

**Run Bud:
A running tracker and planner**

Final Report

Submitted for the BSc in
Computer Software Development

February 2016

by

Hector Standen

Word Count: 14845

Table of Contents

1	Introduction	4
2	Aim and Objectives.....	5
2.1	Primary Objectives	5
2.2	Secondary Objectives	5
3	Background.....	6
3.1	Comparison of smartphone OS	6
3.2	Comparison of software	8
3.3	Comparison of running styles	9
4	Technical Development	11
4.1	Design	11
4.1.1	Project Management.....	11
4.1.2	Interface	11
4.1.3	Use Case Diagram.....	12
4.1.4	Class UML.....	13
4.1.5	Workout Templates.....	13
4.1.6	Calculating a User's ability.....	14
4.2	System Implementation.....	14
4.2.1	Development Methodology	14
4.2.2	Xamarin & Mobile development.....	15
4.2.3	Interface	17
4.2.4	GPS/Map.....	18
4.2.5	Storing Data	19
4.2.6	Graph	21
4.2.7	Workout Tracking.....	22
4.2.8	Calculating user's ability	22
5	Testing	24
5.1	Functional Testing.....	24
5.1.1	Function Test Table	24
5.1.2	Generated Test Data	25
5.2	GPS Testing	26
5.2.1	Introduction	26
5.2.2	Deciding the accuracy limit	27
5.2.3	100-meter & 200-meter Test	28
5.3	Further Optimization.....	29
5.3.1	Accuracy Decay	29
5.3.2	3 Point Average	30
5.3.3	100 meter & 1KM Test - Optimized methods	30
5.4	User Testing	32

5.4.1	Introduction	32
5.4.2	Results	32
5.4.3	Evaluation	33
5.4.4	Conclusion	34
6	Evaluation	35
6.1	Objectives	35
6.2	Management Review	36
6.3	Further Work	37
6.3.1	Template workout rework	37
6.3.2	User creation of template workouts	37
6.3.3	Tracking of user health	37
7	Conclusion	38
Appendix A:	UI Design	39
Appendix B:	Accuracy Results (excerpt)	40
Appendix C:	Distance Testing	41
Appendix D:	User Testing Sheet	43
Appendix E:	Running Templates	46
Appendix F:	Ethics Checklist	47
Appendix G:	Time Plans	48
Appendix H:	Class UML	50
Appendix I:	Use Case Diagram	51
Appendix J:	User Testing Results	52
Appendix K:	Task List	54
References	57

1 Introduction

This report will be used to present the design and development of the smartphone application Run Bud.

The aim of the project is to produce a smartphone application specifically for joggers and runners. The application will rely largely on the GPS and the information gathered, this will be used to track user's workouts. With the information gathered, users can analyse their performance over the workout session or over the past months with a collection of results. The information can also be used to calculate a user's physical ability and from that calculation offer workout suggestions. For example, if a user tracked a workout where the user traveled for 40 minutes at 2.5 meters per second the application the application should suggest a similar workout such as 45 minutes at 3 meters per second the next time the user uses the application.

The application needs to be lightweight and easy to understand so that users not experienced with either fitness regimes or technology will be able to use it. This is expected to be the core user base. The title of the project has been revised from a running planner and diary to "Run Bud" in order to sound more friendly and catchy rather than generic.

The following section will decide some aims and objectives required for achieving the development of a successful smartphone fitness application. Following that, present some of the ideas and technology involved in developing such an application such as the demand for the product and ideal OS, and then compare and contrast them. Then the designs of the smartphone application will be presented, analysed and explained with reference to how they achieve and the aims and objectives. Along with design is, system implementation; how the designs were realized in the project with explanation of the technical details of code and other features implemented. The testing of the application along with the following refinements is an important and large section and has its own chapter composed of functional testing, user testing and GPS testing/optimization. The final sections rounding off the report will be the evaluation and conclusion to reflect on the project as a whole and by section to decide and conclude if it were a success or not and why.

2 Aim and Objectives

To develop a simple phone application to track running workouts of a user, store their data, provide a clear communication of the data and assist them in developing a fitness regime.

2.1 Primary Objectives

1 - Develop Fitness smartphone application

Design and develop a smartphone application making use of the smartphone specific hardware and capabilities. The application will assist users with physical activity or assist them in improving their physical capabilities and develop a fitness regime.

2 – Track the workout of the user

The application will be able to track the user's location with GPS, using this it will be able to calculate how far and fast the user is running and display a map of where the user travelled on their session.

3 - Track the progress of the user

The user will have the option to enter their personal details such as gender, age, weight, running experience and the application will keep track of their personal details and all of their previous workouts.

4 – Offer different types of workouts customized towards each user

Using a combination of data including the user's past workouts, their weight/gender and other preferences they have set, the application will be able to suggest workout routines customizing the distance, varying speeds throughout the run and length of the run.

5 – Personalised data display/graphs

The application will be able to display all of the information recorded for the user in different forms to communicate the information in a choice of suitable formats. This can include different types of graphs, tables or text to present the personalized information.

6 – Graphical User Interface

A simple interface providing a home page linking to the different sections of the app such as recording a running session, editing personal information or viewing their progress.

2.2 Secondary Objectives

Not essential to the completion of the application but would be useful additions.

1 – Personal accounts for users and server to store all data

As users may switch phones, the data will be kept on a server and each user will have a login used to download a personalised local cache of information.

2 – Reward System

The user will be able to earn different badges or themes for their account and application if they make significant progress in the different areas being tracked such as weight loss or total distance ran.

3 Background

For most people involved in exercise and fitness, the key to improving is progressive overload, this means increasing the intensity of that activity slightly every time in order to improve over time (Densmore, 2015). To increase a variable of the exercise such as speed, repetitions or distance they need to be tracked accurately, this is where smartphones have become very useful in recent years.

Phone applications are a consistently rising market in modern software development due to the rise of the smartphone and the evolving capabilities of them. The fitness industry is perhaps surprisingly well suited to the smartphone evolution because many sports/physical activities allow a phone to be carried with the person acting as a notepad, tracker, music device and more all on the go.

Evidence of the popularity of fitness phone applications is clear with the category becoming one of the most populated and downloaded.

“Usage of health and fitness apps has grown exponentially since 2013. Looking at data from over 6,800 iPhone and iPad apps listed under the health and fitness category, Flurry Analytics reports a 62% increase in the usage of health and fitness apps over the past 6 months, while the overall mobile app industry has only seen a 33% increase. Such a statistic means that growth in health and fitness apps is 87% faster than the entire mobile app industry.” (The Realtime Report, 2014)

With the population of apps in the health and fitness category, Run Bud will try to compete with them by offering a specialized service in tracking and planning running workouts.

The application is being designed to take advantage of the many new utilities of the modern smartphone such as GPS tracking and WIFI/3G connections to the internet but will compete with other smartphone exercise applications by being simpler and easier to understand.

3.1 Comparison of smartphone OS

With the worldwide growth in smartphones there has been competition over the operating systems to be run on the phones, Android currently owns the largest share of the market with 82.8% in quarter 2 of 2015 (*shown in figure 1*) but it does not necessarily mean it's the most suited to the project at hand.

With so many smartphones, “The worldwide smartphone market grew 13.0% year over year in 2015 Q2, with 341.5 million shipments” (IDC Research, Inc., 2015), even Windows phone with 2.6% of the total market has a huge user base with a total of 8.8 million units shipped Q2 of 2015 (IDC Research, Inc., 2015).



Figure 1 - Graph of smartphone OS market share

IOS

Developed by apple and used for the iPhone, this operating system is very popular and well-designed after many iterations improving on the functionality and the HCI. Developing an application for ISO isn't as simple as it could be, usually you are required to develop or at least compile on an Apple OS computer. Though Apple does impose more restrictions on developers working with their OS it does result in a safer, more secure and stable operating systems and applications. Unlike Android applications, apps on iOS can't access the operating system directly, which means if they crash, your phone still works fine – you just have to restart the app (Tennant, 2013).

Android

This is one of the most popular operating systems for phones to be running on because of how flexible it is. Usually apps developed for android will be made in Java which can be very effective. Xamarin is a software pack which allows the programming of both ISO and Android OS applications to be done in C#, it works well with Android as it can compile and run the apps on a Windows computer and the package can be run as a plugin for visual studio (Xamarin, 2015). Access to hardware components isn't very restricted, “It provides many packages of higher-level services to application that collectively form the environment within which they are constructed from reusable, interchangeable and replaceable components” (Kiran Bala, 2015). This means that accessing and using components such as the GPS would be easier and more flexible.

Windows Phone

Windows phone is great to develop on because it has a whole programming environment built into Visual Studio. With previous experience using Visual Studio it is easy to move to Windows phone programming, it can be programmed in C# as well as compiled and run on an emulator through Visual Studio. A large issue with the Windows phone is the lack of applications available on the app store, this could be seen as a positive as there would be very little competition on the market but it also means there is very little activity in the market and regardless of the quality of the application the user base would be limited.

Conclusion

Android would be the best operating system for Run Bud, this is because with Xamarin it can be made and compiled with visual studio in C#, this reduces a lot of the issues with programming in an unfamiliar language and reduces restrictions with compiling. Android is superior to Windows Phone in terms of its popularity, has more APIs and libraries to pull from as well as more documentation "Android has an amazing open source community and most answers are a code lookup or a search away. In fact, if you are concerned about performance and like understanding exactly what is happening with your views and data structures, Android is the platform of choice." (Kapoor, n.d.).

The simple access to hardware components with Android will also be a large advantage over other operating systems as it is a key part of the project.

3.2 Comparison of software

Android applications are designed to be coded and compiled from Java. Java is currently the most popular programming language in the world, one of the biggest reason for this popularity are probably its platform independence and wide use on mobile platforms (Walker, 2015).

A modern tool for mobile development is Xamarin which allows ISO and android software to be programmed in C# and compiled into the relevant code for each device; giving another choice of programming tool "Xamarin is used by over 1 million developers to build consumer, gaming, and enterprise apps, and has more than 15,000 paying customers in 120 countries" (Xamarin, 2015).

"if you have limited development time and need an application that can run on almost any operating system, then Java seems to be the obvious current choice. But if you know that your application will run on Windows, and you have or don't have limited development time, using a good type-safe, powerful language like C# seems to a very excellent decision". (Kirk Radeck, 2003)

Programming in C# with Xamarin gives it a lot more portability allowing it to generate code for IOS and android in addition to being the natural coding language for Windows phones. This would give a lot more flexibility and the possibility for expansion in the future, in addition to C# being a safer and more powerful language making it a priority choice for programming language in this project.

Getting access to Xamarin software usually requires a subscription to the service. Student offers allow university students to sign up for the service free for the duration of their university course (Xamarin, 2015). Once an account is obtained the software package can be downloaded and used, it offers the option of using its own programming environment or using Visual Studio with an extension package to use all of the features from Xamarin, within Visual Studio.

Visual studio is a larger, safer and has more documentation, the developer also has a more experience using Visual studio and wider access to the IDE having it installed on all available devices, so it is a better-suited choice for the project than the Xamarin environment. As of April 2016, Microsoft acquired Xamarin and are including Xamarin as part of every Visual Studio edition for free, showing the growing popularity of the software and Microsoft's support of it.

There are two main APIs offering location services; Androids default locations API and there is the Google locations API. Although accessing Google's API requires installation of extra components and verification of the license, it offers more information and better services than the default Android API including updates for bugs and fixes that otherwise along with Google maps which is an important feature of the application, this makes Googles API the better choice for the project.

“with the new API, you have an easy to use solution that works across all Android devices and is guaranteed to work as long as the device has Google Play Service installed” (Hellman, 2013)

Data storage will no doubt be required in the application, to store user data as well as template workouts. Most of the data on a phone application is not persistent, this means that when the application is closed all of that data is lost. Two options are offered by the standard API for persistent data storage; preference files and SQLite databases (Lee, 2011). Shared preferences stores primitive data types, each object is stored in a pair with a key, the key is then used to request that object. SQLite stores custom objects because it can divide up an object into its primitive types and use each variable as a column in a table. Generation of these tables is done by SQLite and recreation of the objects on request is also done automatically. SQLite can perform SQL request like a normal database, for example selecting template workouts where the distance variable is over 500. To store custom objects such as a template workout in shared preferences would require serialization of data, this means processing the object and writing it as a primitive type so it can be stored or transferred. With the amount of data that will be stored and the data largely being custom objects, SQLite seems a lot more suited to the task.

3.3 Comparison of running styles

The application will be providing template workouts for users. These will be suggesting the speeds and interval pattern of the run in order for optimal workouts.

The first style of training to be offered is standard distance training, alternating between walking and running, working up to running for an entire workout and scaling the intensity by changing the distance/time of the workout. This is the most common form of running exercised.

High intensity training is offered as an alternate option to the traditional medium intensity training as it has been shown in studies to increase metabolism post workout and in many cases burn more calories (Exercise Science Department, 2005) . It has grown in popularity in recent years not only because of the increase in metabolism but also because it reduces the amount of muscle loss compared to other cardiovascular fitness exercises. This is being added as a second style of running that the user can use as a template workout; it works well as an alternative style because the format is quite similar to that of the long distance style training.

In Both styles of the workouts the distance/speeds of the walk and run periods are consistent throughout the workouts meaning all of the workouts can be stored in the same template which at a minimum only needs to store 3 variables; walk seconds, run seconds and cycles.

There are different experience level runners meaning the workouts also need to be scaled up or down in terms of physical effort required, this can also be done by adjusting the speed, distance and intervals.

Having the same template for both styles of running is useful in keeping the application simple and not confusing the user, it also makes the workouts both scalable in the same way, for example every workout could have their walking seconds increased by 20% in order to make the entire set of template workouts slightly easier.

