SD for Maps.Activity.onRequestPermissionsResult(int,String[],int[])

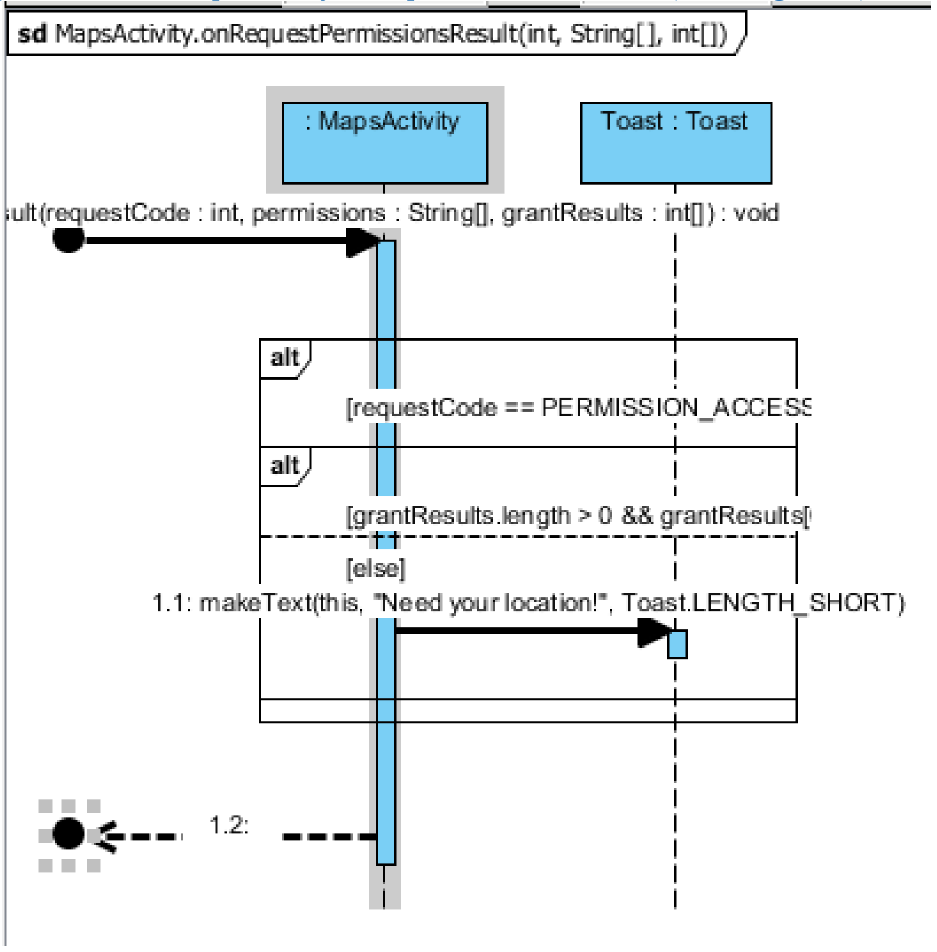

sd MapsActivity.onRequestPermissionsResult(int, String[], int[])
: MapsActivity
Toast : Toast
ult(requestCode : int, permissions : String[], grantResults : int[]) : void
alt
[requestCode == PERMISSION_ACCESS
alt
[grantResults.length > 0 && grantResults[
[else]
1.1: makeText(this, "Need your location!", Toast.LENGTH_SHORT)
1.2:

Figure 11: SD for MapsActivity.showNotification(Trophy)

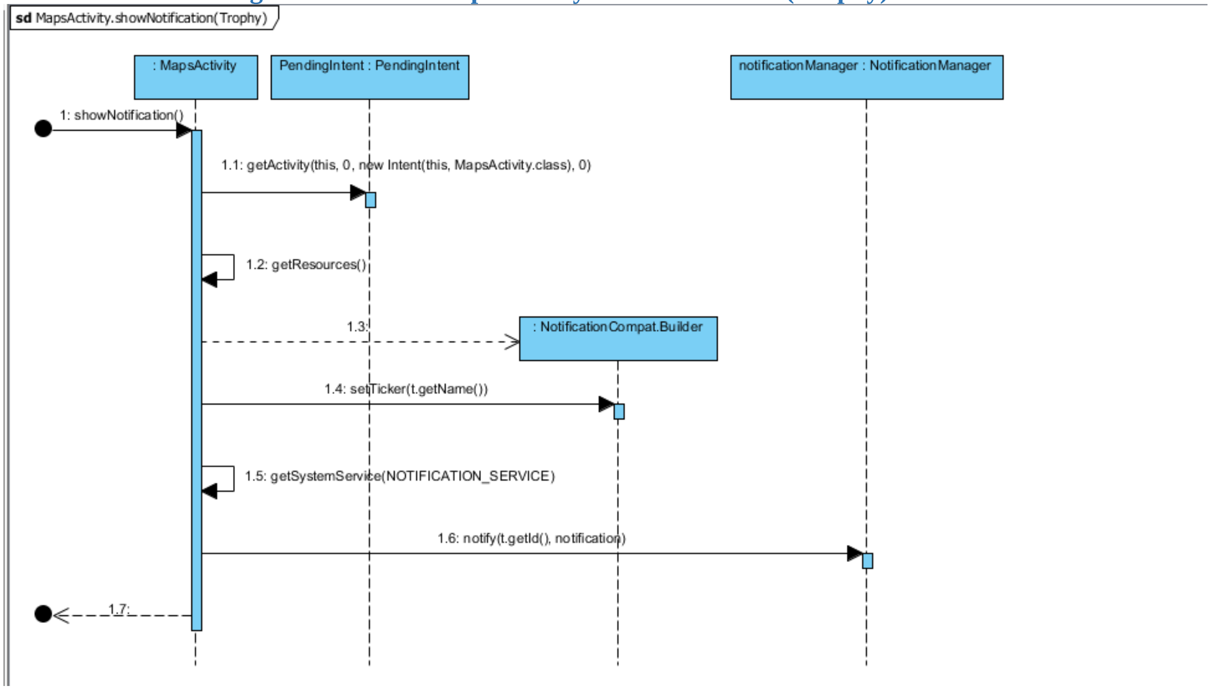

sd MapsActivity.showNotification(Trophy)
: MapsActivity
PendingIntent : PendingIntent
notificationManager : NotificationManager
1: showNotification()
1.1: getActivity(this, 0, new Intent(this, MapsActivity.class), 0)
1.2: getResources()
1.3:
: NotificationCompat.Builder
1.4: setTicker(t.getName())
1.5: getSystemService(NOTIFICATION_SERVICE)
1.6: notify(t.getId(), notification)
1.7:

Figure 12: SD for MapsActivity.turnGPSOff()

