Interim Report

Project Title	Progressive overload mobile application
Module Number	UO8096
Student Number	09092543
Name	Lee Hudson
Supervisor	David Lightfoot
Date	19th November 2013

Contents

Introduction:

As outlined in the proposal the aim of this project is to produce an application to address the needs of a gym workout method known as progressive overload.

Progressive overload is a weight lifting technique that is commonly used to improve performance and weight gain. The principle of progressive overload is simple, one must strive to increase the load placed on one's body in order to stimulate muscle growth (Goulet,2004). In practice this means having a set workout and increasing each exercise in some way, be it number of reps, weight or time between sets. The aforementioned increases are quantities and easy to represent numerically but technique is also important, this will be hard to record.

The above technique of progressive overload can be hard to implement effectively in a gym environment, it requires a lot of note taking and can be time consuming in itself. It can sometimes be hard to remember where you left off and it can be hard to determine if you really are improving. This is where the advantages of mobile computing can assist.

The above activity is essentially record keeping and data analysis. A mobile application can drastically improve the practicality of progressive overload.

It was decided in the proposal that the platform choice would be android. This is still the case. The main reason for choosing android was the ease and cost of development on this platform. Also due to the wide range of devices that run android I will provide an additional dimension of complexity, this application must be able to run and be usable on both phones and tablets. The other possible platforms were IOS and windows mobile. IOS was rejected due to the cost of developing and the lack of experience in objective C, windows was rejected due to it not being such a popular platform.

Further Research:

So far in addition to the research material detailed in the proposal further research has been conducted. The two main articles that have been studied are as follows

Article 1: Multi-pane development in android with fragments - Tutorial

Vogel,L.(2014*) Multi-pane development in android with fragments - Tutorial [*Online] 29.04.14. Available from - http://www.vogella.com/tutorials/AndroidFragments/article.html [Accessed: 03.10.14]

The above article is a tutorial detailing showing how to develop android applications for a variety of screen sizes and in particular catering for both tablets and phones by displaying applications differently depending on both screen size and orientation. This tutorial has a specific example which is extremely applicable to the progressive overload application. In the RSSFeed example it shows the class structure and interactions required to display multiple GUIs on the screen at once for larger landscape orientated displays and single GUIs for smaller or portrait displays.

The author Lars Vogel is a prolific tutorial write for a variety of computer science based tutorials and is the founder of the professional training company Vogella. This company not only write tutorials online but assist in training other companies in many areas of computing. From personal experience the tutorials online are fairly clear and easy to follow, they also seem to work.

This article is kept up to date with the last update on 13/11/2014. Therefore this article can be considered current and up to date with the current state of technology, in particular android.

Article 2: Fragments

Unknown(unknown) *Fragments.* [Online] 20.11.14. Available from- http://developer.android.com/guide/components/fragments.html [Accessed: 20.11.14]

This article is a tutorial / explanation of how to use android fragments. As with article 1 it details the class structure required to create flexible user interfaces in android using fragments. This article comes across as more of a technical reference than a step by step tutorial but is still extremely helpful in assisting ones understanding of how fragments work.

This article is on the android developers website. Whilst the actual author is not mentioned it can be assumed that if this article is on the Android website that it will be correct.

As this article is published on the Android website it will probably be up to date. This website is the main point of reference for android developers.

In addition to the above articles many other articles have been seen to be relevant and so have been used. These have been detailed in the references.

Methodology and Resources:

In the proposal it was stated that the waterfall software development method would be used. This is still the case and consists of the following stages:

1. Requirements:
    a. Complete the requirement specification
2. Design
    a. Decide on the classes to be used
    b. Decide what these classes will do and how they will interact
    c. Compose a UML diagram of the system
    d. Compose and normalise the database
    e. Design the user interface
3. Implementation
    a. Create android activities
    b. Implement GUIs for each activity
    c. Implement communication between activities
    d. Implement database
    e. Implement custom graph view
    f. Implement timer option
    g. Implement any audio
4. Verification
    a. Create a test schedule
    b. Test the mechanics of the system
    c. Perform beta testing on application by deploying to a few test subjects

This application will be developed using Eclipse with the android plugin. I am familiar with both Eclipse and the Android plugin. For testing purposes both a large screen and a small screen device will be needed. These are available and will be a Samsung 10.1 tablet and a Nexus 5 phone. Basic testing has proven that I can develop with this software on both devices.

For debugging the eclipse package has a useful logging tool called “Logcat” which allows live readable outputs whilst the android device is connected. This is much like printing to the terminal in other languages.

The main language used to develop android is Dalvik, this is basically java with some additional API. Therefore I am already familiar with this language and this doesn’t really pose any additional barriers for me. So far the hardware and software used hasn’t posed any limitations.