4 Technical Development

4.1 Design

4.1.1 Project Management

Breaking the objectives (Chapter 2) down further helps to identify all of the tasks including research, development and testing, the full task list can be found in Appendix K. Once all tasks have been identified, they can be assigned an estimated duration which is the development time assigned to them, the majority of tasks for this project averaged about a week's development time. Using the task list with estimated durations, a time plan can be created for the Run Bud project duration, time plan found in Appendix G.

The Run Bud time plan aims to get the majority of the software development completed in the first half of the development time, the second half is then dedicated to secondary objectives, testing and the final report. This allows for mistakes in the first half of the project, if the development schedule is behind the plan, secondary objectives can be cut out.

The project will involve physical exertion and will also collection some personal information from users, for these reasons it's important to conduct a risk assessment to identify any other risks and to find ways of avoiding or countering any of the risks identified, this can be found in Appendix F.

Additionally, an ethics approval form was completed and accepted for the project, assuring test participants would be in a safe environment and steps would be taken to protect their identity and person information.

4.1.2 Interface

The interface was designed using Microsoft PowerPoint as a rapid prototyping environment, as this accelerates the design process, increases creativity through fast user feedback and helps detect errors earlier (Steven D. Tripp, 1990). This was chosen over writing out the designs by hand because items such as frames and buttons can easily be designed with the selection of shapes, text boxes and lines provided with PowerPoint. All of the items can be reshaped, colored and copied with ease and a consistent color scheme and theme can be kept throughout the different pages of the application. Buttons can be hyperlinked to open other pages allowing the design to be interactive and the user journey through the application be experienced before any code is written.

The UI was designed using a blue colour scheme, this is a friendly and gentle color and allows the black text to easily be visible over it, shown in figure 2. It has been designed with vision disabilities taken into consideration; "yellow and blue are true colours which stand out to those with deficient red/green colour vision." (COLOUR BLIND AWARENES, n.d.).

The buttons styles and text styles are kept consistent throughout the application for presentation and ease of use. In order to keep the application looking simple there is an effort to keep the amount of information presented on a single page limited. This means that some objectives have been designed over several pages, such as selecting a suggested

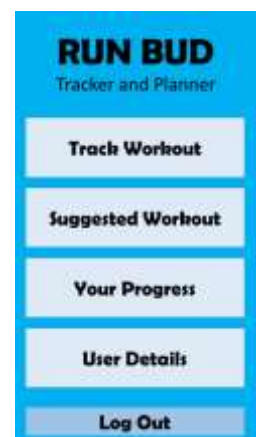


Figure 2 PowerPoint home screen design

workout which allows you to open a page listing your suggested workouts, all the workouts and another page which shows more information on a particular workout. The full table of UI designs can be found in Appendix A.

When designing for smartphones it's important to remember that users will have different screen sizes, in resolution, size and in aspect ratios. This means not only designing an interface that will shape and change depending on the device but also an interface that will be clear and visible on lower end devices with low resolution (Lehtimäki, 2013).

4.1.3 Use Case Diagram

The Use case diagram (figure 3) was designed with the home page of the application being the starting and ending points of all journeys within the application. When developing for Android it is very important to keep in mind the activities opened and where they are opened from. This is because when a user opens a child activity from a parent activity the parent activity is removed from focus but kept on a stack of active windows so if the user were to hit the back button it would return them the previous parent activity at the top of the stack (Android, 2015).

User journeys are designed so be separate in order to avoid any backlog of active windows which would cause the application to run slower, the system architecture to be more vulnerable to errors and for users to lose their place in the application.

In the design, the customer starts and ends every journey from the home page and there is no way to access an activity from another user journey without starting again from the Home page or finishing the current journey.

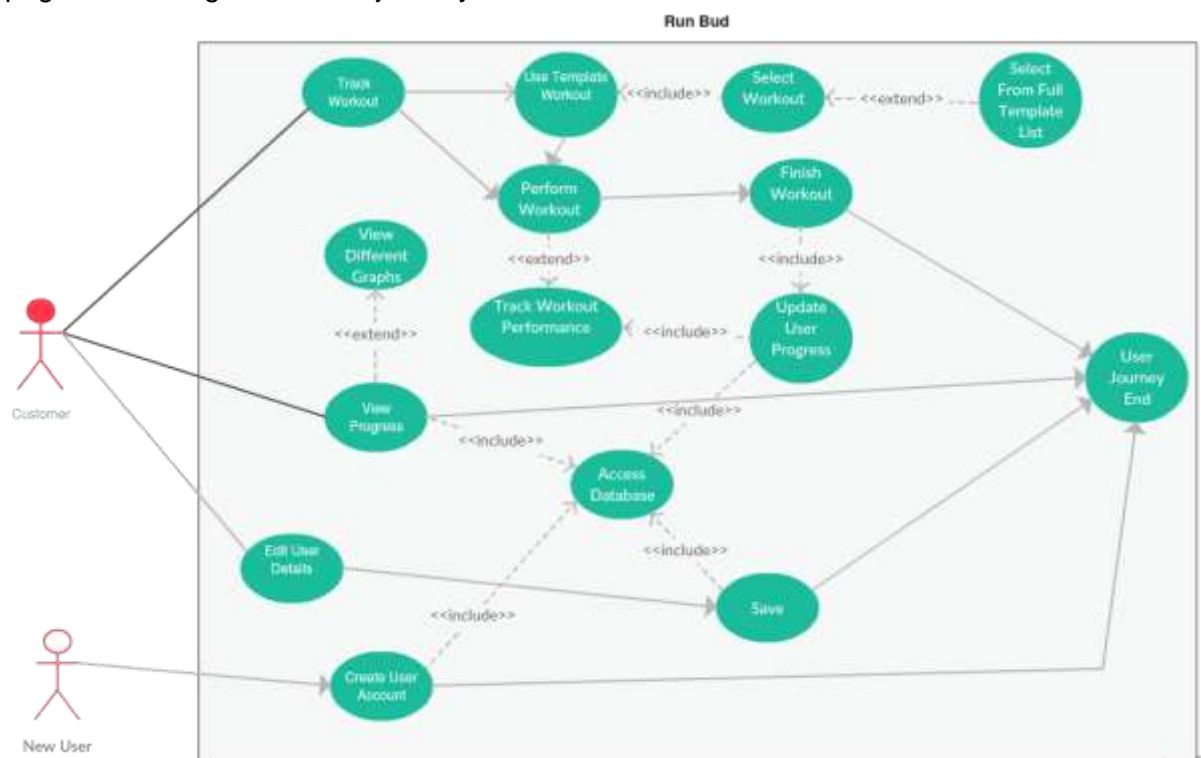


Figure 3: Use case diagram

The diagram helps to emphasize the importance of the database; it is accessed from every user journey so it will need to be designed and implemented in a way that makes it accessible from all parts of the application.

4.1.4 Class UML

This class UML (*figure 4*) shows the design of the main class components of the application. At the center of the diagram and the core component of the model is the user, the application is designed so each user has their own version of the program where only their data is available and it feels customized towards them. This is done by generating things relevant to the user, for example when the user views a graph the same class is used every time only the input variables are slightly changed. A larger, more readable copy of the UML can be found in Appendix H.

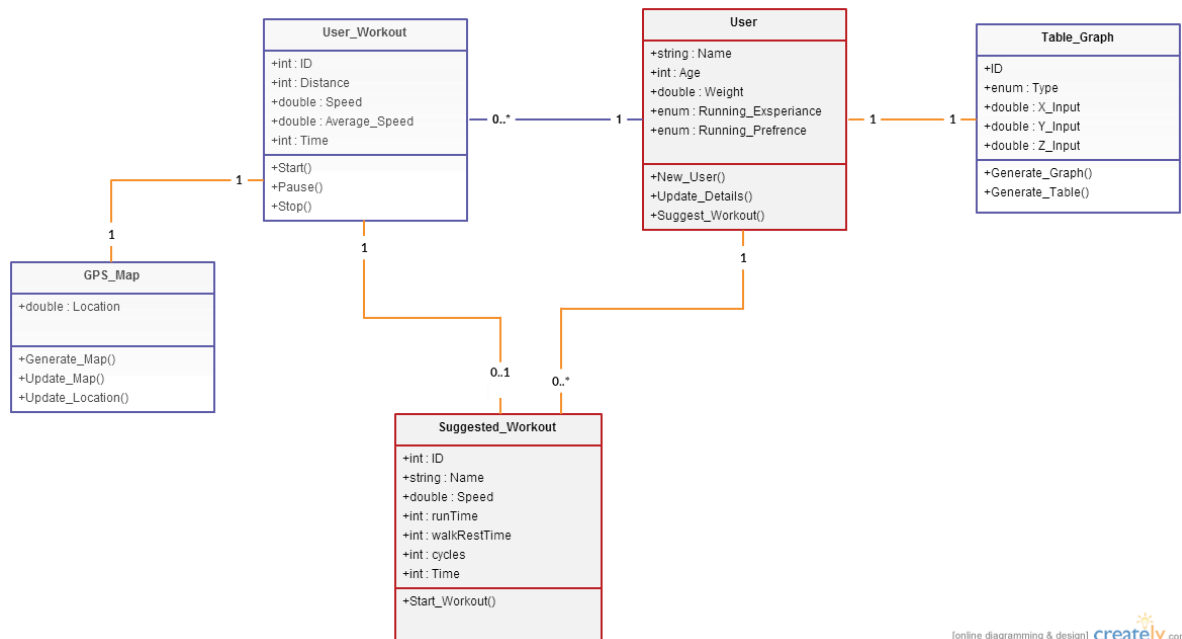


Figure 4: Class UML design

4.1.5 Workout Templates

A range of template workouts is required so that users of different skill ranges are able to find a workout matched to their ability. As discussed in section two, a comparison of running styles, two different styles are offered to cater to a wider crowd and different goals. Each style has three difficulty levels and 6 to 10 stages to progress through in each level.

The levels have been designed for a user to start at stage one and progress through the all the stages in the level before moving to the next level and starting from stage one again, although a user had freedom to pick any workout they want (See appendix E for workout templates design).

As each workout consisted of cycles between running and walking or resting the class to hold the templates (*figure 5*) was simple and easy to design and both styles of workout were able to be kept in the same class. This means that when they are fed into the workout activity only one algorithm to process them will be required for any template workout.



Figure 5: Class design for workout template

4.1.6 Calculating a User's ability

To gauge a user's running ability, they must complete a workout so their time and distance can be analysed. This creates the problem of needing to suggest a workout for the user before knowing their skill level, a simple solution to this is to ask the user on their first use of the application what their estimation of their ability is or their previous experience with running. This way the first suggested workout the user performs isn't too slow and easy for them but more importantly it isn't too hard for them which could result in injury.

Average jogging and running speeds will be set for the different training styles, if the user is exceeding the expected speeds throughout the workout then they will be suggested more challenging workouts. Conversely, if the user isn't meeting the expected speeds they will be suggested the same or less challenging workouts. A design of this algorithm as shown in figure 6, the algorithm has been designed so that users are only being analysed when they are running and not when they are in a walk/rest phase.

```
EndRunCycle() \\repeated after each running phase
{
    speedDif = runSpeed - suggestedSpeed;
}

EndWorkout()
{
    Levelchange = speedDif / difficultyDifference;
    Suggestworkout(Levelchange);
}
```

Figure 6: Performance algorithm design

This design is updated upon implementation as explained in the next section, system implementation.

4.2 System Implementation

4.2.1 Development Methodology

The development methodology used for the project was agile development, in this way objectives can be considered and tackled one by one. In addition to this, the developer has little experience with mobile development and changes to the design are to be expected throughout development.

“In Agile Software Development, developers continually deliver a small scope to production environment in short cycles by doing analysis, design, coding and testing in parallel. This process enables developers to respond to mid-way changes.” (Yoshihito Kuranuki, 2014)

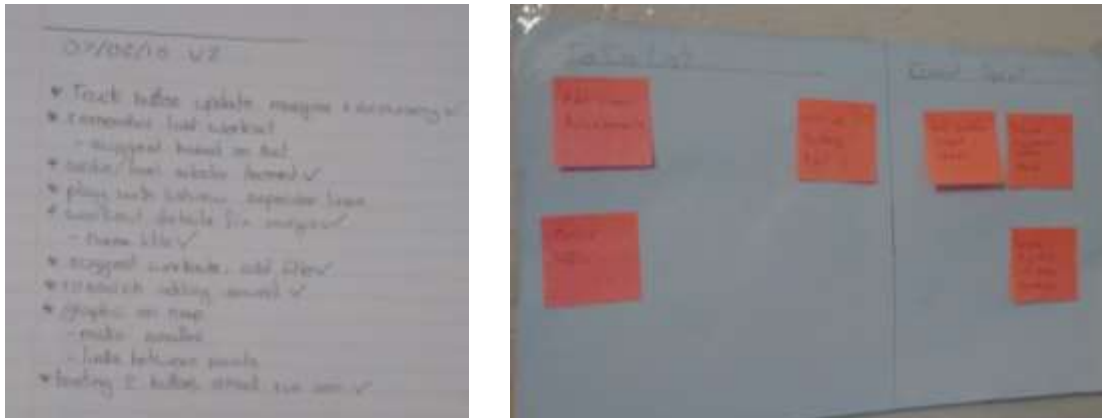


Figure 7: Daily checklist & Weekly SPRINT chart(photo)

This also can be considered as incremental development as sections were developed one by one as presented on the time plans in Appendix G. Although the project had only one developer a SCUM/SPRINT mentality was used on day to day basis, this meant organizing daily and weekly goals with a chart on the wall to hold post it notes and daily checklist objectives made to monitor progress throughout the day.