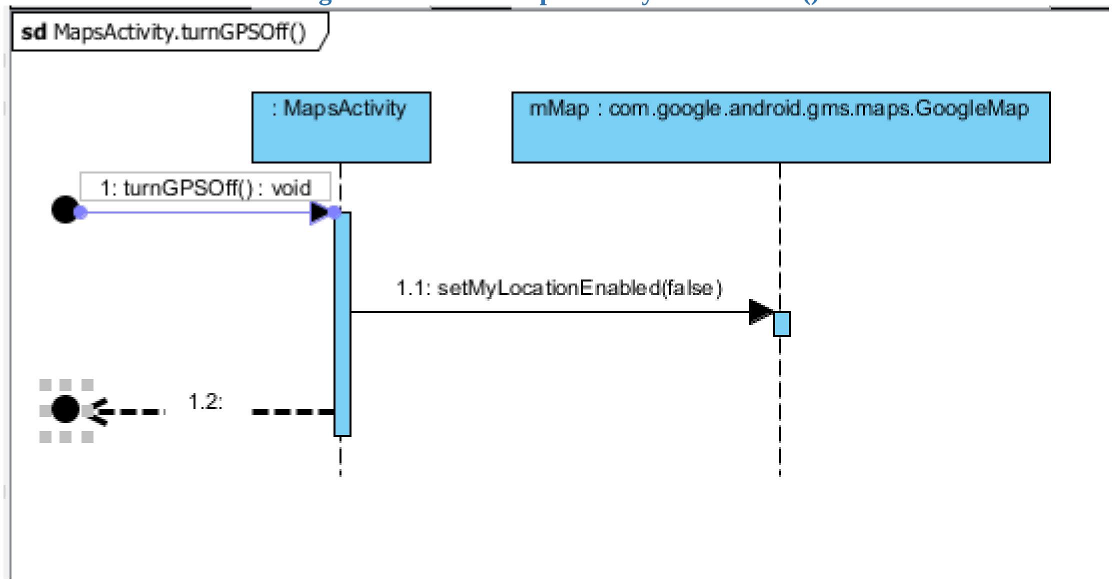
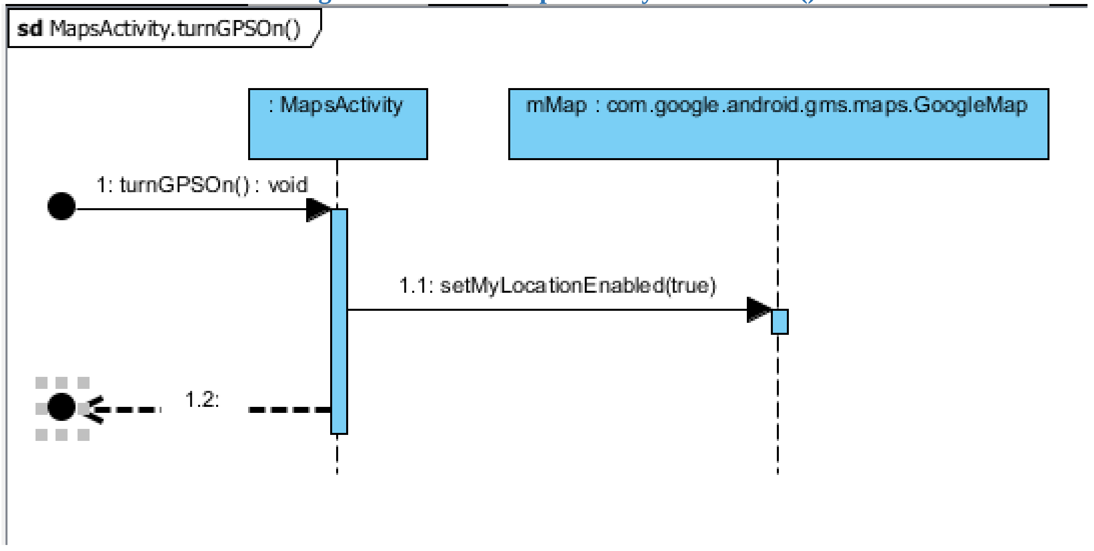

sd MapsActivity.turnGPSOff()
: MapsActivity
mMap : com.google.android.gms.maps.GoogleMap
1: turnGPSOff() : void
1.1: setMyLocationEnabled(false)
1.2:

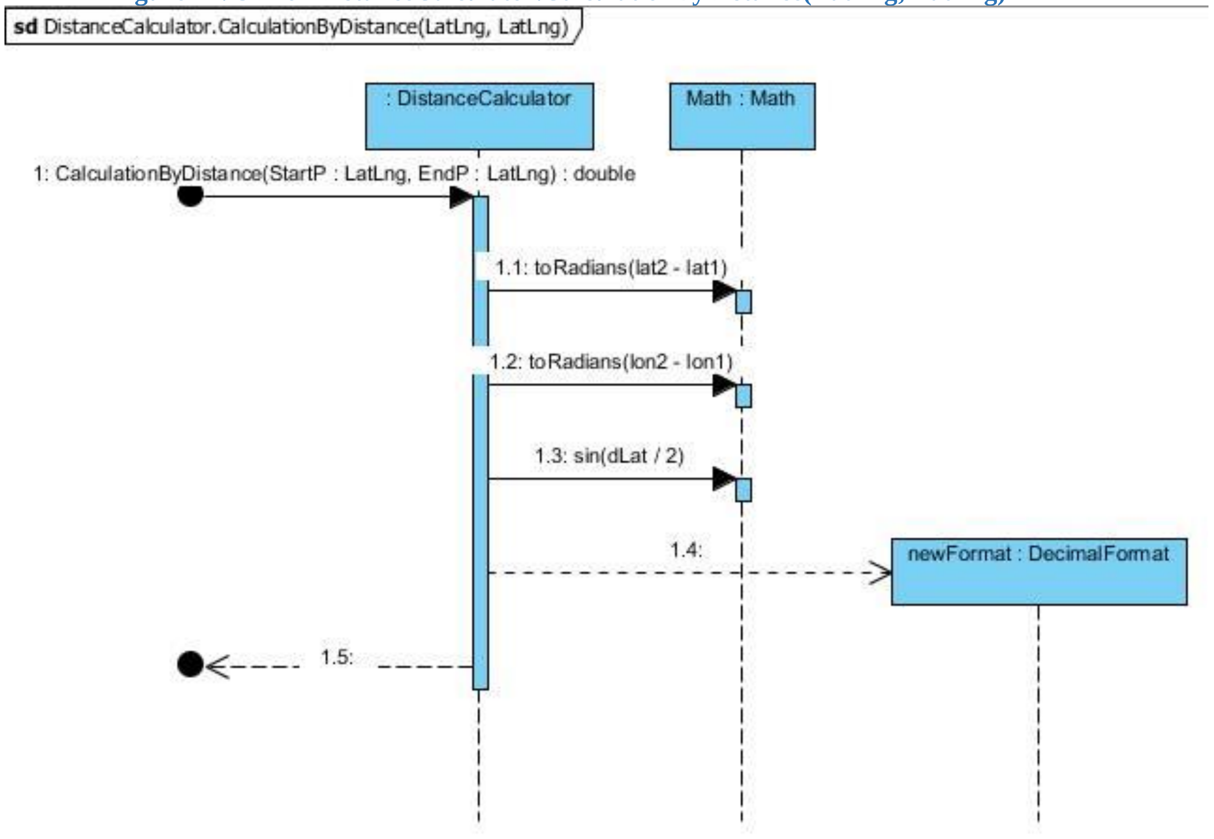
Figure 13: SD for MapsActivity.turnGPSOn()



sd MapsActivity.turnGPSOn()
: MapsActivity
mMap : com.google.android.gms.maps.GoogleMap
1: turnGPSOn() : void
1.1: setMyLocationEnabled(true)
1.2:

Distance Calculator:

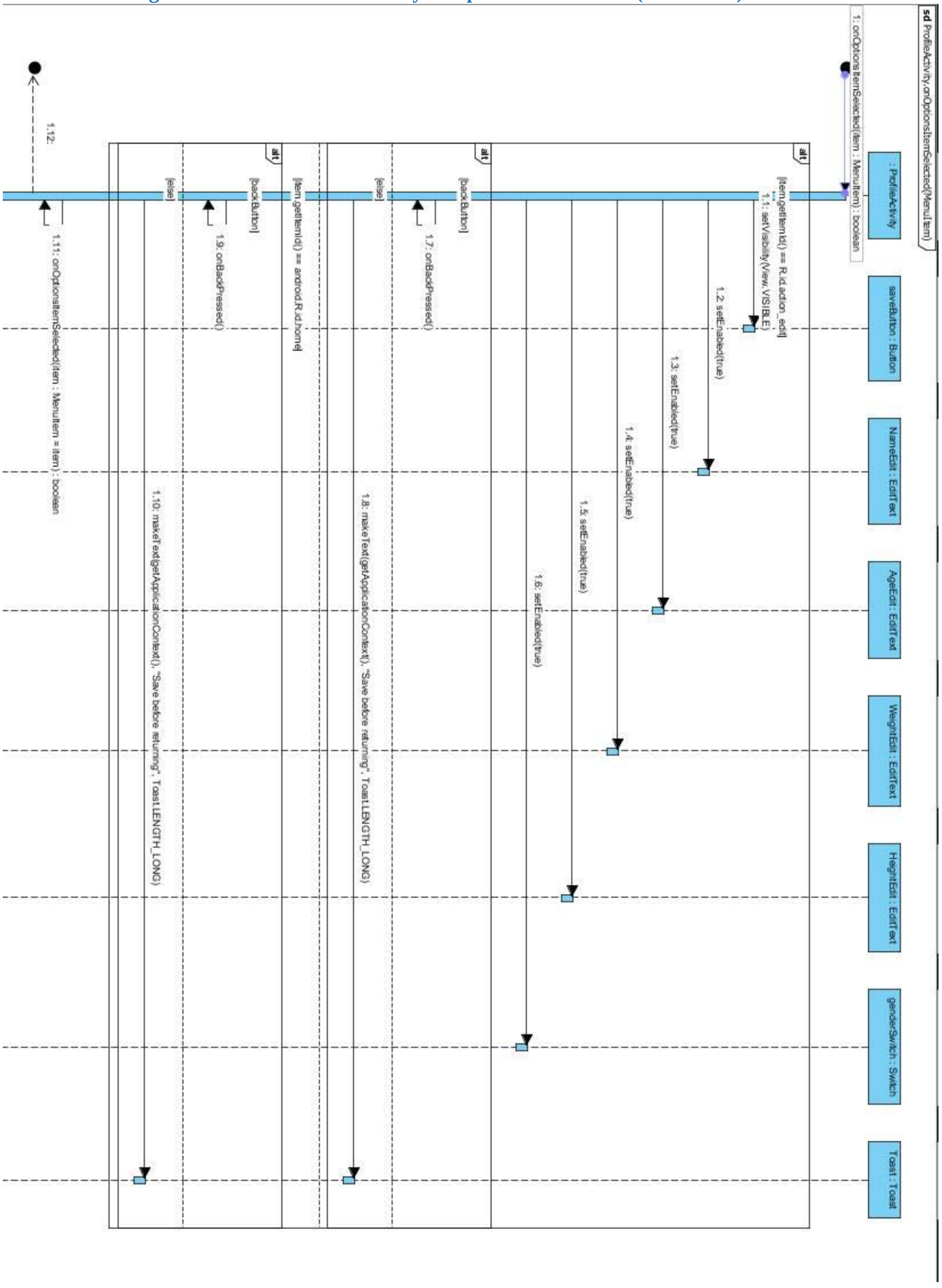
Figure 14: SD for DistanceCalculator.CalculationByDistance(LatLng, LatLng)