After playing out different use cases the below requirement specification has been finalised:

Functional requirements:

1. Allow the user to create a workout .
2. Allow the user to create an exercise.
3. Allow the user to add exercises to a work out.
4. Allow the user to remove a workout.
5. Allow the user to remove an exercise.
6. Allow the user to record the performance each time an exercise is performed.
7. Allow the user to see the performance from their last workout to allow them to clearly see what they have to better.
8. Give the user an audible alarm to signify the end of a rest period.
9. Allow the user to see how their performance has increased in a graphical way.

Non-functional requirements:

1. The application should respond to an input within 500ms.
2. The application must retain data on power off.
3. The data stored by the application must be consistent.
4. The application must work in both landscape and portrait orientations.
5. The applications must work on screen sizes from 640x480 to 1920x1080.

Using the above functional requirements it was possible to design the GUIs and verify that once the functional code has been implemented all the functional requirements will be met.

Looking at the non-functional requirements I can now comment on how these will be met:

Non-functional requirements:

1. The application should respond to an input within 500ms.
    a. Any time consuming functions such as the rest timer or audio playing will be done on a separate thread.
2. The application must retain data on power off.
    a. The data will be stored in an SQL database, in android this is implemented using what is called a content provider.
3. The data stored by the application must be consistent.
    a. The database will be normalised to BCNF.
4. The application must work in both landscape and portrait orientations.
    a. This it catered for in the android framework. Separate GUI layout files are used for portrait and landscape.
    b. Android fragments improve on this further by allowing multiple user interfaces to be displayed at one time.
5. The applications must work on screen sizes from 640x480 to 1920x1080.
    a. The screen size is a variable that can be queried directly in the code and so can be catered for.

With the above project there are inherent risks:

1. Project taking too long to due lack unforeseen complexity
    a. The GUI programming was initialled seen as a possible source for unforeseen complexity. As further work has been done and GUIs are now understood well this is only a small risk.
    b. Implementing the content provider on an android application is always quite complex, the addition of having multiple SQL tables adds to this complexity. If this proves too complex it may be easier to use multiple content providers, one for each table.
2. Some aspects not possible on android.
    a. This risk has been mitigated as all areas have been researched to the extent that I now know all functional and non-functional requirements can be met using android.

Progress and Achievements:

So far good progress has been made but the order in which components were designed needed to be swapped around. So far the following has been completed:

Finalised requirement specification:

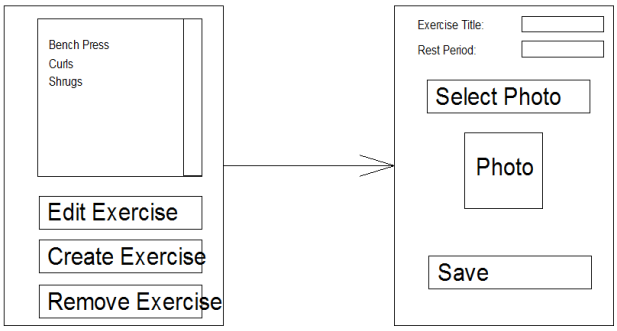
As seen in the methodology section the requirement specification has now been finalised.

GUI design:

Using the requirement specifications as a guide it has been possible to design the user interface for both portrait and landscape modes. Much work has been done in order to ascertain how exactly to implement different user interfaces for different screen sizes and orientations. It has been decided that any phone size device will be locked in portrait mode. When it comes to tablets the same user interface used on the phone will be used if the tablet is in portrait, if it is in landscape however the tablet will display more than one layout file as in the below example:

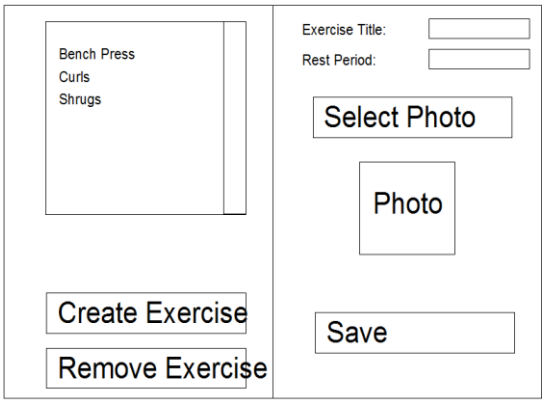
Editing an exercise:

Phone



On a phone the user will select an exercise from the list and press edit exercise. This will then start another android activity which will invoke another GUI to allow the user to edit and save an exercise. Due to the size of most phones this allows the interface to be readable by not over complicating it.