4.2.2 Xamarin & Mobile development

As noted previous in section 3.2, development for the application is to be done using a special software tool Xamarin. This allows the coding of Android project, previously restricted to Java, to be coded in C#. For this project, Visual Studio was used with a Xamarin extension rather than the Xamarin IDE because Visual Studio is a modern and highly refined IDE with features such as code completion, solution management and integration with other software such as visual SVN.

Programming Android applications in C# is quite similar to the traditional way with Java or objective C, this is because Xamarin provides bindings for nearly the entire underlying platform SDKs with strong typing meaning they're easier to navigate and use (Xamarin, n.d.).

Included in the Xamarin toolkit is an SDK emulator, an essential tool in the development and testing of Android applications. In order to test software throughout development, phones are emulated on the same computer being developed on and the software is installed on the emulators. The virtual phones can be interacted with like a normal mobile but indicated with the mouse rather than with touch, configurations can be adjusted such as network connection and events triggered such as an incoming call in order to test the phone in dynamic environments.

The emulation allows for many phones types to be tested on, with different screen sizes and hardware configurations and is important for mobile development and testing.

Creating the project using Xamarin is similar to with a normal project within visual studio, only the templates are different (figure 8).

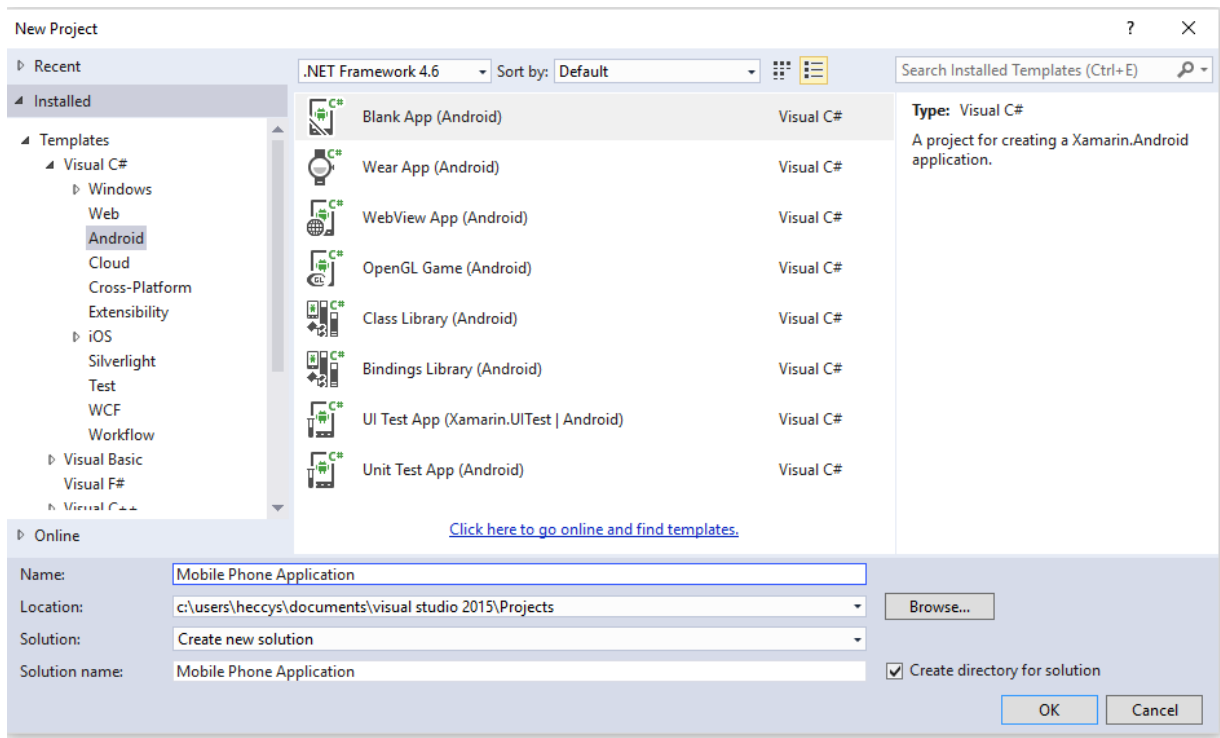


Figure 8: Android project selection

Once the project has been created and the android virtual emulator has been setup the solution is ready for code. Some fundamentals to understand when coding for Android is that that different android phones runs on different versions and so applications are developed for target versions and have minimum versions of the Android OS they run on (Xamarin, n.d.), (figure 9).

Another core feature with Android development is activity lifecycles, activities are sections of code, often with an interface attached. Each activity has a lifecycle of states such as running, paused, backgrounded and stopped, changing states causes an event to be triggered in the activity for that specific states. For example, the entry point for most activities is the OnCreate event which is called every time a new instance of an activity is created.



Figure 9: Android target versions

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);
    // Set our view from the "main" layout resource
    this.RequestWindowFeature(WindowFeatures.NoTitle);
    SetContentView(Resource.Layout.Map_GPS);
    this.Title = "RunBud : Workout Tracking";
}
```

Figure 10: OnCreate method in Android activity

A final mobile development only feature is permissions, these are components of the phone that the application needs to declare it wants permission to use at compile time for security purposes. Depending on the component and the phone the software is running on, sometimes the phone will also ask the user for permission when the application starts up or

when it tries to use it (Meier, 2008). Permissions, specifically with regards to the GPS and accessing it was very important in the development of Run Bud.

4.2.3 Interface

When developing for Android, the interface and the code are logically separated. This provides a lot more flexibility, for example the UI can be developed separately and could have multiple versions of the interface for the same section of code used for different situations, with different types of phones or hardware (Meier, 2008).

For this project all the UI design was done with XML, the common choice for interface design with Android. Using the combination of Xamarin and Visual Studio the programmer can switch between design view and the source code for the XML interface (*figure 11 & 12*). Most of the work is done in the source code view as it provides a lot more accuracy and more values to customise whereas the design view is used to visualise the interface without having to deploy it to a device saving a lot of time.



Figure 11: Design view of interface



Figure 12: Source code view of interface in XML

In order to keep a consistence theme throughout the application in colour and button/shape styles, two techniques were used. The first was defining a style theme for all the interface pages in the application, this was designed in XML and inherited from a premade theme already offered (android:Theme.Material.Light).

“Starting with Android 4.0, all manufacturers are required to include Android Holo themes in their Android distribution to fulfill the Android compatibility requirements. In practice, this means that developers and designers can depend on the default themes and define their user interfaces by using the default themes or extending them” (Lehtimaki, 2013)

Adding some customization such as different background and control colours to match the design of the interface and saving it as a custom style allows it to be assigned as the theme for the application as shown in *figure 13*.

```
<application android:label="RunBud" android:icon="@drawable/Icon" android:theme="@style/MyCustomTheme">
```

Figure 13: Code view of assigning a custom theme

In addition to this styles were design for buttons and textboxes specifying the colour, text gravity, border width, corner sharpness and padding so all the UI controls are consistent throughout the application.

A problem discovered in development is that screen rotation caused the UI to be recreated and so the whole activity reset and acted as if it had just been restarted. There are ways around this using shared resources and saving states, but for this project it didn't seem like there would be any advantage in having the screen rotate and could potentially be frustrating for users on the run having their screen unintentionally rotate because of their movement when they are trying to use it. The simplest and best fix for this was to disable any screen rotation throughout the application.

In an effort to keep the application easy to use, throughout the application changes were made to keep the amount of data entered as minimal as possible, spinner controls and sliders should be implemented where possible rather than text entry (McClure, 2012) (*figure 14*).

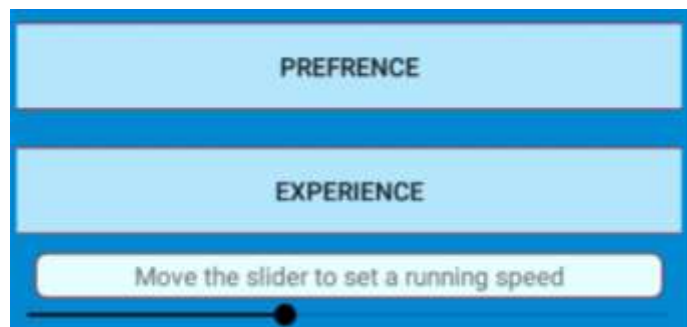


Figure 14: Spinner and slider controls

4.2.4 GPS/Map

The graphical positioning system is a very important part of the program, because of this it was the first function to be implemented when building the application. As noted in the background research (3.4) it was decided that the google API would serve better than the default android location components.

The first challenge was installing the components; this was a simple process of using an inbuilt interface of Xamarin to search and install the google components (*figure 15*).

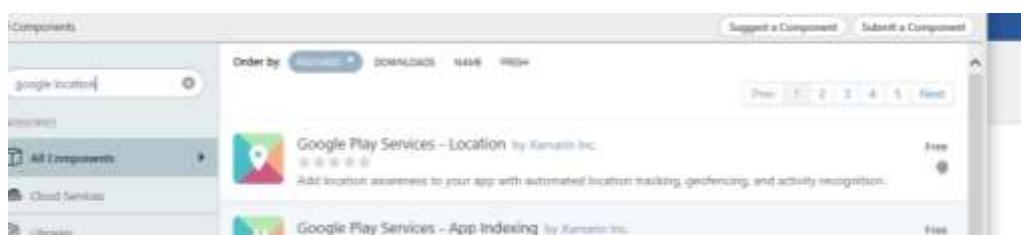


Figure 15: Installing components

This is the way all components have been installed in the project.

Next, an activity was designed to attach to an interface and display the map fragment in addition to tracking the current location of the user.

In order to access and use the components, the class needs to be extended to them (*figure 16*).

```
[Activity(Label = "@string/GPS", ScreenOrientation = Android.Content.PM.ScreenOrientation.Portrait)]
public class GPS : FragmentActivity, GoogleApiClient.IConnectionCallbacks, GoogleApiClient.IOnConnectionFailedListener
{
}
```

Figure 16: Code view extending class

Once the imported classes are instantiated they can be used easily and customized. There are two components installed from Google, location service as well as the map service. They are installed and initialized separately but work well together in the same activity. The location services work with the map by supplying it with the current GPS location. The `OnLocationChanged()` method which is called every time there is a new location result (set to be every second) also contains a method for updating the camera location on the map and drawing the path from old location to new in order to display the user journey.

The map is defined as a global variable in the activity, this allows it to be accessed everywhere else in the activity, including the location methods. The Google Maps class allows us to draw on the map by feeding the `GroundOverlayOptions` a graphic to place on the map and a geo position to place it at on the map. In addition to this, the Google Maps class also has a `PolyLine` class line which allows us to link the location updates visually on the map with lines between the points.

```
float[] distance = new float[1];
Android.Locations.Location.DistanceBetween(oldpos.Latitude, oldpos.Longitude, pos.Latitude, pos.Longitude, distance);

LatLng NEWARK = new LatLng(location.Latitude, location.Longitude);
GroundOverlayOptions newmarkMap = new GroundOverlayOptions().InvokeImage(BitmapDescriptorFactory.FromResource(RunBud.Resource.Drawable.Red)).Position(NEWARK, 10f, 10f);
mMap.AddGroundOverlay(newmarkMap);
Polyline line = mMap.AddPolyline(new PolylineOptions().Add(oldpos, pos));
```

Figure 17: Code view of groundoverlay

Calculating Distance and speed were quite simple, the Google locations API provides a method for calculating the distance between two geo locations so every time a new location update is received (figure 17), the distance between that and the last point received is calculated and added to a running total for the session. Using this distance calculation alongside a timer makes it a trivial calculation to find the average speed the user is moving at (figure 18).

```
avrSpeed = total / (stopwatch.ElapsedMilliseconds * 1000);
```

Figure 18: Speed calculation

The user's current traveling speed rather than average is calculated from the last 2 updates and done automatically by the Google location API so it's easy to retrieve the information and display it.

4.2.5 Storing Data

Storing and transferring data was one of the first obstacles encountered. Starting activities with adding extra information as well as using return intents for transferring data back, was the first method of data transfer between activities. Only primitive data types are able to be transferred through this method so another component `Json.NET` was installed in order to serialise custom objects into a string and allow then to be transferred to other activities through Intent extras. Although this did work it was not very flexible.

Rather than hold information in a central activity and pass it through to other activities it would be a lot easier to hold all of the information in a local database and pass through an identifier as an Intent extra, then the activities can query the database with the identifier to find the object required.

SQLite is a data engine running in Android and is the native database on Android, it works different to traditional SQL databases in that the database runs on the same machine as the program, rather than sending requests over a network they stay on the physical device (McClure, 2012). This makes it very lightweight, and efficient with the potential to upgrade to a stronger and more powerful SQL with ease. It is designed to work with custom defined objects so items like `templateWorkout` or `User` require no configuration of extra work to store in the database.