sd DistanceCalculator.CalculationByDistance(LatLng, LatLng)
: DistanceCalculator
Math : Math
1: CalculationByDistance(StartP : LatLng, EndP : LatLng) : double
1.1: toRadians(lat2 - lat1)
1.2: toRadians(lon2 - lon1)
1.3: sin(dLat / 2)
1.4:
newFormat : DecimalFormat
1.5:

ProfileActivity:

Figure 15: SD for ProfileActivity.onOptionsItemSelected(MenuItem)


sd ProfileActivity.onOptionsItemSelected(MenuItem)
1: onOptionsItemSelected(item : MenuItem) : boolean
: ProfileActivity
saveButton : Button
NameEdit : EditText
AgeEdit : EditText
WeightEdit : EditText
HeightEdit : EditText
genderSwitch : Switch
Toast : Toast
alt
[item.getItemId() == R.id.action_edit]
1.1: setVisibility(View.VISIBLE)
1.2: setEnabled(true)
1.3: setEnabled(true)
1.4: setEnabled(true)
1.5: setEnabled(true)
1.6: setEnabled(true)
alt
[backButton]
1.7: onBackPressed()
[else]
1.8: makeText(getApplicationContext(), "Save before returning", Toast.LENGTH_LONG)
[item.getItemId() == android.R.id.home]
alt
[backButton]
1.9: onBackPressed()
[else]
1.10: makeText(getApplicationContext(), "Save before returning", Toast.LENGTH_LONG)
1.11: onOptionsItemSelected(item : MenuItem = item) : boolean
1.12:

ReportActivity:

Figure 16: SD for ReportActivity.onCreate(Bundle)

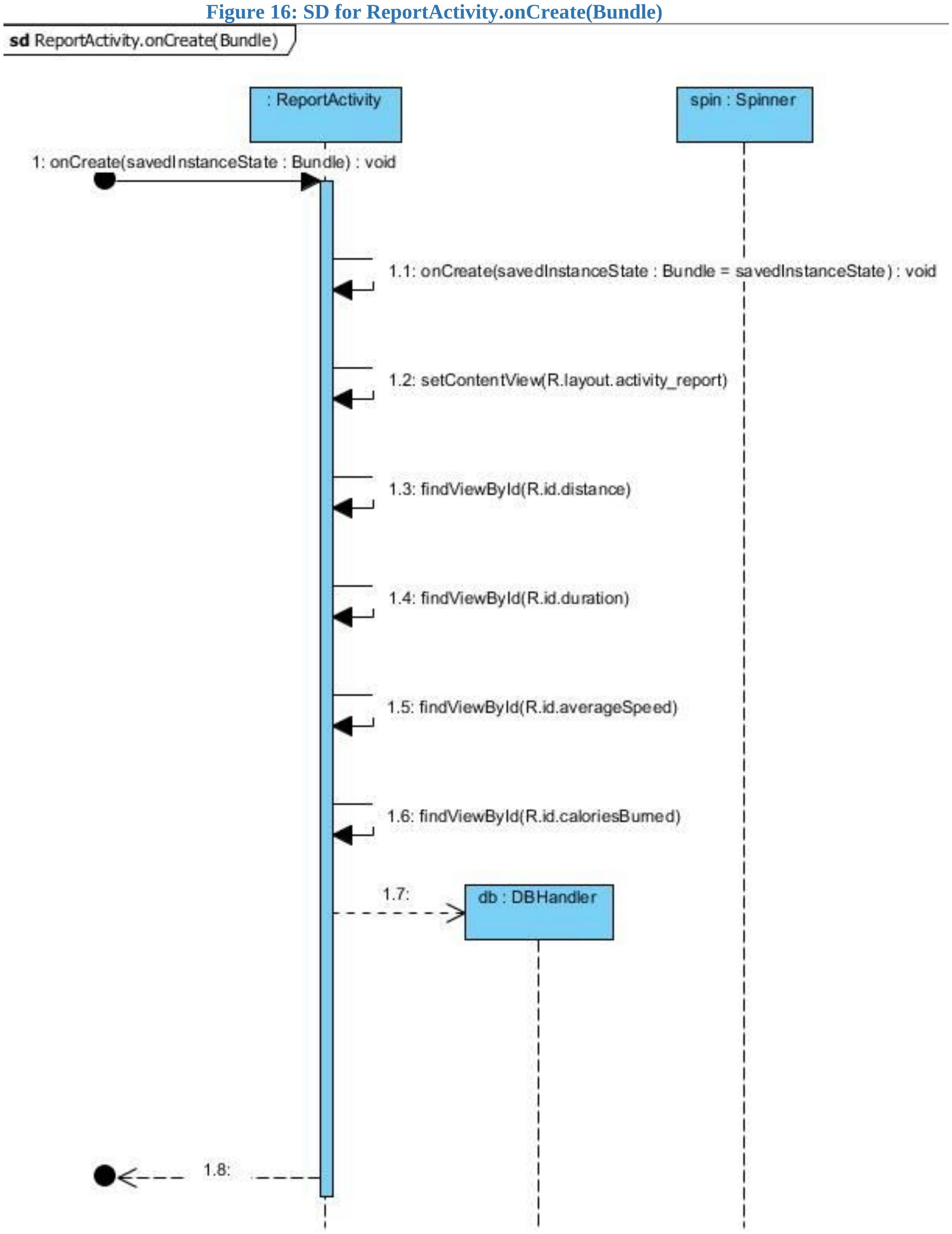
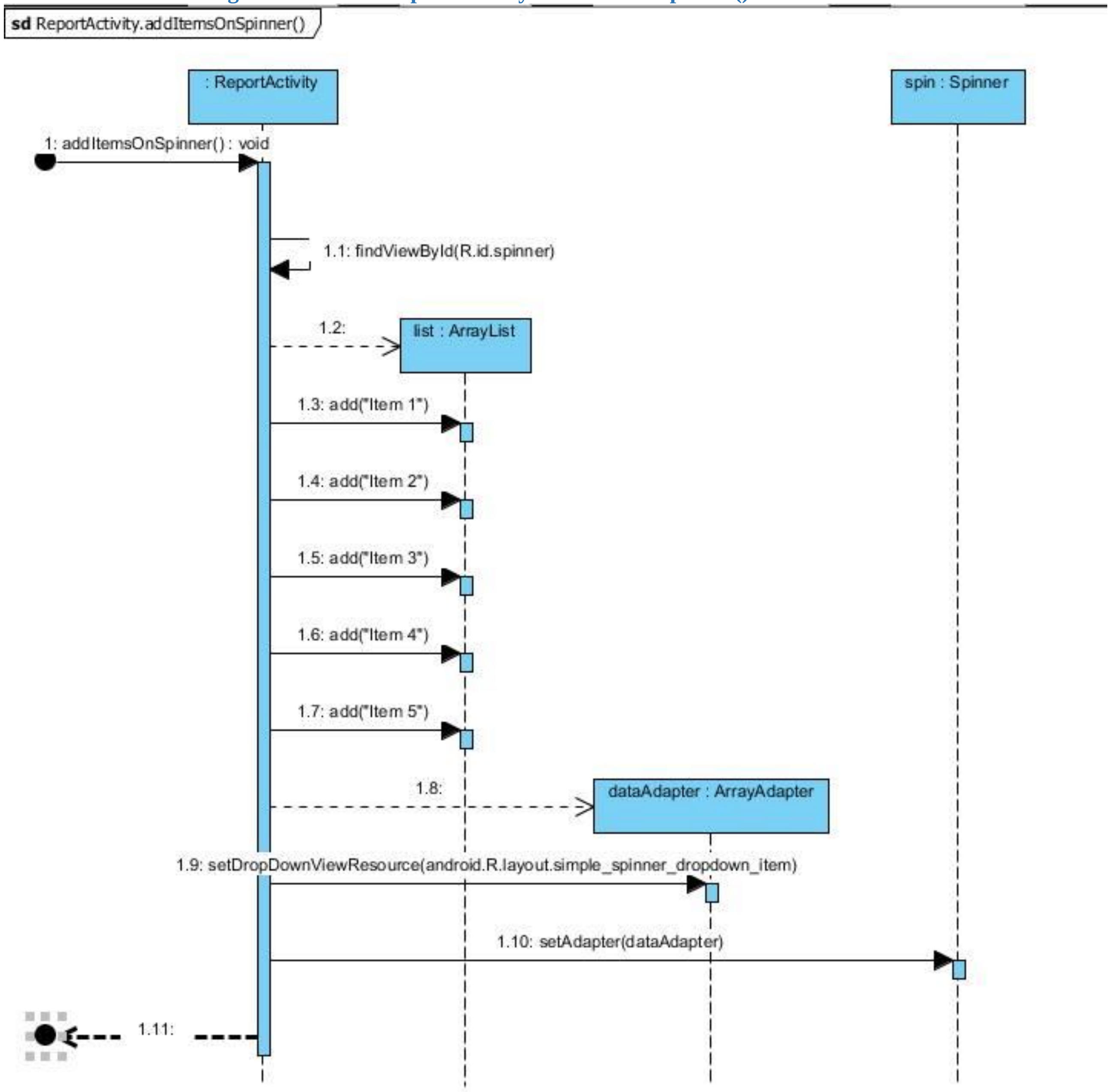
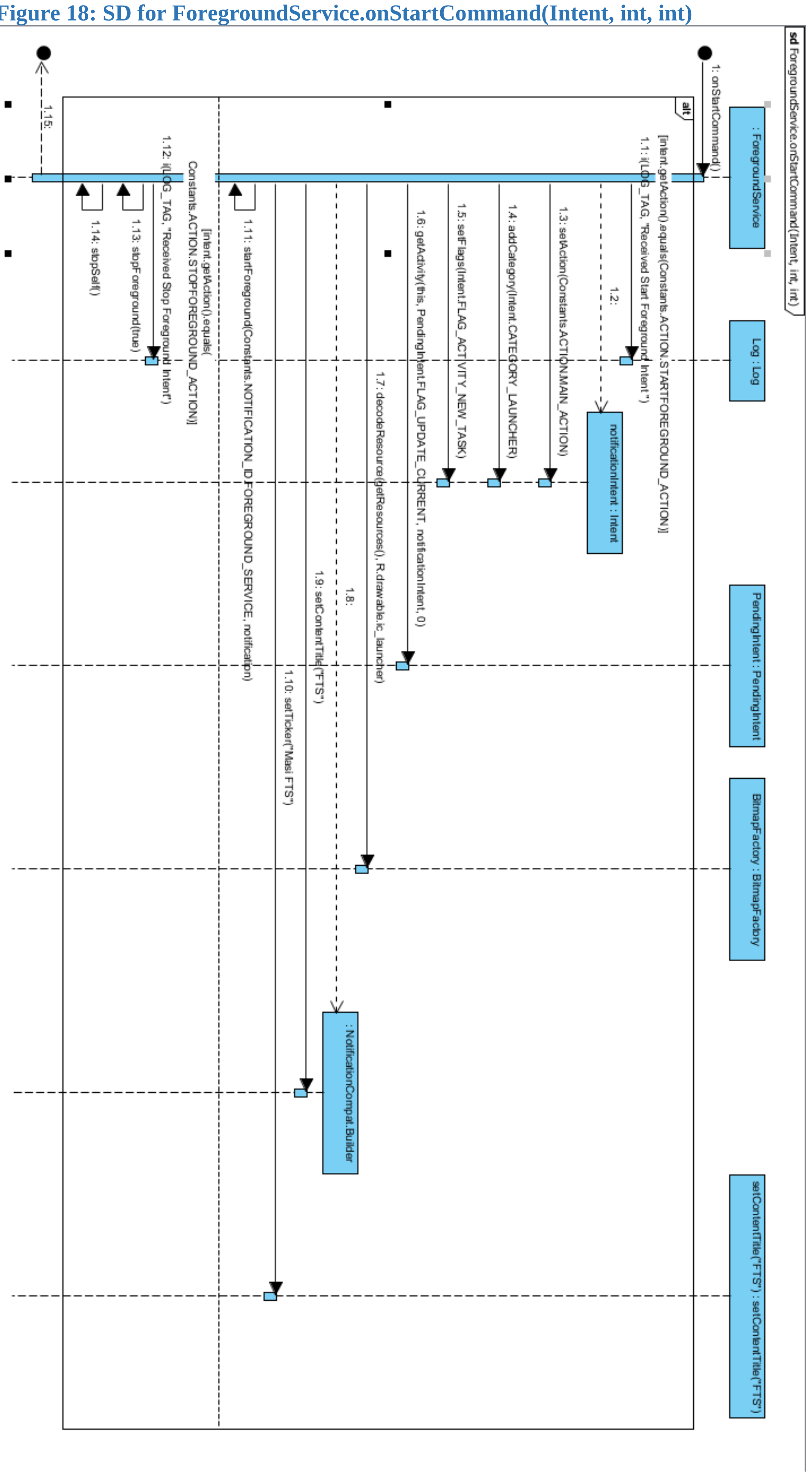