Tablet:



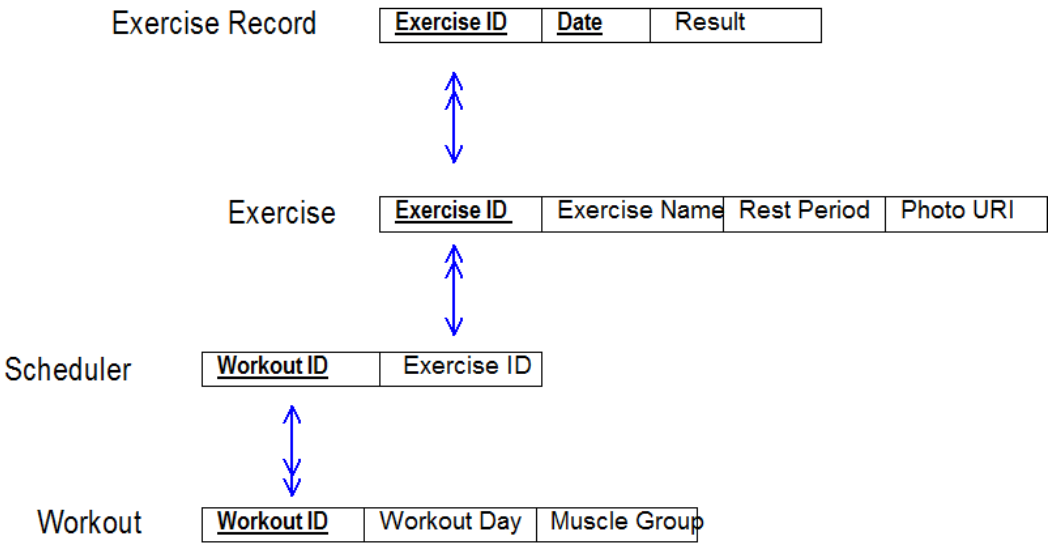
For a tablet however there is much more real estate in terms of screen size. It is possible to fit both GUIs on the same screen. Now when a user want to edit an exercise they select the exercise from the list and the details then come up in the right hand gui. The user then edits the exercise in this GUI and clicks save.

Whilst this looks quite simply it is rather convoluted. In order to understand how to do this better is created an android app using the tutorial from article 1. I now know how to do this with fragments.

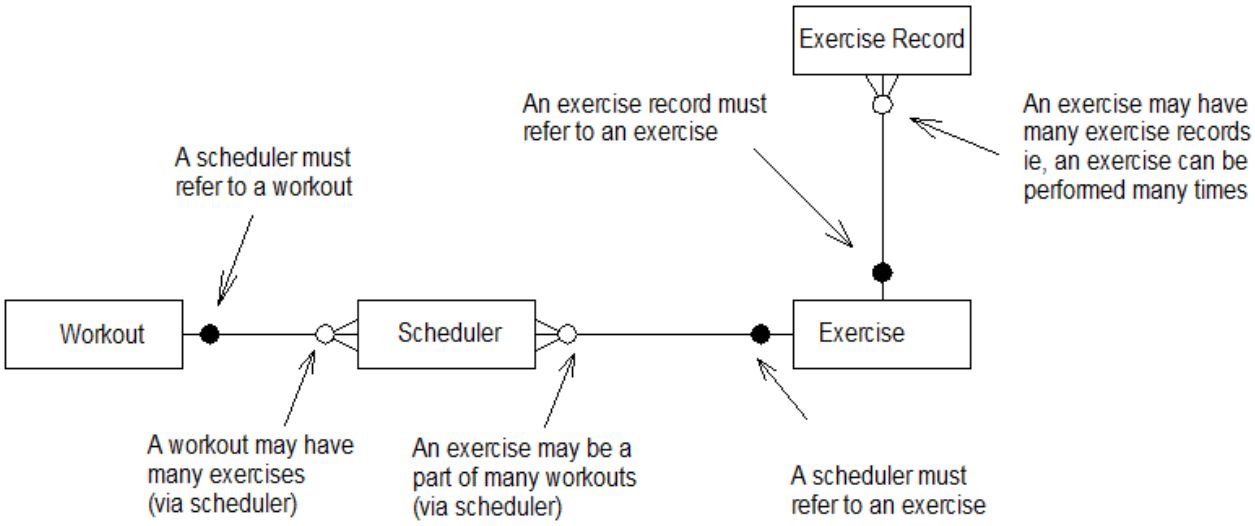
Database composition and normalisation:

Taking into account the requirement specifications the following database has been composed, here it is in BCNF:

RI Diagram:



EAR Diagram:



The full details of how the database was normalised will be detailed in the final report.