```
var db = new SQLiteConnection(dbPath);
db.CreateTable<Workout>();
if (db.Table<Workout>().Count() != 0) // Is there saved workouts??
{
    var table = db.Table<Workout>();
    foreach (var s in table) // yep load them in
    { Workouts.Add(s); }
}
```

Figure 19: Code view of SQLite retrieving data

The `templateWorkouts` table in the SQLite database needed to be populated with premade workouts but its not possible to have it populated when the application is first installed onto

```
Beginner stage 3
Beginner
cardio
3
300
150
4

Beginner stage 4
Beginner
cardio
4
420
180
3
```



```
using (StreamReader sr = new StreamReader(Assets.Open("premadeworkout.txt")))
{
    int counter = 0;
    do
    {
        counter += 1;
        workoutTemplate temp = new workoutTemplate();
        temp.name = sr.ReadLine();
        content = sr.ReadLine();
        IF (content == "Beginner")
            temp.level = Running_Experience.beginner;
        IF (content == "Intermediate")
            temp.level = Running_Experience.intermediate;
        IF (content == "Advanced")
            temp.level = Running_Experience.advanced;
        content = sr.ReadLine();
        IF (content == "cardio")
            temp.pref = Running_preference.long_distance;
        IF (content == "HIT")
            temp.pref = Running_preference.high_intensity;
        content = sr.ReadLine();
        temp.stage = int.Parse(content);
        content = sr.ReadLine();
        temp.runSec = int.Parse(content);
        content = sr.ReadLine();
        temp.walkRestSec = int.Parse(content);
        content = sr.ReadLine();
        temp.cyles = int.Parse(content);
        end = sr.ReadLine();
        temp.id = counter;
        suggestedworkouts.Add(temp);
    }
}
```

Figure 20: Text version of template workouts(left) and code view of loading in the text file to populate objects (right)

the device. In order to populate it before the user accesses it, a txt document is included in the project assets holding all the details of the template workouts. When the application is run for the first time and no template workouts are detected in the SQLite database, it reads from the text document line by line populating the 7 fields of the `workoutTemplate` objects which each 7 line section (figure 20).

Using SQLite made it easy to retrieve specific set of data, for example in the view all suggested workouts page, it was possible to separate the large database of workouts into manageable chunks to present to the user by filtering the results by workout style and difficulty (*figure 21*). The filter was applied by setting a WHERE clause in the SQL request for both the difficulty level and style.



Figure 21: Screen shot of template workouts - full selection

4.2.6 Graph

In order to present the user's progress in a meaningful way, a graph was required. To do this a component was installed called OxyPlot. Similar to the Google Map component, it is assigned an object in the XML interface which is then linked to the custom graph object in the activity code (*figure 22*).

```
<oxyplot.xamarin.android.PlotView
    android:layout_width="fill_parent"
    android:layout_height="150dp"
    android:id="@+id/graphViewMode"
    android:layout_marginTop="20dp"
    android:layout_marginBottom="20dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp" />
```

Figure 22: OxyPlot implemented in XML

In order to populate the graph with data, a series is designed and populated which can then be assigned to the graph. The series object describes the colour, thickness and style of the line of the graph as well as holding all of the X and Y data points for the graph. Populating the series data points is a simple matter of looping through the array workouts and choosing to assign a variable of the workout to the data point.

To provide the user with more graph options the three main variables of the workout; distance, speed and average speed were set as options and the user can switch between them with the press of a button. All of the variables are paired with DateTime as the X-axis because the page is designed to show how the user is progressing over time (*figure 23*).



Figure 23: Screen capture of OxyPlot graph

4.2.7 Workout Tracking

The main function of the application is the tracking of workouts through GPS. The functions of the GPS and how speed and distance are calculated have been explained in section 4.2.4.

When following a suggested workout, the user uses the application to time how long they should be running for and how long they should be resting or walking rather than by distance, this requires a clock system. Provided in System.Threading.Tasks is a Timer class, this class is provided with a method to call and a time interval to call the method at. As all of the template timers are set in seconds the interval is set for every 1000 milliseconds or 1 second.

The timer is assigned a reference because otherwise it is subject to garbage collection. Using a global reference also allows us to easily access the timer in other methods and dispose of it once there is no more use for it. As the timer is used largely to update the UI, it is specified to run on the UI thread meaning it can interact with it.

Suggested workouts switch between running or resting/walking, users need to keep track of which mode they are currently in. Looking at a text box to check which mode they're in isn't very practical, especially in HIT workouts where modes last 15-60 seconds. To counter this a larger visual cue was made to signify the mode the user is currently in, the most obvious way to display this is with changing the background of the page between blue and red, this way users can glance at their phone and see the colour of the background rather than trying to read a textbox (*figure 24 & 25*). User's on the run could possibly not want to check their phone at all so in addition to the visual cue an audio cue was included. Every time the mode switching between running and resting a simple beep noise is made, as long as the user has sound on it can be heard, even when listening to other audio such as music.

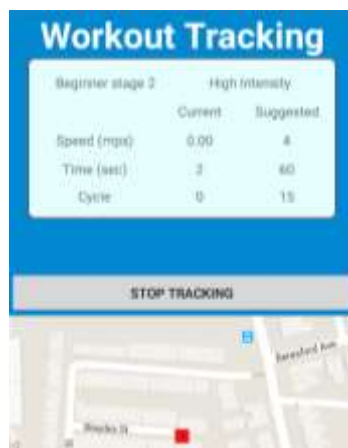


Figure 24: Workout tracking blue (walk/rest phase)

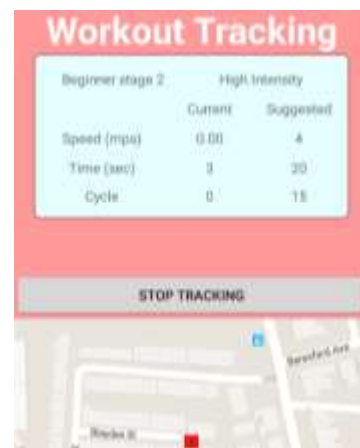


Figure 25: Workout tracking red (running phase)

4.2.8 Calculating user's ability

The original algorithm designed to compare the running speed of the user to the suggested running speed every cycle was implemented but it was quickly discovered it did not effectively because it had a bias to workouts with more cycles. As if a user was either ahead of their target speed or below this would be multiplied by cycles completed and some workouts have one cycle compared to others having up to 15.

A simple fix to this was for to track the average speed of the user while in running phase over the whole workout and compare this to the target speed.

At the end of the workout, the user has a pop-up notice informing them if they were on target, lower or higher and by how much their suggested workouts will change (*figure 26*). As the target speeds for jogging and running are simple numbers it would provide a lot more optimisation if the user were able to customise the targets. Two drag bars were installed in the details page allowing the user to customised the target speeds for jogging and running.

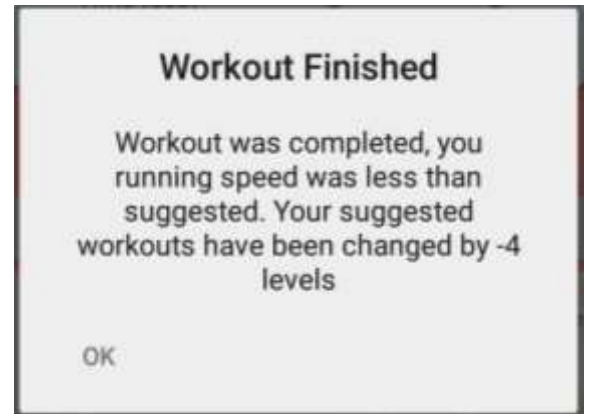


Figure 26: Screenshot of workout finished popup

5 Testing

5.1 Functional Testing

5.1.1 Function Test Table

The main form of testing used in the project is function testing; this focuses on defining tasks and actions for users to perform and requiring them as criteria for the program functioning as designed and intended. This simulates the system in the way the users will experience the system, uncovering any bugs or issues they would be likely to encounter (Black, 2011). The designs created previously such as the task list and the user journey can be broken down to form the functions required of the user, these functions need to be quantifiable in terms of working or not working.

With a project like Run Bud, white box methods such as unit tests are often not suitable due to the technology involved, for example creating unit tests for the GPS functionality wouldn't be suitable as the environment it would be setup in with virtual device and scripted geolocation updates wouldn't replicate issues the problems users would encounter in a realistic situation with their smartphone.

ID	Name	Functionality	Notes
1	Track the workout of the user		
1.1	Use phone GPS	Working	More testing/analysis in (section 5.2 & 5.3)
1.2	Display map	Working	
1.3	Start tracking	Working	
1.4	Display performance information	Working	More testing/analysis in (section 5.4)
1.5	Stop & save performance	Working	
2	Track workout (With a template workout)		
2.1	Display suggested workout	Working	
2.2	Suggested workouts match user	Working	
2.3	Display all suggested workouts	Working	(share function 3.4)
2.4	Select a workout	Working	
2.5	Start a selected workout	Working	
2.6	Display performance against suggested	Working	
2.7	Stop workout early (on request)	Working	
2.8	Stop after suggested finished	Working	
2.9	Analyse performance & display	Working	
2.10	Change user's suggested workouts	Working	
3	Data Storage		
3.1	Save new user	Working	
3.2	Change user details & save	Working	(share function 5)
3.3	View previous workouts	Working	(share function 4.3)
3.4	View list of suggested workouts	Working	(share function 2.3)
3.5	Filter list of suggested workouts	Working	

4	Progress		More testing/analysis in (section 5.5)
4.1	View graph, presenting progress	Working	
4.2	Customize graph	Working	
4.3	View workout history	Working	(share function 3.3)
5	User Details		
5.1	Edit user details	Working	(share function 6
5.2	Edit running speed	Working	
5.3	Save user details	Working	(share function 3.2)
6	User Interface		
6.1	Textbox inputs (valid inputs)	Working	
6.2	Buttons controls	Working	
6.3	Spinner controls	Working	
6.4	Slider control	Working	

5.1.2 Generated Test Data

When testing the device, it is important for it to behave as it would with a user in order to find the same situations a potential user would be in.

When compiling and launching the application, it is a fresh install so none of the previous workout history or user history is there and during development the application is being recompiled and reinstalled several times a day. This does not give a very good representation of how the user would be interacting with the application because they would be expected to have some user details and a workout history.

For use in development 3 buttons were installed on the home page called testing 1,2 and 3. The first testing button generates a workout history. It creates a customizable amount of previous workouts, replicating a typical history of a user after several days of months of use. This has been automated by setting each workout variable to be randomly assigned a value in the expected range, this is known as partial coverage using domain based criteria and is a simple method of partial coverage where exhaustive coverage is not an option (Bottaci, 2015). Looping through this many times generates any array of results which cover all the expected values which would be expected to be generated by normal users. This method of testing is one of the most efficient; "because they operate automatically, they can be used to generate arbitrarily large volumes of data and provide arbitrary levels of test throughput at relatively little cost and risk" (Ali Mili, 2015).

This method was vital in the development and testing of the progress page for the application. As shown in figure 28 generated data was required to populate the page and test the functionality of the graph and workout history list.



Figure 28: Screen shot of progress page with test data

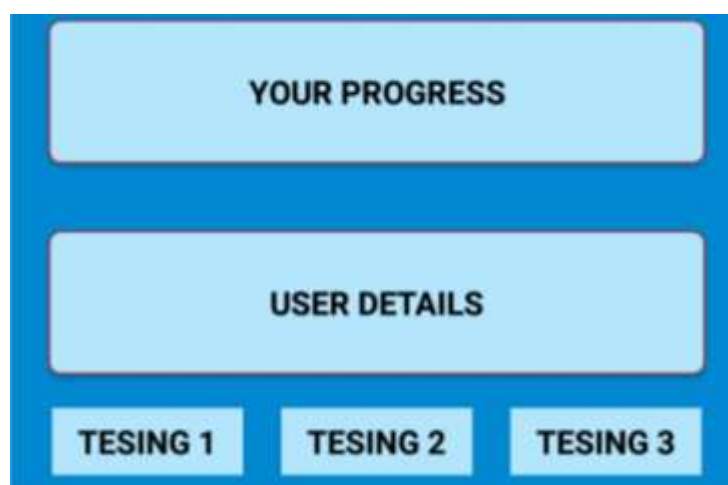


Figure 27: Screen shot of test buttons

The second and third buttons were used to generate and use template workouts, this is because the template workouts range from 20 minutes up to 2 hours and there are many functions to test when the workout the workout ends such as calculation of performance, updating of user progress and the finishing of the activity and return to home screen. Using this system meant that workouts could be started and finished in under 15 seconds and the process at the end of the workout could be tested and debugged over and over.

5.2 GPS Testing

5.2.1 Introduction

GPS works by locking on to the signal of a minimum of 3 satellites orbiting the earth and use the time between sending and receiving radio signals from the satellites to calculate the receiver's position on earth, this process is known as trilateration. There are currently 24 satellites orbiting the earth placed into orbit in 1973 by the U.S. Department of Defense which cover the earth meaning theoretically you can be in sight of 3 satellites at all times (Md. Palash Uddin, 2013).

The accuracy of the GPS is a vital part of the application as the recording of the distances and speeds are done via it. On the first testing session of Run Bud, the time distance between GPS location results was set to be 10 seconds (figure 29).

This was not showing an accurate path of travel on the map and would also result in inaccurate measurement of distance traveled and speed. Adjustment of GPS location updates was changed to update every 2 seconds in order to get a more constant reading of GPS locations for distance/speed calculations and also to present locations accurately on the map (figure 30).

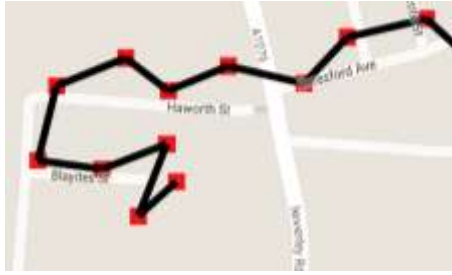


Figure 29: Screen capture of GPS recording with default value

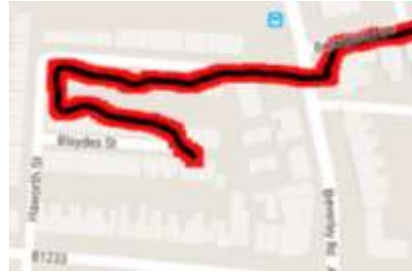


Figure 30: Screen capture of GPS recording with slight optimizations

The Google location API has other configurations which could enable the GPS to work more efficiently or more suited to the task at hand. The power consumption is a large factor of the GPS accuracy, it can be in 4 different power use modes; high, balanced, low and off. Low and off are not applicable for the application because it relies heavily on GPS information and they are too inaccurate. High would be expected to be the best option but users could want a longer battery life when using the application and if balanced power use gives the same or similar accuracy then it may be more suitable.

“Care must be taken to ensure that the power and energy behavior of modern smartphones are in line with expectations” (Alam, 2014)

To test the optimal configuration of the GPS several variables have been selected. The GPS will be tested with several different settings altering the power use of the GPS, the smallest displacement and by setting lower accuracy updates to be ignored.

Method 1	High power priority on the location requests.
Method 2	High power priority on the location requests and smallest displacement set to 5 meters.
Method 3	High power priority on the location requests and accuracy barrier on incoming location updates
Method 4	Balanced power priority on the location requests

The exercise the application is designed for could be used in different environments, although it is always expected to be used outdoors it may be through busy streets surrounded by houses and other people or in a field far from any other objects or routers. In order to find the best method, each will be tested in on different running routes in order to find the method that works best in all situations.

5.2.2 Deciding the accuracy limit

In order to get a more accurate reading of the user location, some of the GPS results should be filtered out. To decide what the accuracy limit should be set in order to filter out a reasonable amount of results an array of accuracy estimates have been gathered for analysis.

Analysis of several tests with the GPS recording showed that with over 200 location updates the average accuracy with the results was 20meter. please note that the accuracy reading attached to the location update is the maximum variation possible determined but not how far the result is actually off (see Appendix B for a list of results).