sd ReportActivity.onCreate(Bundle)
: ReportActivity
spin : Spinner
1: onCreate(savedInstanceState : Bundle) : void
1.1: onCreate(savedInstanceState : Bundle = savedInstanceState) : void
1.2: setContentView(R.layout.activity_report)
1.3: findViewById(R.id.distance)
1.4: findViewById(R.id.duration)
1.5: findViewById(R.id.averageSpeed)
1.6: findViewById(R.id.caloriesBurned)
1.7:
db : DBHandler
1.8:

Figure 17: SD for ReportActivity.addItemsOnSpinner()


sd ReportActivity.addItemsOnSpinner()
: ReportActivity
spin : Spinner
1: addItemsOnSpinner() : void
1.1: findViewById(R.id.spinner)
1.2:
list : ArrayList
1.3: add("Item 1")
1.4: add("Item 2")
1.5: add("Item 3")
1.6: add("Item 4")
1.7: add("Item 5")
1.8:
dataAdapter : ArrayAdapter
1.9: setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
1.10: setAdapter(dataAdapter)
1.11:

ForegroundService:

Figure 18: SD for ForegroundService.onStartCommand(Intent, int, int)



sd ForegroundService.onStartCommand(Intent, int, int)
: ForegroundService
Log : Log
notificationIntent : Intent
PendingIntent : PendingIntent
BitmapFactory : BitmapFactory
: NotificationCompat.Builder
setContentTitle("FTS") : setContentTitle("FTS")
1: onStartCommand()
alt
[intent.getAction().equals(Constants.ACTION.STARTFOREGROUND_ACTION)]
1.1: i(LOG_TAG, "Received Start Foreground Intent ")
1.2:
1.3: setAction(Constants.ACTION.MAIN_ACTION)
1.4: addCategory(Intent.CATEGORY_LAUNCHER)
1.5: setFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
1.6: getActivity(this, PendingIntent.FLAG_UPDATE_CURRENT, notificationIntent, 0)
1.7: decodeResource(getResources(), R.drawable.ic_launcher)
1.8:
1.9: setContentTitle("FTS")
1.10: setTicker("Masi FTS")
1.11: startForeground(Constants.NOTIFICATION_ID.FOREGROUND_SERVICE, notification)
[intent.getAction().equals(
Constants.ACTION.STOPFOREGROUND_ACTION)]
1.12: i(LOG_TAG, "Received Stop Foreground Intent")
1.13: stopForeground(true)
1.14: stopSelf()
1.15:

Timer:

Figure 19: SD for Timer.startTimer()