Class diagram in UML:

Looking at how android applications work I believe it will be hard and probably not that informative to produce a UML diagram for the system. The reason I believe this to be the case is the way android activities interact. Each GUI will have its own android activity (or vice versa). The android activity will be a specialised subclass of the android activity class. Interactions or relationships between these activities is a form if explicit invocation, an android activity will pass focus onto another activity sending an amount of data with it via what is called an "Intent". The receiving activity has no knowledge of or association with the previous activity. Therefore a more useful representation of the system would be a flow diagram. This will be derived from the GUI layouts.

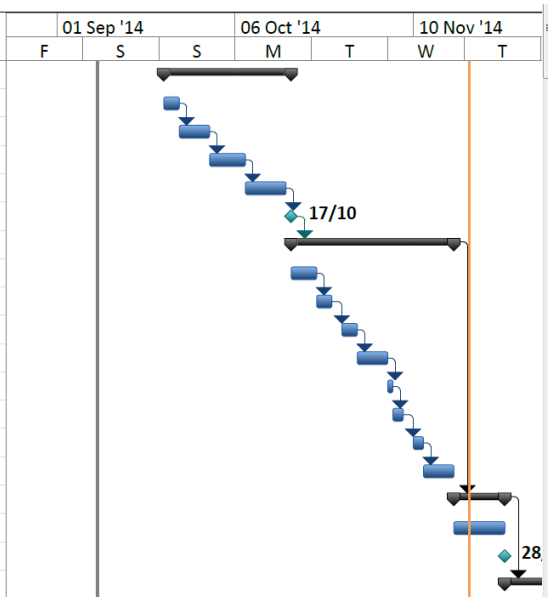
Problems encountered:

The main issue that has been encountered so far is the class diagram issue. I originally intended on doing the class diagram first but the class diagram heavily depended on my knowledge of fragments and GUIs. Therefore I needed to compose the GUI layouts and fully understand fragments before I could work on the main class diagram. As detailed above I also realised that UML is not the best format for this. The GUI layouts have been completed and provide enough information to construct a flow chart.

Progress with respect to the project plan:

So far I am up to date with the original project plan. The order of some components were swapped around (interfaces before class design) but the design stage is complete.

ID	Task Mode	Task Name	Duration
1		**Planning and research**	**19 days**
2		Project Plan	3 days
3		Gather Literature	4 days
4		Assess literature	5 days
5		Compile proposal	6 days
6		Submit proposal	0 days
7		**Design**	**22 days**
8		Compose Requirement Specification	3 days
9		Determine main classes and interactions	3 days
10		Compose UML diagram of system	3 days
11		Determine database entities	4 days
12		Normalise database	1 day
13		Design user interface	2 days
14		Design fragments for different screen sizes	2 days
15		Spare Time	4 days
16		**Interim Report**	**8 days**
17		Compose interim report	8 days
18		Submit interim report	0 days
19		**Implementation**	**45 days**



References:

I. Vogel,L.(2014*) Multi-pane development in android with fragments - Tutorial [*Online] 29.04.14. Available from - http://www.vogella.com/tutorials/AndroidFragments/article.html [Accessed: 03.10.14]

II. Unknown(unknown) *Fragments.* [Online] 20.11.14. Available from- http://developer.android.com/guide/components/fragments.html [Accessed: 20.11.14]

Bibliography:

I. Sherman,E.(2012*) How to choose the best platform for your app*.[Online] 08.10.12. Available from - http://www.inc.com/erik-sherman/avoiding-the-single-mobile-platform-trap.html [Accessed: 03.10.14]

II. Bret(2013*) The ten rules of progressive overload*.[Online] 26.02.13. Available from - http://bretcontreras.com/progressive-overload [Accessed: 03.10.14]

III. Vogel,L.(2014*) Multi-pane development in android with fragments - Tutorial [*Online] 29.04.14. Available from - http://www.vogella.com/tutorials/AndroidFragments/article.html [Accessed: 03.10.14]

IV. Varshneya,R.(2013*) iOS or Android? Choosing the best platform for your application*.[Online] 17.07.13. Available from - http://www.entrepreneur.com/article/227426 [Accessed: 03.10.14]

V. Goulet,C.(2004*) Progressive overload: The concept you must know to grow!*[Online] 19.10.04. Available from - http://www.bodybuilding.com/fun/goulet11.htm [Accessed: 03.10.14]

VI. Vogel,L.(2014*) Android SQLite database and content provider - Tutorial* .[Online] 19.08.14. Available from - http://www.vogella.com/tutorials/AndroidSQLite/article.html [Accessed: 03.10.14]

VII. Unknown(unknown) *Fragments.* [Online] 20.11.14. Available from- http://developer.android.com/guide/components/fragments.html [Accessed: 20.11.14]