```

float accuracy = location.Accuracy;
accavr += accuracy;
float accuracyLimit = 15;
if (accuracy < accuracyLimit) // only updates if gps is accurate by at least 15m
{

```

Figure 31: Code view of accuracy filter

Using 15 meters as an accuracy limit should filter the results and make the GPS tracking over the trip more reliable and measure distance/speed accurately (*figure 31*). Notably another paper had a similar reading for unoptimised average accuracy; “From the point of accuracy of the position data, 16metererror in maximum was the best value in our system” (Hideo Makino, 1996), supporting the validity of these results.

5.2.3 100-meter & 200-meter Test

This test will be a short distance test, 100 and 200 meters in a field in order to test a realistic running scenario in a field with no interference, or large buildings around which could affect the accuracy of the GPS (see Appendix C).

Evaluation

The results were closer than expected, the high power mode with no other modifications calculated the distance to be slightly further by 6 meters on the 100-meter and by 2 meters on the 200-meter trip.

The high power mode with 5-meter displacement modifier was very close to the actual distance, just a few meters under the actual on both the 100-meter and 200-meter. The reason it is just under can be explained because the last GPS update received between 95 meter and 100 meter will not be replaced by a new one unless they travel more than 5 meters away from it. This effect is even more noticeable on the 200-meter trip because the route involves turning around and so the distance between the last GPS point recorded and where the user walks past that is doubled. In a longer running session involving lots of turns this effect would be amplified.

High power with accuracy limit was 5 meters off in both tests, this means it is recording more accurately than the high power mode with no other modifiers but only by slightly, it could be beneficial to try and further this method of accuracy filtering,

Balanced power mode as expected was the least accurate in measuring distance, the 100-meter result was 11 meters off and the 200-meter result was 15 meters off. Not only are the results the furthest off the actual but they are an inconsistent error from the actual distance meaning they would be hard to normalize with algorithms.

Conclusion

High power with 5-meter displacement has a positive impact on the accuracy of a session but only when travelling in a straight line, when turning with the GPS it may not register a large part of the distance travelled. The diagram (*figure 32*) illustrates how this method may record a turn, black representing the real journey and red representing how the GPS records it.

High power with an accuracy limit had a slight advantage over the normal high power method with more consistent values, a potential problem with this method is that accuracy results change a lot depending on time, location and phone type or quality. So although this method worked well and improved results, it would not translate well to real world situations where accuracy cannot be filtered with a static number.

Balanced power mode as expected was the least accurate of the methods and not a practical option.

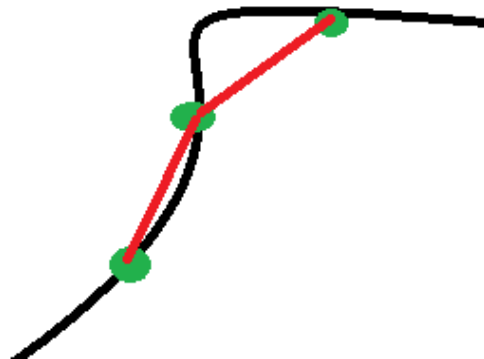


Figure 32: Mockup of how GPS records compared to actual movement

5.3 Further Optimization

Some basic power options offered by the Google API have been optimized and basic filtering of the results performed, this next section will explore what more can be optimized to get the smoothest and most accurate path of travel recorded. Rather than process more data, the data should be optimized before being processed as a location for the best efficiency (Hung, 2013). Both of the following methods focus on feeding less information through to be processed as a location meaning distance, speed and the map will be recalculated, but on filtering out bad results and averaging results resulting in less total being processed.

5.3.1 Accuracy Decay

Accuracy can vary over time and location; this would mean using a single value to determine whether a location is accurate would not be appropriate because in some situations it could filter out 90% of your location updates but in another situation only filter out 10%. The accuracy requirement on incoming location updates should vary in order to filter out 50% of the updates in all situations. The accuracy limit is calculated by averaging the previous 10 updates received, this means if a user changes location such as moving from a busy street into a field where GPS results are more accurate the limit will change to the new average progressively over 10 seconds.

If the user were to move from a high accuracy location to a low accuracy location the average would change over 10 seconds but it could possibly mean the user receives no updates for 5-15 seconds affecting their distance and speed estimates. In order to counter this, a decay rate is introduced. This increases the accuracy limit to allow more results through every time a result is rejected due to low accuracy and resets back to nothing every time a result is accepted. When several results are rejected in a row the accuracy limit is raised faster making less time between a user receiving results in an accuracy transition period.