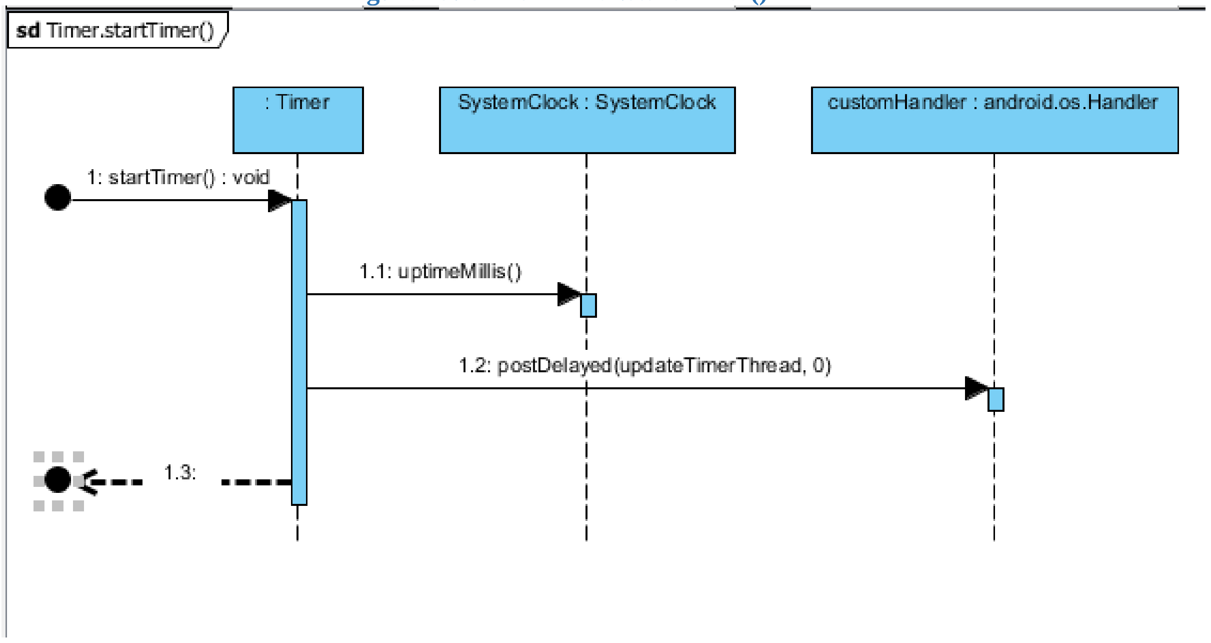
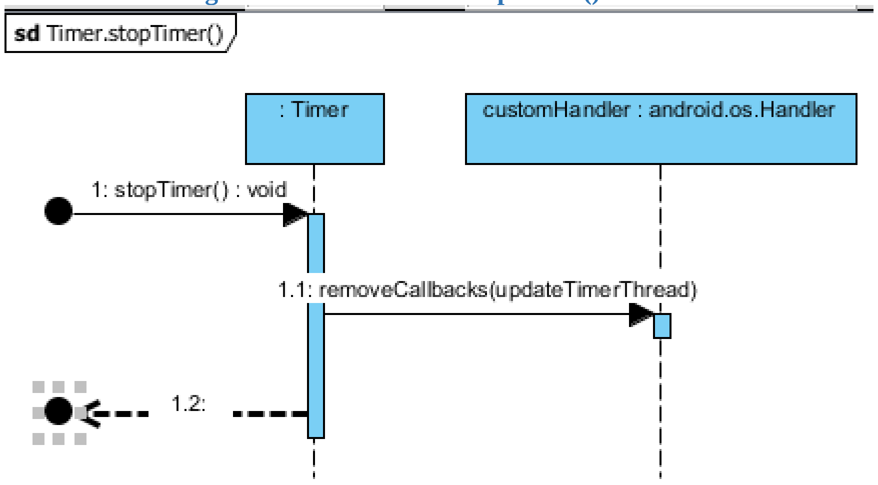
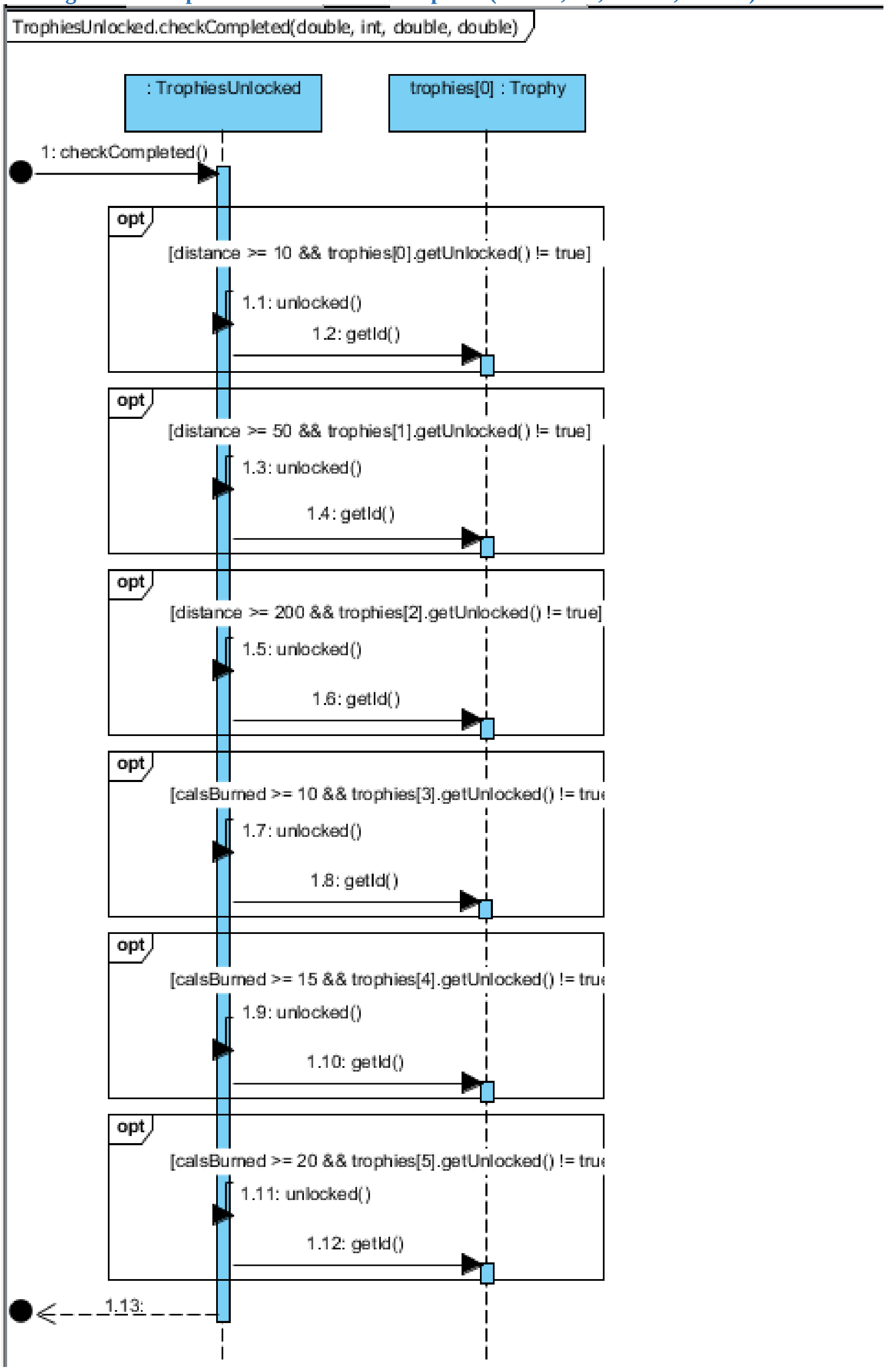

sd Timer.startTimer()
: Timer
SystemClock : SystemClock
customHandler : android.os.Handler
1: startTimer() : void
1.1: uptimeMillis()
1.2: postDelayed(updateTimerThread, 0)
1.3:

Figure 20: SD for Timer.stopTimer()


sd Timer.stopTimer()
: Timer
customHandler : android.os.Handler
1: stopTimer() : void
1.1: removeCallbacks(updateTimerThread)
1.2:

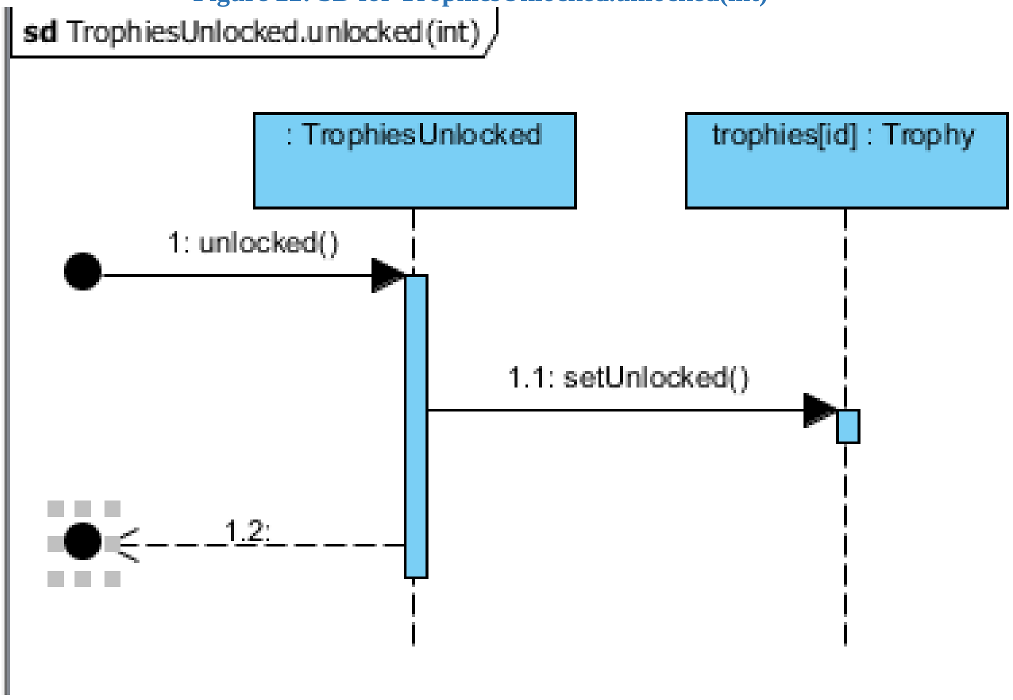
TrophiesUnlocked:

Figure 21: TrophiesUnlocked.checkCompleted(double, int, double, double)



TrophiesUnlocked.checkCompleted(double, int, double, double)
: TrophiesUnlocked
trophies[0] : Trophy
1: checkCompleted()
opt
[distance >= 10 && trophies[0].getUnlocked() != true]
1.1: unlocked()
1.2: getId()
opt
[distance >= 50 && trophies[1].getUnlocked() != true]
1.3: unlocked()
1.4: getId()
opt
[distance >= 200 && trophies[2].getUnlocked() != true]
1.5: unlocked()
1.6: getId()
opt
[calsBurned >= 10 && trophies[3].getUnlocked() != true
1.7: unlocked()
1.8: getId()
opt
[calsBurned >= 15 && trophies[4].getUnlocked() != true
1.9: unlocked()
1.10: getId()
opt
[calsBurned >= 20 && trophies[5].getUnlocked() != true
1.11: unlocked()
1.12: getId()
1.13:

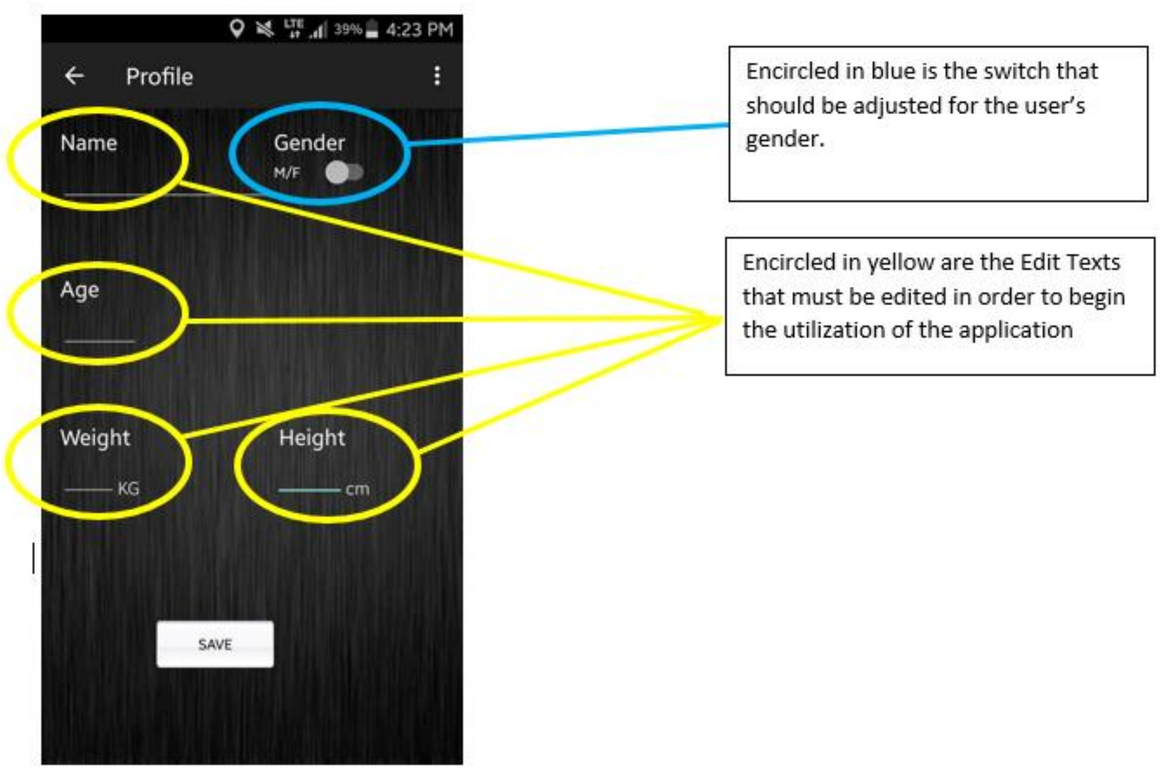
Figure 22: SD for TrophiesUnlocked.unlocked(int)



III. TESTING DOCUMENT

IV. MINI USER-MANUAL

Before beginning the workout, the user must allow to share their location via GPS. Once the user has given permission to the application to utilize the internal GPS of the cellular device, the application can be used without internet. To introduce the Fitness Tracking Software Application, the user begins by inputting initial information about themselves. This aspect of the application allows for the calculation of calories burned during a workout. When selecting the “Name” edit text, a keyboard will appear and this is where the name of the user belongs. For the gender option, there is a switch and either “male” or “female” may be selected by swiping the circular shape either to the left or to the right depending on whether you associate with a male or female respectfully. Upon selecting the “Age” edit text, a keypad with numbers will appear and the user must input their age; the same is done for the user’s weight and height in kilograms and centimeters respectfully. Without this information, the application will not proceed to save the runs that would be done by the user since this information is necessary when calculating the variables after the run.

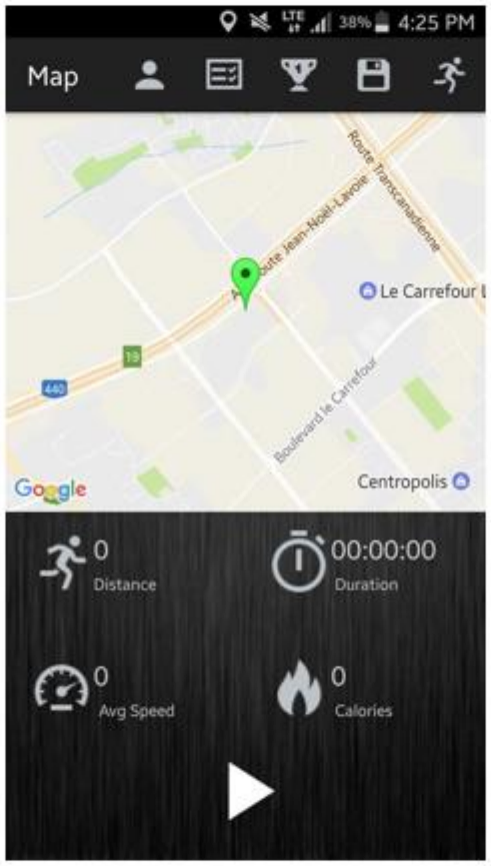


Encircled in blue is the switch that should be adjusted for the user’s gender.