```

float total = 0;
foreach (float item in accuracys)
{
    if (item != 0)
    {
        count += 1;
        total += item;
    }
}
float limit = total / count;
limit += decay;
if (accuracy <= limit)
{
    averageacc += accuracy;
    counter = counter + 1;
    avr = averageacc / counter;
    LatLng templating = new LatLng(location.Latitude, location.Longitude);
    useLocation(templating);
    decay = 0;
}

```

Figure 33: Code view of accuracy decay algorithm:

5.3.2 3 Point Average

With GPS data there is often noise, this is small interference which causes the GPS to be slightly off the real location and so less accurate. In an effort to get a smoother path GPS location updates can be averaged. This mean waiting for 3 location updates, totaling their latitudes and longitudes and then dividing them by 3 to get an average location from the three.

In a situation where the noise is truly random meaning the offset from the real location is random in direction and not weighted. For example, if a user were standing next to a cliff the results would likely estimate the user to be further away from the cliff rather than closer due to the interference caused by it. For our situation, the noise is expected to be random and so this method is applicable to improve accuracy (Vavra, 2010).

5.3.3 100 meter & 1KM Test - Optimized methods

This test will be an attempt to replicate more realistic running conditions in order to get results which can be analysed in such a context. The first test will be a simple 100-meter route following a track in a park with slight turns and minimal interference (trees). The second route will transition from city to field or vice versa over a 1000-meter route and involve normal situations which could affect the tracking such as waiting at a traffic light and tight turns building corners (see Appendix C).

Evaluation

The accuracy decay method had a similar precision to the normal high power mode without modifiers on the 100-meter test. The 1000-meter test slightly overran and ended up traveling 1030 meters, with the method reading 1034 meters it is only 4 meters or 0.38% different from the actual.

Using a 3-point average had a large effect on the distance recorded for 100M, it made it short by 10 meters or by 10.41%, this can be explained because the GPS doesn't start recording distance till it has 3 results it can average, this would cause several meters at the start of the route to be ignored, additionally if 1 or 2 GPS updates come in before the user hits stop then they will be ignored because there are not 3 to average and process as a location. This causes several meters to be lost in every trip. The 1000-meter route was 28 meter off the actual distance or 2.8% which is more accurate than the 100-meter test, this is

because the 1-3 GPS updates ignored have less impact with larger total amounts GPS results in the session.

Conclusion

Both methods had an effect on the accuracy of the distance recording, the 3-point method had the worst accuracy with the 100-meter route being 10 meters short, almost as far off in distance as the balanced power method. It makes up for this as with longer distances it becomes a lot more accurate moving from 10.41% to 2.8% with 10 times the distance. Accuracy decay method was more accurate in both tests, in the long distance test it performed outstandingly being only 0.38% off the actual distance, similar to 3-point its 100-meter test was not as accurate but this can be considered a positive as over longer time and distance we want the % difference between actual and recorded to get smaller.

5.4 User Testing

5.4.1 Introduction

In order to test the usability of the application, someone other than the developer would be required as commonly the developer of an application tends to avoid triggering bugs or designs the application in a way that is clear and visually appealing to them but may not be for the general population. “A user test is successful if it uncovers problems. Be aware of the normal tendency of falling in love with your own creations.” (Lehtimäki, 2013)

User testing involves testing with the types of people who typically would use your product.

The test sheet used asks the user to go follow basic instructions, guiding them through the general process of making a new account, starting a workout, starting a template workout and finally reviewing their account progress. The instructions are loose in order to allow them to make mistakes and use the application in the way a new user would. This is important in discovering flaws not yet noticed by the programmer.

After completing the short instruction set the user fills out a short questionnaire consisting of 15 attributes or sections of the application to be rated from 1 to 5. The questions are split into groups of each section of the application and for the general interface, with each section there is indicated space to add extra comments or any problems encountered with that area of the application (see Appendix D for test sheets).

5.4.2 Results

All user sheets were collected and each one was assigned a number, all of the user feedback was entered into an excel spreadsheet for easier analysis, Excel data can be found in Appendix J. In order to analyse the results better averages were calculated, for each component (separated by colour) and for each question (separated by column), this helps us to identify sections that need improvement and where the application has succeeded.

In order to analyse the results better averages were calculated, for each component (separated by colour) and for each question (separated by column), this helps us to identify sections that need improvement and where the application has succeeded.

In order to keep all tester's feedback anonymous, the number the test sheet assigned is used rather than tester's names in accordance with section 12 of the student ethics checklist: “All the data collected from the participants will be stored in an anonymous form”. If it is necessary to identify a particular tester perhaps to question them further or for justification on their opinions the test sheet can be referred back to where the number of the test sheet is in addition to the dated name and signature of the tester.

For presentation and analysis purposes extracts of information have been placed into graphs; section graph, presenting the results of each question from the user testing and component graph, presenting the analysis of the results as the average score for each component and standard deviation of sections within the component.

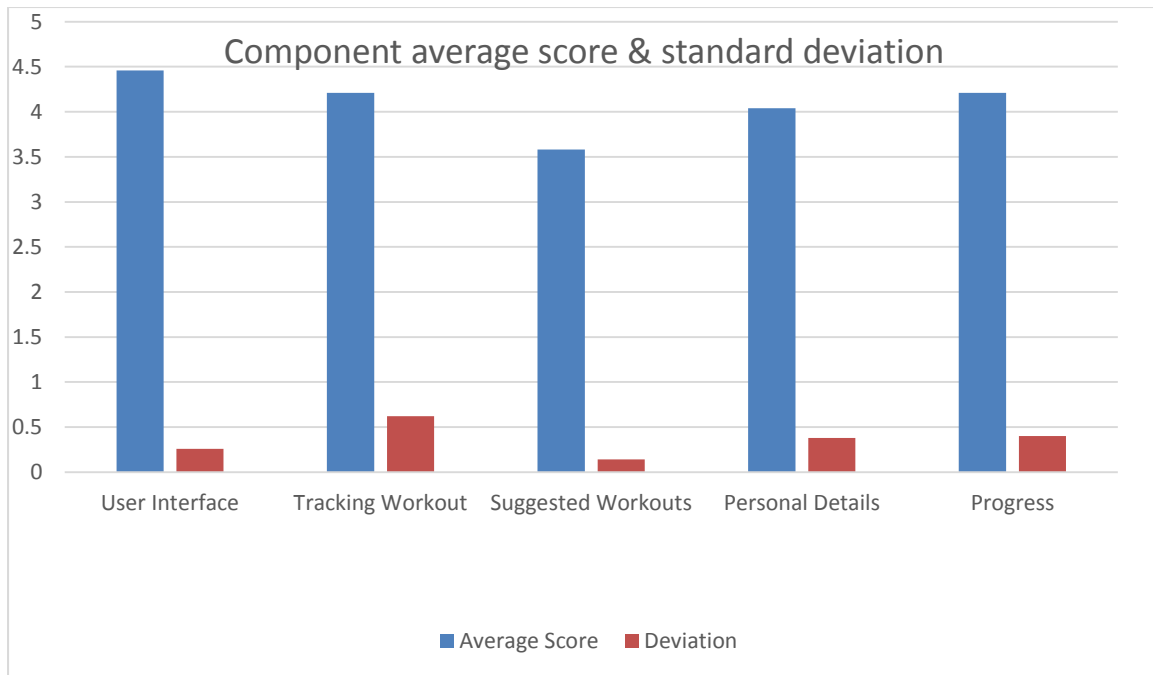
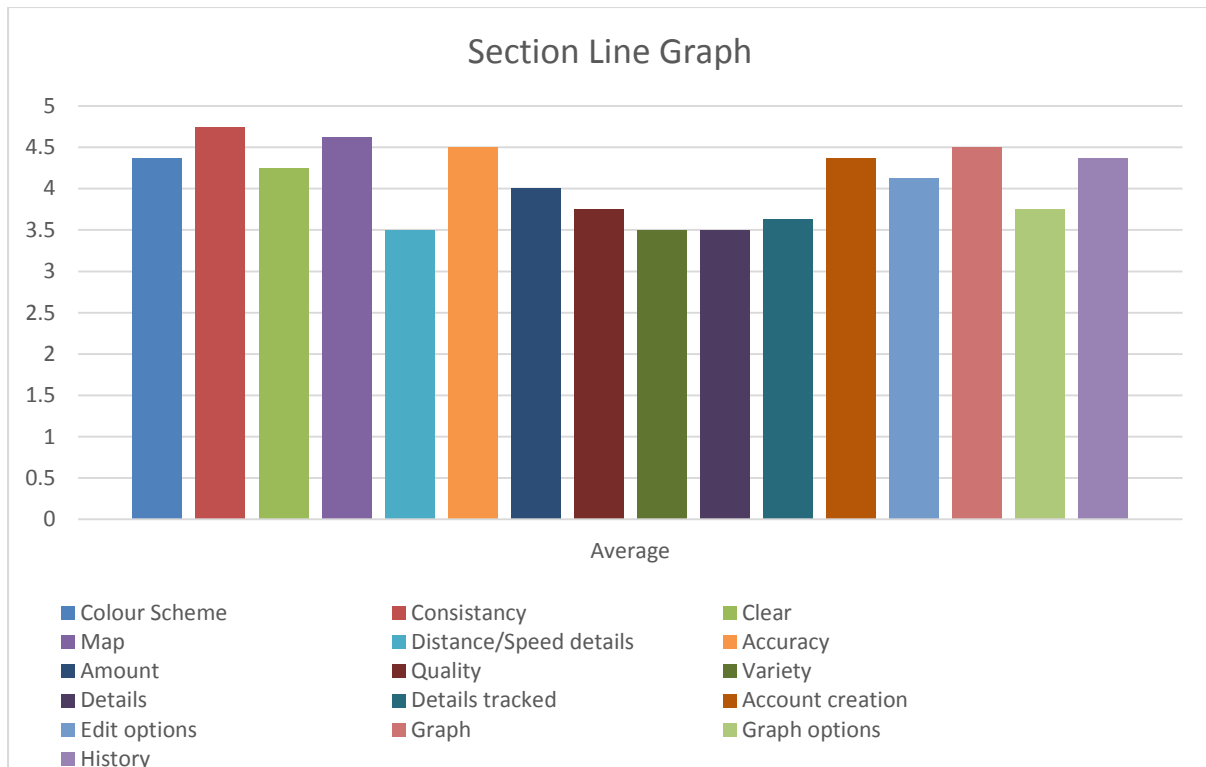


Figure 34: Component Line Graph, User testing results

5.4.3 Evaluation

The lowest scoring component of the application is the suggested workouts with an average score of 3.58. Looking more closely at the component, the highest scoring section of the component was the amount of suggested workouts with an average of 4 and the lowest scoring sections variety and details tying with 3.5, only a 0.5 difference between the largest and lowest scores in the component and a 0.14 standard deviation show that it is not a single fault or issue with one section bringing the whole component down but that the component as a whole is not performing as well as the rest of the application. Keeping in mind that suggested workouts component still scored 71% which is acceptable, but it doesn't match up to the other components scoring 80% and above, this suggests component could require a rework.

The highest scoring component is the user interface with an average of 4.45, in particular, the consistency of the interface was rated highly, this can be attributed to the colour scheme decided in the interface design (chapter 4.1.1) and to the themes designed and used in the interface implementation (chapter 4.2.3).



Some of the other outliers in section scores are details tracked from the personal details component with a score of 3.62 and distance/speed details from the tracking workout component with an average score of 3.5. Some of the extra information added by the testers can help to shed light on why these areas, in particular, did not score as highly. One of the users wrote a comment saying “wasn’t sure what the speed was measured in” with another user making a similar comment it is clear there is a problem with the display metric which is confusing users. A user commented on the personal details component writing “why is weight being tracked if there is no history of it”, in addition to this age is also being tracked but has no real use within the application.

5.4.4 Conclusion

The user testing overall was a success in that the application averaged a respectable score of 4.1 or 82%. It also revealed some issues with the application which can now be fixed and improve the overall experience with the application. The first and more obvious fixable fault is the speed display; this can be changed in order to be clearer that the display is in meters per second. User details could also be updated to reflect the use of the application more accurately by no longer having an entry for weight or age.

6 Evaluation

6.1 Objectives

In order to evaluate the Run Bud project as a whole, each objective stated in chapter 2 will be reviewed individually and then as a collective. Then the management styles of the project will be evaluated such as the Agile development methodology and the time planning throughout the year. Finally, some ideas for expansion of Run Bud or redesign of the current system as future work will be explained and reviewed.

Objective 1 - Develop Fitness smartphone application

The application is successful as both a fitness based application, focused on running/jogging workouts and also a smartphone application, running on Android. A simple objective but also a core one for the project in achieving what was set out from the start. This objective has been successfully completed.

Objective 2 – Track the workout of the user

This objective has been very successful, the user is able to track workouts, showing their location with GPS on a map as well as their speed and distance traveled. GPS tracking was tested and optimized in order to display the most accurate path on the map and to track the distance and speed optimally.

Also included in this objective is was that users will be able to add their own notes to workouts, this has not been included as in the development of the application it seemed that feature would be redundant.

Objective 3 - Track the progress of the user

The progress tracking has been implemented but not quite as first envisioned. This component is now largely based on the progress of the user athletically rather than their personal health. Throughout development, it became clearer that the application was a fitness application rather than personal health. Upon user testing, it was discovered that user's weight, height and age didn't have any use within the application and so it should only be tracking physical characteristics so it has been changed to suit this.

All workout history is stored, meaning users get a clear view of their progress in several physical aspects including their distance and speed, users also get to decide their own progress when using the premade workouts, being able to move up or down in levels of difficulty of the workouts.

Objective 4 – Offer different types of workouts for the user customized for them

The application has succeeded in this objective; it has a large range of over 45 premade workouts of either high intensity or cardio based workouts. The workouts are suggested to the user based on their preferences they set in account creation or in user details. They are also suggested to users based on their performance in previous workouts, their performance is based on how behind or ahead of the suggested speed. The suggested speed can also be customized in user details making the premade workouts and the progression system within it completely customizable to the customer.

Objective 5 – Personalized data display/graphs

Tables and graphs have both been implemented in the user progress page. All the information automatically recorded about users is from their workouts, this is used to chart their progress. The objective details that the information should be presented in different forms; the progress page displays workout history as a list in a table and then as a graph, plotting selected variables from the workout history over time. Although it only supplies workout history to be presented and not personal details such as weight, this objective

Objective 6 – Graphical User Interface