Encircled in yellow are the Edit Texts that must be edited in order to begin the utilization of the application

Figure 23: User-Manual for Profile Page

After the user has input their information they can start their workouts that they would like to be saved. A map will appear with the user's current location and a button that looks like a "PLAY" symbol on it. To begin a run, press the "PLAY" button. This will activate the timer and allows the user



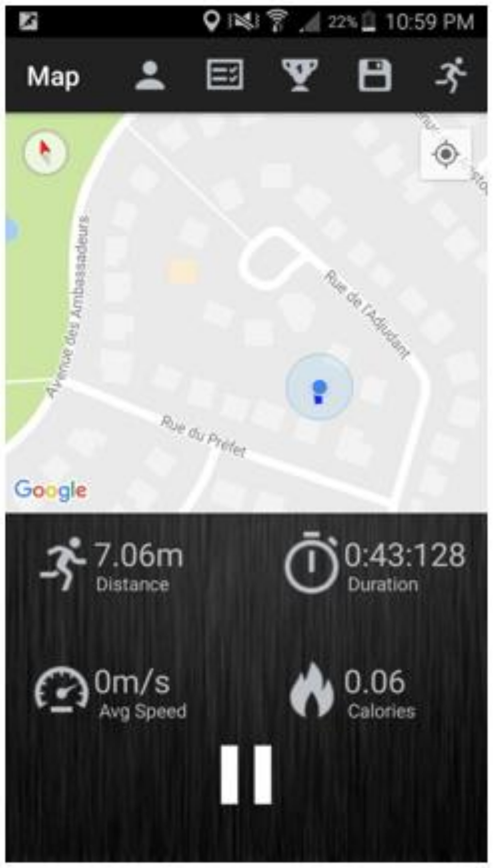
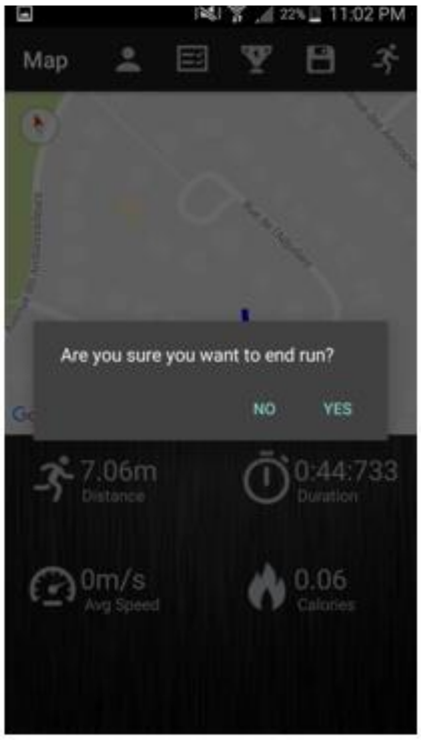


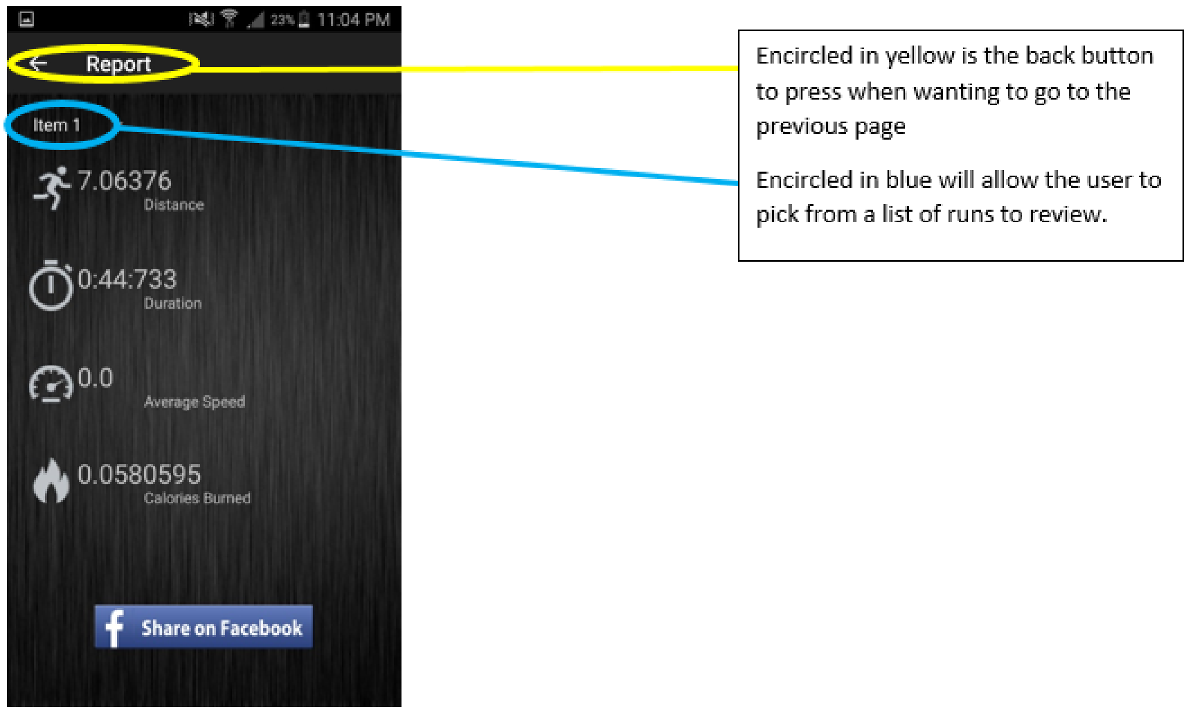
Figure 24: User-Manual for Map Page

to begin tracking their route. Once the run has begun, a blue line will appear in the path that is being taken by the user. The initial location of the user is depicted by a green marker. On the bottom of this screen, a number of elements will also continue updating throughout the workout. These elements are distance, duration, average speed, and calories. The average speed variable is calculated by dividing the duration by the distance and the calories burned is an equation that is based on the initial information inserted by the user and constantly updating with the new information of the run. The run can be paused at any time by pressing the pause button at the bottom of the screen. Once the pause button is applied, the play button will appear in which the runner can press at any time in order to continue tracking their route.

Figure 25: User-Manual for End Run



At the end of the user's workout, the user will press the pause button and then the universal "SAVE" button. This "SAVE" button is at the top right of the toolbar located at the top of the application screen. Once the user would like to end their workout, a prompt appears asking "Are you sure you want to end run?". If the user accidentally presses the save button without finishing the run, simply select "NO" and continue with the workout. If the user selects "YES", then a new page will appear that will reveal the final statistics of the workout. These variables include all of the ones that were visible during the run, except these are the final results that will be stored in the database that can be accessed at any time. The database can hold five runs in storage. The sixth run that is done by the user will then replace the oldest file, and so on and so forth. To view these previous workouts, there is a dropdown arrow next to the word "Item" followed by a number. This number corresponds to the run that is being viewed by the user. The report will show the user of the distance, duration of the workout, average speed during the entire workout, and the calories burned during the workout. To return back to the homepage, there is a back arrow next to the word "Report" in the upper left-hand corner of the application.



Encircled in yellow is the back button to press when wanting to go to the previous page

Encircled in blue will allow the user to pick from a list of runs to review.

Figure 26: User-Manual for Stats Page

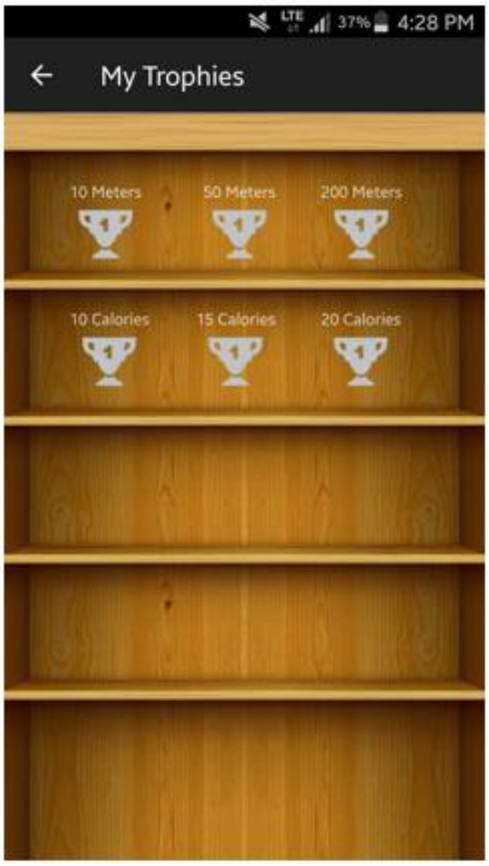


Figure 27: User-Manual for Trophy Page

Another feature of this application is the utilization of trophies. Once the user completes a certain predetermined goal the trophy will reveal itself as a different color. For example, once the user walks their first 10 meters, the first trophy will become unlocked. This feature can be accessed by selecting the "TROPHY" symbol at the top toolbar of the application. Once this button is pressed, a display case will show itself and all the trophies that have been received since getting the application.

References

--

Appendices

Appendix A – GROUP MEETINGS LOG

September 19, 2016

The Dream Team was formed after lecture, phone numbers and contact information was exchanged as well as a social media page that is used for communication.

September 26, 2016

A meeting was held to discuss the first theoretical assignment. The team created a schedule that prioritized certain sections and jobs for each member to complete and by when.

September 29, 2016

The team discussed possible and probable clients for the overall project as well as application ideas. The initial steps for the first theoretical assignment were made including an outline and a draft.