The interface was designed and implemented very successfully as it connects of the sections of the application and presents them as specified in the objective. It also scored the highest on user feedback out of all the sections of the application with 89% positive feedback suggesting it may have been the most successful part of the application.

Secondary Objective 1 – Personal accounts for users and server to store all data

SQLite was used for the data storage throughout the application, this allowed for potential upgrade to using SQL with a server. This would mean that users could have their account information and workout history stored online and would be able to download all of their information onto any phone that they decide to use.

Time limitations mean that an upgrade to SQL and setup of a server was not able to happen and this objective was not achieved.

Secondary Objective 2 – Reward System

The framework for this objective was setup as methods to track user progress were in place. Due to time limitations, other primary objectives were prioritized and this objective was mostly incomplete.

Objectives summary

All of the primary objectives were fully met or mostly met so, for this reason, I would consider the objectives achieved. Small changes realized in implementation meant that objective 3 and 5 could not be completed in the way they were initially stated. Both secondary objectives while feasible were left uncompleted due to time constraints.

6.2 Management Review

The time plan, for the most part, was followed, after being revised mid-way through the project in order to allow slightly more time for some of the key components encountering problems and to revise the GUI task to run throughout the development time rather than have a secluded time period to develop it.

Testing was started earlier than the secondary objectives because after review it seemed more important to make sure the application was working correctly and was suited to users before including more features. Several bugs were discovered after testing as well as small changes to the program requiring work and not allowing enough time for the secondary objectives.

The first half the time plan was followed very well and was successful in managing the order and time limit for each component to be installed. The second half could have been

organized due to testing overrunning schedule. Perhaps it would have been better to plan for two testing sessions, one midway through development to optimize or discover faults with code already written and revise goals or designs based on user feedback and another towards the end of development.

The agile development methodology was very effective with this project, implementing each section at a time made it much easier to focus concentration

6.3 Further Work

6.3.1 Template workout rework

The suggested workout component was the lowest scoring component in the user testing, to improve the template workouts the class would have to be remade. This would also mean the way template workouts are read by the GPS activity would have to be remade.

The benefit of this is that more types of workouts would be available with new customization, for example, the lengths or speed of running phases could ramp up throughout the workout or sections of increased difficulty like in hill interval training.

Another feature which would be great to include is the ability to record a workout route. For example, a hiking trail would be great for a user to have as an option to follow but other simple options could be a nearby field with a path or a quiet route around some suburbs. It is likely that crowdsourcing would be required for this so that routes could be uploaded everywhere, a single person or even a group would likely not be able to create enough routes to satisfy everyone in England never mind people all around the world.

“Smartphones can reveal crowdsourcing’s full potential and let users transparently contribute to complex and novel problem solving” (Chatzimilioudis, 2012)

6.3.2 User creation of template workouts

Providing the ability for users to design and create their own templates would be very powerful. Not only could workouts be made by users for themselves but they could also be uploaded to a server where other users would be able to browse and download template workouts.

With an online database of workouts, users could rate workouts in order to develop a list over time where the best workouts are at the top and new users can quickly select ideal workouts voted for by all the other users.

6.3.3 Tracking of user health

Originally the application was designed to track user weight, age and body fat percent. As development continued it became apparent these were not very relevant to the core of the application and were removed. For future expansion, it would be good to implement a component for tracking the user’s health which would include food tracking. This component could calculate user’s intake of calories and amount of nutrients as well as offering plans to work with in order to reach their nutrition goals whether that be losing or gaining weight or even types of diet such as paleo diet. Mixing this with the exercise component could turn it into a fully-fledged health application where performance in the health section could affect the exercise plans, for example, planning for extra exercise when a user eats over their calorie target.

7 Conclusion

This original aim of the project first devised is as follows:

The aim of the project is to produce a smartphone application specifically for running/jogging. The application will focus largely on the GPS and the information gathered, this will be used to track user's workouts. With the information gathered, template workouts will be suggested to the user, relevant to their ability level and style of running. The application needs to be lightweight and easy to understand so that users not experienced with either fitness or technology will be able to use it.

I believe the aim of this project has been mostly fulfilled. The application has succeeded in focusing on the GPS in order to generate information about the user's movements and using that information suggest new workouts and the interface is simple and clear for unexperienced users. I think there is a lot of room for improvements for the application including making it server based which was a secondary objective left unachieved but in its current form it is a usable product.

I feel I have learned a lot over the course of the project, having never programmed for Android before it was a daunting task to plan, design and then code for it. Just some of the technologies, standards and techniques learned or improved over the project are listed below

- Xamarin
- XML
- SQLite
- Android development
- AVD/SDK management
- Google Play services
- OxyPlot
- Json
- GPS
- Time planning/management
- AGILE development
- Multi-threading
- Algorithm design & analysis
- Report writing

I have never committed to a project of this size or length before and feel it has given me a much better understanding of software development and my future in which larger projects are expected to come. The volume of work to complete, although very daunting gave me an insight to an industry where programmers work on large scale project but given sections of work to do at a time and may not have much context around it meaning it must be developed to specification in order to fit into the system correctly. Through this, I feel I have had a massive learning experience and am a lot more prepared for industry work following this project.

The project as a whole I feel has been a success, Run Bud is a fully functional smartphone application making use of the phone specific hardware capabilities and providing a simple and clear application as evidenced by the user testing.

Appendix A: UI Design


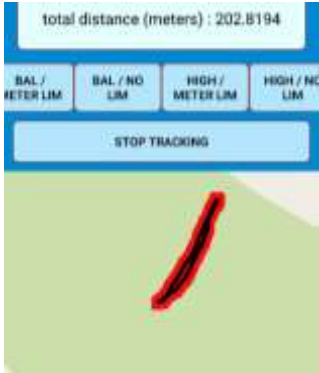




Login	Home	Suggest Workout	All Suggested																
RUN BUD Tracker and Planner Username: <input type="text"/> Password: <input type="password"/> LOGIN	RUN BUD Tracker and Planner Track Workout Suggested Workout Your Progress User Details Log Out	RUN BUD Tracker and Planner Based on your details recorded in the application we recommend you choose one of the workouts offered below. Click on a workout to show more details. Intermediate : 10h Intermediate : 12h Intermediate : HIT List of all workouts Back	RUN BUD Tracker and Planner Beginner : Week 1 Beginner : Week 2 Beginner : Week 3 Beginner : Week 4 Beginner : Week 5 Back																
Suggested Details	Track Workout	Track/Suggested	Progress																
RUN BUD Tracker and Planner Intermediate : 10h A longer but slower pace run, designed for improving cardio fitness. Run Details Distance: Walk - 30MPS - 10 Run - 30MPS - 10 Walk - 30MPS - 10 Run - 30MPS - 10 Walk - 30MPS - 10 Run - 30MPS - 10 Start Workout Back	 Start/Stop Tracking Distance: <input type="text"/> Speed: <input type="text"/> Avg Speed: <input type="text"/> Time: <input type="text"/> Back	 Start/Stop Tracking Current Aim Distance: <input type="text"/> <input type="text"/> Speed: <input type="text"/> <input type="text"/> Time: <input type="text"/> <input type="text"/> Workout: <input type="text"/> Speed: <input type="text"/> For: <input type="text"/> Back	RUN BUD Tracker and Planner Average Speed Weight Distance Previous Workouts Back																
Graph	Change Details																		
RUN BUD Tracker and Planner Average Speed <table border="1"> <thead> <tr> <th>Time</th> <th>Speed</th> </tr> </thead> <tbody> <tr> <td>0:00</td> <td>0.00</td> </tr> <tr> <td>0:10</td> <td>0.10</td> </tr> <tr> <td>0:20</td> <td>0.20</td> </tr> <tr> <td>0:30</td> <td>0.30</td> </tr> <tr> <td>0:40</td> <td>0.40</td> </tr> <tr> <td>0:50</td> <td>0.50</td> </tr> <tr> <td>1:00</td> <td>0.60</td> </tr> </tbody> </table> Back	Time	Speed	0:00	0.00	0:10	0.10	0:20	0.20	0:30	0.30	0:40	0.40	0:50	0.50	1:00	0.60	RUN BUD Tracker and Planner User Details Name: <input type="text"/> Age: <input type="text"/> Gender: <input type="text"/> <input type="text"/> Speed: <input type="text"/> Experience: <input type="text"/> <input type="text"/> Goal: <input type="text"/> <input type="text"/> Save Back		
Time	Speed																		
0:00	0.00																		
0:10	0.10																		
0:20	0.20																		
0:30	0.30																		
0:40	0.40																		
0:50	0.50																		
1:00	0.60																		



Appendix B: Accuracy Results (excerpt)

38	Accuracy = 14 Running average = 18.6849
39	Accuracy = 13 Running average = 18.5712
40	Accuracy = 14 Running average = 18.48157
41	Accuracy = 18 Running average = 18.47231
42	Accuracy = 18 Running average = 18.4634
43	Accuracy = 19 Running average = 18.47333
44	Accuracy = 18 Running average = 18.46473
45	Accuracy = 33.867 Running average = 18.73977
46	Accuracy = 16 Running average = 18.6917
47	Accuracy = 18 Running average = 18.67978
48	Accuracy = 24 Running average = 18.76995
49	Accuracy = 31 Running average = 18.97378





Appendix C: Distance Testing

100M & 200M test

Method High P	Distance 106.46 M	Map 	Distance 202.81 M	
High P/ 5m Displacement	98.68 M		196.44 M	
High P/ Accuracy Limit				

Balanced	111.56 M		215.04 M	
----------	----------	---	----------	---

100M & 1000M Test (Further optimization methods)

Method	Distance			
Accuracy Decay	100M Actual: 105.6M		1030M Actual: 1034M	
3 point	100M Actual: 90.4M		1000M Actual: 1028.5M	

Appendix D: User Testing Sheet

Run Bud: User Testing

Run Bud is a smartphone software application designed to track and record running workouts as well as suggesting template workouts to the user based on their performances.

The testing procedure is expected to take between 10 and 20 minutes and will take place in a field, track or other suitable running location.

Please follow the steps listed below in your own time. If you are unable to complete a task feel free to skip the task and note, why you were unable to complete the task. This is not a measure of your skill but a measure of the application, take your time completing the steps and interact with the application as you would expect a fully developed and released application to act.

1 New user

- 1.1 Start Application
- 1.2 Enter personal details & Create new user
- 1.3 Edit personal details & preferences

2 First Workout

- 2.1 Start a new workout
- 2.2 Travel 150 meters
- 2.3 Finish workout

3 Suggested Workout

- 3.1 Find a High Intensity template workout
- 3.2 Start the template workout
- 3.3 Perform the workout (don't have to do entire workout 2/3 mins will suffice)

4 Progress

- 4.1 Navigate to the Progress interface
- 4.2 compare speed and distance of the workout(s) you have performed

Run Bud: Questionnaire

Please answer the following questions regarding Run Bud, the smartphone application recently used. Please rate the features of the application from 1 to 5, 1 being the lowest and 5 the highest. Room is left to leave any addition feedback.

User Interface

1 2 3 4 5

Colour scheme

--	--	--	--	--

Consistency

--	--	--	--	--

Clear

--	--	--	--	--

.....

.....

.....

Tracking Workout

1 2 3 4 5

Map

--	--	--	--	--

Distance/Speed details

--	--	--	--	--

Accuracy

--	--	--	--	--

.....

.....

.....

Suggested Workouts

1 2 3 4 5

Template amount

--	--	--	--	--

Template quality

--	--	--	--	--

Template variety

--	--	--	--	--

Details

--	--	--	--	--

.....

.....

.....

Personal Details

	1	2	3	4	5
Details tracked					
Account creation					
Edit options					

.....

.....

.....

Progress

	1	2	3	4	5
Graph					
Graph options					
Workout history					

.....

.....

.....

Name

Signature Date

Appendix E: Running Templates

Long Distance

Stage	Run (mins)	Walk (mins)	Cycles
1	2	4	5
2	3	3	5
3	5	2.5	4
4	7	3	3
5	8	2	3
6	9	2	3
7	9	1	3
8	13	2	2
9	14	1	2
10	30	0	1

To scale the workout series to an intermediate level the run and walk values are doubled and to an advanced level those values are doubled again.

High Intensity Training

Beginner

Stage	Run (secs)	Rest (secs)	Cycles
1	15	60	15
2	20	60	15
3	25	60	15
4	30	60	15
5	30	50	15

Intermediate

Stage	Run (secs)	Rest (secs)	Cycles
1	20	50	15
2	30	50	15
3	30	40	15
4	40	40	15
5	40	30	15

Advanced

Stage	Run (secs)	Rest (secs)	Cycles
1	40	35	15
2	45	30	15
3	50	25	15
4	60	25	15
5	60	20	15

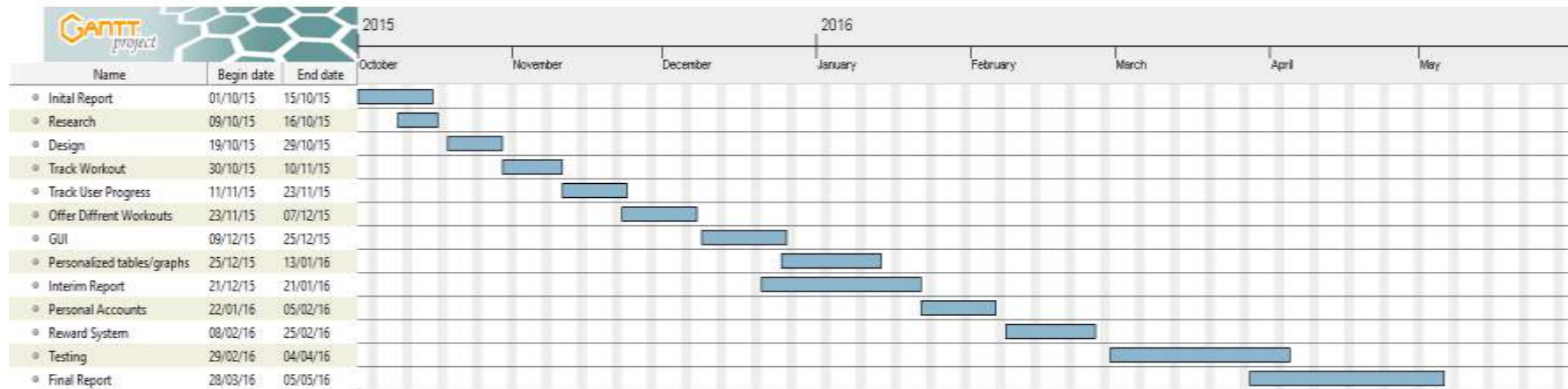
To scale the HIT workout series time running was increased and rest time reduced to increase the difficulty and intensity.

Appendix F: Ethics Checklist

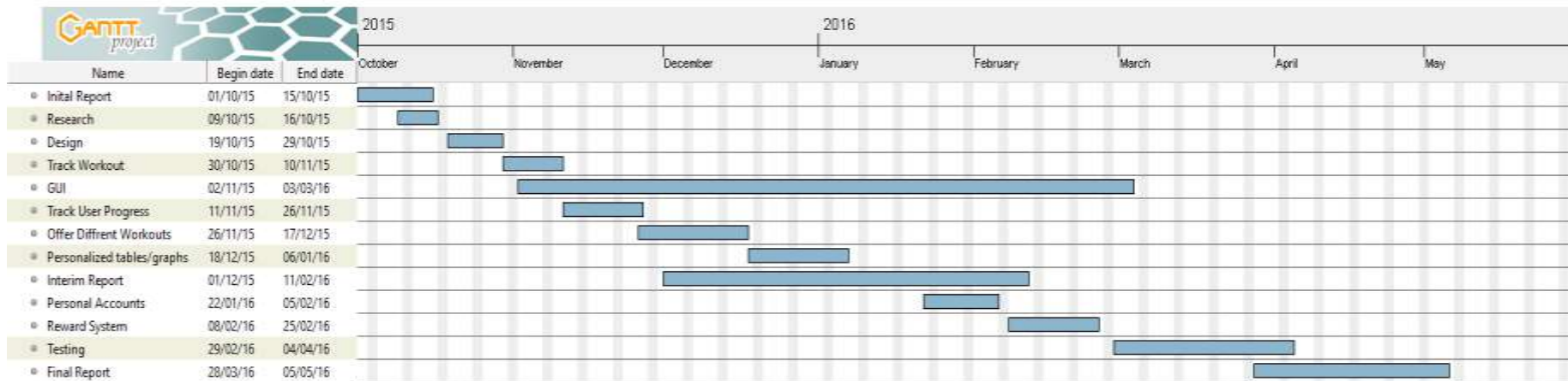
Risk	Severity (L/M/H)	Likelihood (L/M/H)	Significance (Sev. x Like.)	How to Avoid	How to Recover
Development software not available	H	H	HH	Do my project on own home computer where there is all the required software installed	Have all the software also installed on a laptop which can brought anywhere.
Development falls behind schedule	H	M	HM	Pay close attention to the schedule and try to keep to it	Some of the secondary objectives can be removed from the project
User injured using application	H	L	HL	No unauthorized use of the application in development/testing. Provide suitable clothing/equipment for users.	Minor injury: Use first aid and lifeguarding qualification to inspect and treat the injury. Serious Injury: call nearby medical support or ambulance
SVN failure	H	L	HL	Use a reputable SVN provider which has a good up-time	Have an updated local copy of the SVN on own computer so project is always accessible
Not compatible with user's phone	M	M	MM	Only offer the application on android stores and add a minimum android OS requirement	

Appendix G: Time Plans

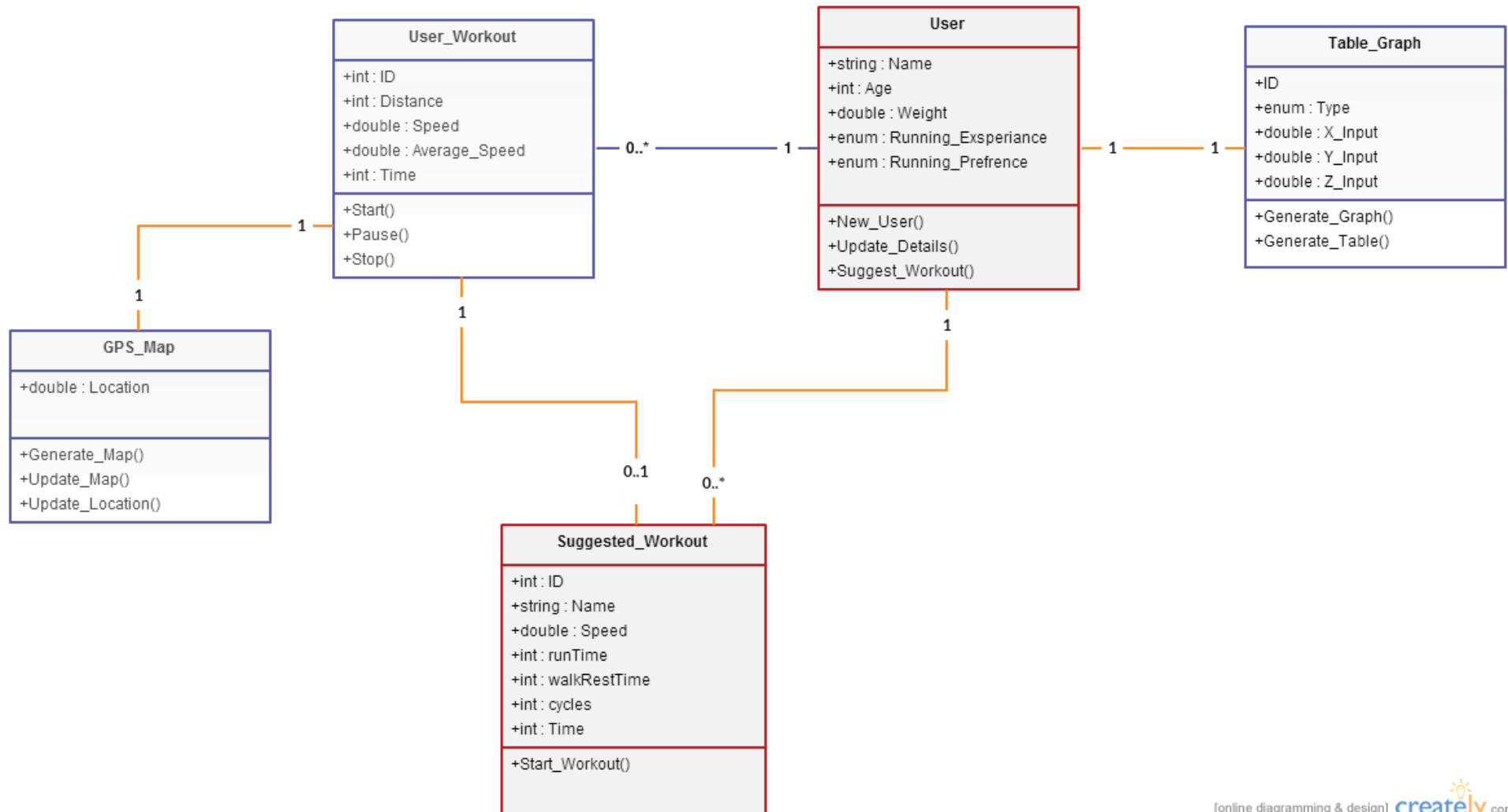
Original Time Plan



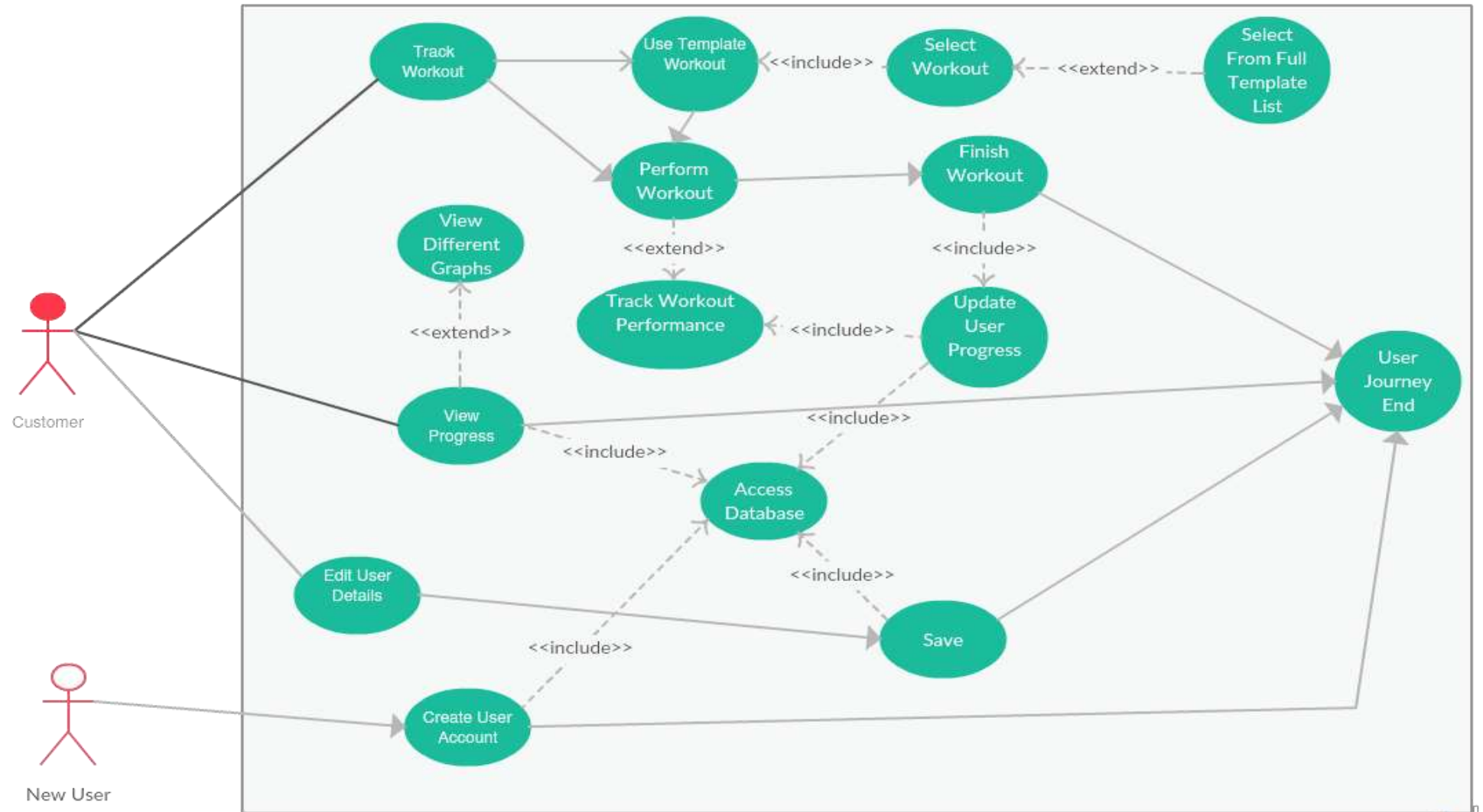
Revised Time plan



Appendix H: Class UML



Appendix I: Use Case Diagram



Appendix J: User Testing Results

	User Interface				Tracking Workout				Suggested Workouts			
	Colour Scheme	Consistency	Clear		Map	Distance/Speed details	Accuracy		Amount	Quality	Variety	Details
1	5	5	4		4	3	5		5	4	3	3
2	5	5	5		5	4	5		4	4	5	3
3	4	5	4		5	5	5		4	5	4	5
4	3	4	4		4	3	4		4	3	3	3
5	5	5	5		5	2	4		3	3	3	3
6	4	4	3		4	3	3		5	5	4	5
7	4	5	4		5	4	5		3	2	2	2
8	5	5	5		5	4	5		4	4	4	4
Average	4.375	4.75	4.25		4.625	3.5	4.5		4	3.75	3.5	3.5
Section Avr			4.46				4.21					3.58
Difference			0.26				0.62					0.14

Personal Details				Progress		
Details tracked	Account creation	Edit options		Graph	Graph options	History
3	4	5		5	4	4
4	5	4		5	3	5
5	5	5		5	4	5
3	4	4		4	3	4
3	4	4		4	4	4
4	5	4		5	5	5
3	4	3		4	3	4
4	4	4		4	4	4
3.625	4.375	4.125		4.5	3.75	4.375
		4.04				4.21
		0.38				0.40

Appendix K: Task List

#	Task Name	Description	Duration (days)
1	Track the workout of the user	Upon activation, start tracking the users running session.	12
1.1	Use phone GPS	Use location and google maps API to gather location information.	
1.2	Display map	Display the current location and path of the user for the run alongside a timer and their speed.	
1.3	Record run	Save all of the data gathered during the session.	
2	Track the Progress of a User	Track analytical information about the user to help them improve and track progress.	10
2.1	Allow user to save details	Keep track of personal details including weight, gender, age, running experience, preference and more in order to suggest suitable workouts.	
2.2	Keep a history of data saved	Save selected previous personal details in order to build a history of the user and track their progress while they have been using the app.	
3	Offer different types of workouts for the user customized for them	Gaging the physical level of the user and offering workouts.	14
3.1	Research suitable range of running workouts	Find common running workouts deigned for different experience and different preference (cardio/fat loss) to make a wide range of workouts to be able to offer.	
3.2	Calculate user's running level	Calculate the user's running ability from their history of workouts and personal details.	
3.3	Suggest one or more workouts to the user	Using the user's preference and ability to run, select one or more running workouts and suggest it to the user.	
3.4	Track current workout with suggested	If a suggested workout is chosen, then it will run alongside the track workout page to show your suggested speeds and run length alongside the real-time numbers.	

4	Personalized data display/graphs		14
4.1	Display range of graphs	Implement a range of tables and graphs so the user's data can be presented in different formats.	
4.2	Offer a range of information options	Using the user's information entered over previous uses of the application offer several different types of statistics to populate the tables and graphs with.	
5	Graphical User Interface		120
5.1	Home page	Main page linking to the other sections of the application.	
5.2	Pages for different features	A page and interface for each main section of the application; track workout, suggested workout, personal details and stat track.	
5.3	Validation	All information entered by the user needs to be validated to reduce the amount of mistakes they can make.	
5.4	Log in/out and registration page (secondary objective)	Register page so new users can make an account, log in page so users can enter their login detail and download their personal data from the server and a logout button so users can upload their updated personal data to the server.	
6	Research		7
6.1	Mobile development	Research and decide which OS to design the application for as well as the minimum requirements of the phone	
6.2	Programming language and environment	Decide which language will be most suited to the project and from there which software would be most suitable programming environment to work in. Any plug-ins or additions software will be researched as well.	
6.3	GPS	Learn about the use of GPS in phone applications, how to access it and make use if it for this project.	

7	Design		10
7.1	Design interface	Make an example of the interface using PowerPoint or paper models.	
7.2	Design the classes and class inheritance	Create documents outlining the classes and class inheritance using specialist software.	
8	Report		
8.1	Initial	First report detailing ideas and objectives.	14
8.2	Interim	Halfway report, tracking and evaluating progress and making decisions based on work so far.	30
8.3	Final	Final report to be submitted detailing how the project went, evaluating it as a whole and coming to conclusions based on experience working on the project.	70
9	Testing		40
9.1	White-box testing	Testing the code and different functions/ methods internally.	
9.2	Black-box testing	Testing the functionality of the program.	
10	Personal Accounts (secondary objective)		14
10.1	Server holding data	Server which can be accessed by the app to download up upload data.	
10.2	Unique login for each user	Personal logins for each user so they can download and access their personal data from the server no matter what phone they are on	
11	Reward System (Secondary Objective)	The user will be able to earn different badges or themes for their account and application if they make significant progress in the different areas being tracked such as weight or total distance ran.	14

References

- Alam, F., 2014. Energy optimization in Android applications through wakelock placement. *2014 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1-4.
- Ali Mili, F. T., 2015. *Software testing: concepts and operations*. s.l.:Wiley-Blackwell.
- Android, 2015. *Tasks and Back Stack*. [Online]
Available at: <http://developer.android.com/guide/components/tasks-and-back-stack.html>
[Accessed 27 10 2015].
- Anon., 2014. *Health And Fitness Apps See 62% Increase In Usage In Last Six Months*. [Online]
Available at: <http://therealtime.com/2014/06/24/health-and-fitness-apps-see-62-increase-in-usage-in-last-six-months/>
[Accessed 10 October 2015].
- Bottaci, I., 2015. *Test and Verification (Advanced Software Engineering)*. Hull: s.n.
- Chatzimilioudis, G., 2012. Crowdsourcing with Smartphones. *IEEE Internet Computing* , 16(5), pp. 36 - 44.
- COLOUR BLIND AWARENESS, n.d. *Business*. [Online]
Available at: <http://www.colourblindawareness.org/business/>
[Accessed 11 1 2016].
- Densmore, M., 2015. *Trainer Tip: Progressive Overload*. [Online]
Available at: <http://www.runnersworld.com/run-matters/trainer-tip-progressive-overload>
[Accessed 14th December 2015].
- Exercise Science Department, S. U., 2005. Acute EPOC response in women to circuit training and treadmill exercise of matched oxygen consumption. *Eur J Appl Physiol*, 94(5), pp. 500-504.
- Hellman, E., 2013. *Android Programming, pushing the limits*. s.l.:Wiley.
- Hideo Makino, I. I. M. N., 1996. Development of navigation system for the blind using GPS and mobile phone combination. *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE* , Volume 2, pp. 506 - 507.
- Hung, P. T., 2013. Solutions to data optimization and security for web services in GNSS applications based on android smartphone. *Information and Communication Technologies (WICT), 2013 Third World Congress on*, pp. 63-68.
- IDC Research, Inc., 2015. *Smartphone OS Market Share, 2015 Q2*. [Online]
Available at: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
[Accessed 19 1 2016].
- Kapoor, R., n.d. *How does Android development compare to iOS development?*. [Online]
Available at: <https://www.quora.com/How-does-Android-development-compare-to-iOS-development>
[Accessed 13th October 2015].
- Kiran Bala, S. S. a. G. K., 2015. A Study on Smartphone based Operating System. *International Journal of Computer Applications*, Volume 121(1), p. 18.

- Kirk Radeck, W. E. M., 2003. *C# and Java: Comparing Programming Languages*. [Online] Available at: <https://msdn.microsoft.com/en-us/library/ms836794.aspx> [Accessed 19 1 2016].
- Lehtimäki, J., 2013. *Smashing Android UI*. s.l.:John Wiley & Sons.
- McClure, W. B., 2012. *Professional Android programming with MonoDroid and .NET/C#*. s.l.:Wrox.
- Md. Palash Uddin, M. Z. I. M. N. M. I. A., 2013. GPS-based Location Tracking System via. *International Journal of Research in Computer Engineering and Electronics*, 2(5), p. 1.
- Meier, R., 2008. *Professional Android Application Development*. s.l.:Wiley Publishing.
- Steven D. Tripp, B. B., 1990. Rapid prototyping: An alternative instructional design strategy. *Educational Technology Research and Development*, 38(1), pp. 31-44.
- Tennant, D., 2013. *IOS vs Android: Which is the Best Smartphone Platform for Me?*. [Online] Available at: <http://cell-phones.toptenreviews.com/smartphones/ios-vs-android-which-is-the-best-smartphone-platform-for-me-.html> [Accessed 16 October 2015].
- Vavra, D., 2010. *Improve Android GPS Position Accuracy With GPS Averaging*. [Online] Available at: <http://androgeoid.com/2010/10/improve-android-gps-position-accuracy-with-gps-averaging/> [Accessed 29 01 2016].
- Vukovi, M., 2009. Crowdsourcing for Enterprises. *Congress on Services*, Volume 1, pp. 686 - 692.
- Walker, M., 2015. *08348 Interpreted Languages*. Hull: Martin Walker.
- Xamarin, 2015. *Frequently Asked Questions*. [Online] Available at: <https://xamarin.com/faq> [Accessed 27 December 2015].
- Xamarin, n.d. *Application Fundamentals*. [Online] Available at: https://developer.xamarin.com/guides/android/application_fundamentals/ [Accessed 14 04 2016].
- Xamarin, n.d. *Introduction to Mobile Development*. [Online] Available at: https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/ [Accessed 08 04 2016].
- Yoshihito Kuranuki, T. U. T. Y. S. Y., 2014. A New Business Model of Custom Software Development. *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices*, pp. 73-77.