October 1, 2016

Each team member handed in personal information necessary for completing the first theoretical assignment as well as their personal assignments by our PO. The document was then assembled by Lucia and finalized by Michael.

October 2, 2016

A project was chosen and agreed upon by the group as well as a client who desired the product to be made.

October 5, 2016

The team finalized the idea about the application and discussed issues or concerns regarding the application. The SRS project was mentioned and steps were taken to progress the SRS report.

October 15, 2016

Together the team began writing a final report of the SRS after multiple drafts had been made. Each member had a specific section and task as to divide the workload evenly for each teammate.

October 20, 2016

Team met to discuss first sprint, what features to be completed and how to divise the work load.

October 23, 2016

Team met to continue work on sprint 1.

October 27, 2016

First week of sprint 1 completed, team met to review was has been done and what has to be done.

October 30, 2016

Application features for sprint 1 have been completed, design document almost completed, SRS version 2 corrected, testing document to be commenced for sprint 1.

November 2, 2016

All documents finished, Grammar check and validity of documents confirmed after review.

November 3, 2016

Meeting with Scrum Master to discuss end of Sprint 1 and beginning of Sprint 2.

November 4, 2016

Work continued for sprint 2 and weekly meeting to discuss work load.

November 9, 2016

First week round up meeting from the beginning of sprint 2.

November 13, 2016

Application features for sprint 2 have been completed, design document almost completed, SRS version 3 corrected, testing document to be commenced for sprint 2.

November 15, 2016

Last meeting to discuss everything about Sprint 2, presentation of all current version of documents and discuss what to change in said documents.

November 16, 2016

All documents finished, Grammar check and validity of documents confirmed after review.

November 18, 2016

First meeting of last sprint, discuss work schedule and lay out work.

November 20, 2016

Meeting to discuss progress on last sprint.

November 22, 2016

Prepared Oral presentation for investors, practice, complete sprint 3

November 23, 2016

Continuing to practice on oral presentation, add some finishing touches to sprint 3

November 24, 2016

Oral presentation completed

November 26, 2016

Meeting to discuss progress on the application and on all documentation.

November 28, 2016

Prepare all we need for the demo, schedule that demo, discuss items on the product backlog that we wish to implement

December 1, 2016

Application and final sprint completed, ready for the demo

December 2, 2016

Demo completed

December 4, 2016

Last meeting between team to discuss final submission and how to proceed

December 5, 2016

All documents finished, Grammar check and validity of documents confirmed after review. Final document submitted on moodle.

Appendix B – Client Interview

Interviewer: Matthew Masi

Interviewee: Carmine Masi

Background: My dad (Carmine Masi) has a bachelor’s degree in Computer Science and works at a software company called Nuance.

Matthew: “Please tell me what you want from the app.”

Carmine: “It is essentially a tool that would be used to view stats of a run/jog including the route taken.”

Matthew: “What are the specific features of the app?”

Carmine: “The main feature would be to track the route taken by the runner on a map.”

Matthew: “Would Google maps be okay to use for the map?”

Carmine: “Yes, that would be fine”.

Matthew: “And what exactly do you mean by tracking the route taken by the runner?”

Carmine: “I would like it to draw out a line that shows the route that was taken.”

Matthew: “For the line, would the standard blue line used in google maps be sufficient to draw out the route?”

Carmine: “Yes.”

Matthew: “To implement this feature, I would need to use the built in GPS of the phone. How accurate would you want it to be? For example would you want it to scan the location every second or every minute?”

Carmine: “It has to be somewhat accurate, however, I think scanning every second is a bit excessive. Every 5-10 seconds should be fine.”

Matthew: “How will you use this feature? Are there any steps that have to be taken before the app starts drawing out your route?”

Carmine: “The only prior step would be to press a button called “start run”. And this also brings me to the next feature that I want which is to display the time taken by the route. The timer would start when the “stat run” button is pressed”

Matthew: “So from what I understand, you want a common button to be pressed to start the route drawing and the timer, right?”

Carmine: “Yes, that is correct.”

Matthew: “And to stop the both the timer and the route drawing, do you also want a common stop button?”

Carmine: “Yes, but there are other things that I want to be calculated upon starting and stopping a run. Namely, distance traveled and calories burnt.”

Matthew: “How would you like to view these stats?”

Carmine: “How I picture the app, is that there is a main screen where there is a start and stop button. When the start button is pressed, on the same screen, I want the time to be displayed in real-time. When the stop button is pressed, there should be another button that can be pressed that takes you to another screen which will be the profile page of the current run.”

Matthew: “This profile page would be where you can view the time, distance and calories burnt for the current run, right?”

Carmine: “Yes.”

Matthew: “So for the route drawn on the map, how would you like to view it?”

Carmine: “Once you are on the profile screen, there should be a button at the bottom that can be pressed that will take you to another screen with a map on it displaying the route taken.”

Matthew: “Are there any other features that you would like to have in your app?”

Carmine: “Yes, I would to have a scoreboard that displays the top runs.”

Matthew: “And by top runs do you mean average speed or distance or time taken?”

Carmine: “Well I always use the same starting point and end point when I run so the time taken is the most important to be. I can then check the best time and see the route that is associated with it.”

Matthew: “How many entries total would you like to have in your scoreboard?”

Carmine: “I think 5 would be enough.”

Matthew: “How would you like to view this scoreboard?”

Carmine: “On the main screen, there should be another button that brings you to the scoreboard screen.”

Matthew: “How would you like to view each entry of the score board?”

Carmine: “On the scoreboard screen, I want something like a drop down menu that has the entries ordered from best time to worst time. When an entry is clicked on in the menu, the associated stats should be displayed on the very same screen. There should also be a button at the bottom of the screen that when pressed, brings you to the map with the route drawn on it for the selected entry.”

Matthew: “What would you like to do when the maximum number of entries has been entered?”

Carmine: “If a new entry needs to be put in the scoreboard and the scoreboard is at its max capacity, then the entry with the worst time will be replaced with the new entry if it is a better time.”

Matthew: “Any other features?”

Carmine: “That’s pretty much everything.”

Appendix C – Final Product Backlog

Product Backlog

Items are listed in order of most important to least important

Item 1	Follow the user and update the position every 5 to 10 seconds
Priority	Highest for sprint 1
Description	The application will get the location of the user, and track him by updating the location every 5 to 10 seconds. The location will be updated and a pin will be placed each time which will show the route of the user on the map
Status	Completed
Size	2 weeks

Item 2	Show a timer on the screen
Priority	Secondary for sprint 1
Description	The application will show a timer when the user starts a run. This timer stops when the user presses a pause button. The user can start again by pressing the start button.
Size	Completed
Size	2 weeks

Item 3	Enter data in “My Profile” page
Priority	Tertiary for sprint 1
Description	The user goes to another page (from the home page) where he can enter information such as, name, height, weight, etc. This information can be used later for calculation
Status	Completed
Size	2 weeks

Item 4	Implement a database
Priority	Highest for sprint 2
Description	Create a SQLite database that stores data to a text file. The database stores the calculated data such as speed, calories burned.
Status	Completed
Size	2 weeks

Item 5	Calculate the speed, the distance and calories burned (stats)
Priority	secondary for sprint 2
Description	From information entered in Item 3, calculate the speed of the user, the distance run and the calories burned in a run. This data will be handled in Item 4
Status	Completed
Size	2 weeks

Item 6	Display stats
Priority	secondary for sprint 2
Description	Display the information calculated in a run such as the speed, the distance and the calories burned. This information is visible to the user.
Status	Completed
Size	2 weeks

Item 7	Display information from recent runs
Priority	Tertiary for sprint 2
Description	Fetch data from the database and display it. Show the information (stats) calculated of the last 5 runs.
Status	Completed
Size	2 weeks

Item 8	Increase the accuracy of the GPS
Priority	Highest for sprint 3
Description	Get a better calculation of the distance ran. Decrease the error in the distance to the minimum possible, therefore increase the accuracy in the tracking.
Status	Completed
Size	2 weeks

Item 9	Add foreground service
Priority	Highest for sprint 3
Description	Add the ability to run this application in the background. Another application can be started and FTS continue running in the background. The run does not pause and the tracking continues
Status	Completed
Size	2 weeks

Item 10	Achievement notification
Priority	Secondary for sprint 3
Description	Notify the user when an achievement is completed by showing a notification. For example, if the user ran 10 meters, show it in a notification
Status	completed
Size	2 weeks

Item 11	Create a trophy activity
Priority	Tertiary for sprint 3
Description	Create an activity that display a bank of rewards to be unlocked such as, run a certain distance to get a trophy, or burn a certain amount of calories to get a trophies. This a tertiary feature to be developed more in the future.
Status	Incomplete
Size	2 weeks