CI360 Mobile Application Development – Reflective Report
George Lancaster – 15808273
Course: BSc (Hons) Computer Science Full-time

Contents

Introduction

This report shows the design and implementation process followed for the creation of a fitness mobile application, FitTracks. Specifically, it shows the research, design rationale and the testing that was performed.

Background

Tools and APIs

FitTracks has been developed using the Android Software Development Kit (SDK). The following packages have been used in its development:

1. Firebase – A database API and toolkit for android applications, owned by Google (Tamplin, 2018).
2. Compact Calendar View – A new calendar library for Android. It has been released under the MIT licence and therefore free to use in this project (Sundeep, 2018).
3. Graph View – An Android graphing library. Released under the Apache v2 licence and also free to use in this project (Gehring, 2018).

Technology in Fitness

The portability of the mobile platform is an important aspect to consider when planning the development of a fitness application. In 2012, 51 per cent of people reported using their mobile phones whilst working out (Pirc, 2012), a number that has only risen in recent years with the ever-increasing popularity of smartphones.

Using technology has also been shown to increase adherence to exercise programs. A 2010 study testing the effects of technology-based exercise programs on individuals aged between 67 and 86 found an 8 per cent increase in adherence, compared to traditional exercise programs (Valenzuela *et al.*, 2010). It can be expected that a greater increase in adherence would be seen in younger participants, who may have a greater dependence on technology (Carroll *et al.,* 2002). This suggests that technology can be used in conjunction with exercise to improve perseverance and consistency when following a workout regimen.

Current research suggests that technology and fitness can have a symbiotic relationship, and that there is a demand for fitness tracking applications.

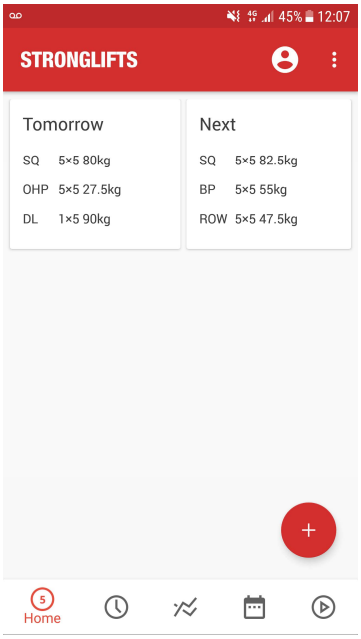
Competitor Analysis

There are currently many fitness tracking applications already on the Google play store. Two of the main competitors that have inspired FitTracks are ‘StrongLifts 5x5’ and ‘FitNotes’.

StrongLifts 5x5

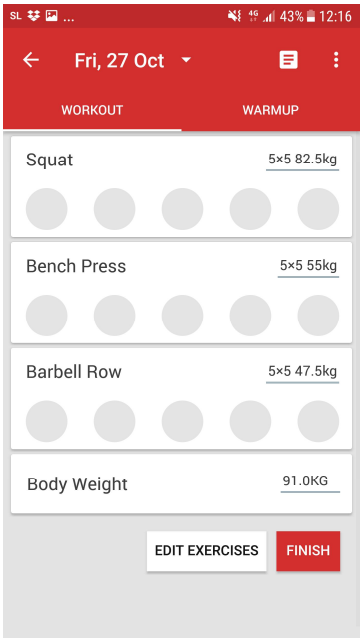
StrongLifts 5x5 is a weight lifting application which tracks progress over time by incrementally adding weight each time a workout is completed. The application has a bottom navigation with 5 tabs to choose from: Home, History, Progress, Calendar and Videos (StrongLifts, 2018).

Layout and Activities



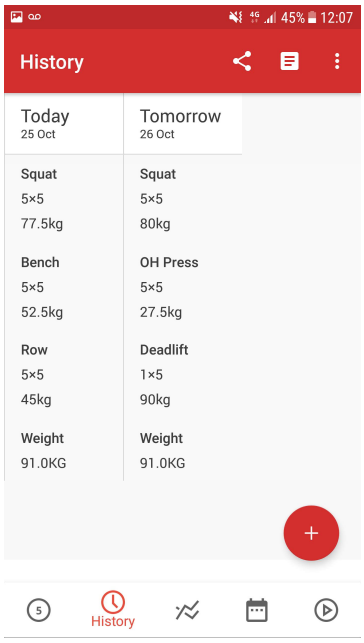
Home page

Shows upcoming workouts.



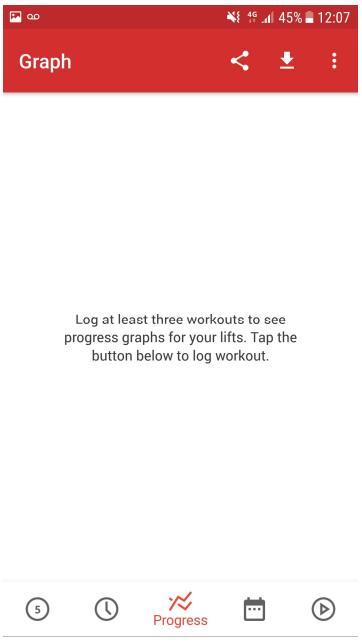
Workout

Shows each workout in detail. This activity is to be used when in the gym.



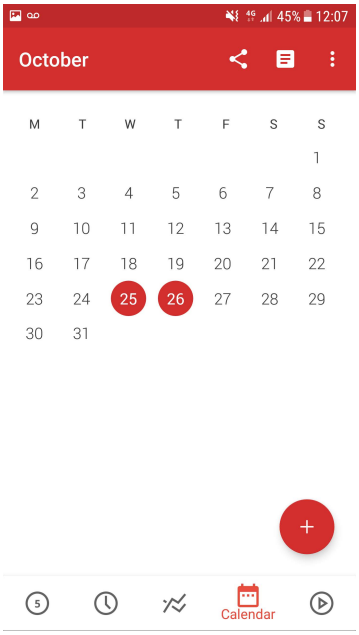
History

Shows all completed workouts in a list.



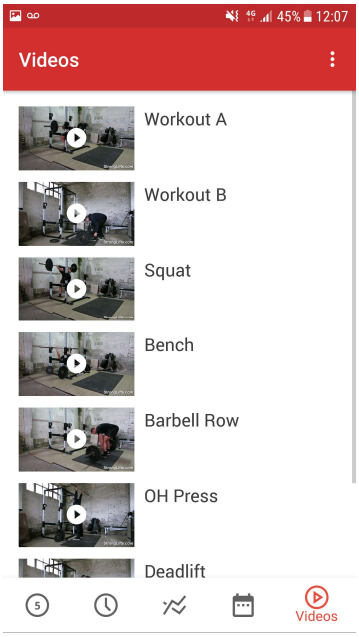
Progress

Plots progress in a chart.



Calendar

Another view for tracking completed workouts.



Video

Video demonstrations on how to perform the exercises.

Analysis

StrongLifts biggest strength is in its simplistic design, the bottom navigation is intuitive and easy to use. The workout tab is also well laid out, tapping on a button after each set is a good feature and makes the application feel interactive.

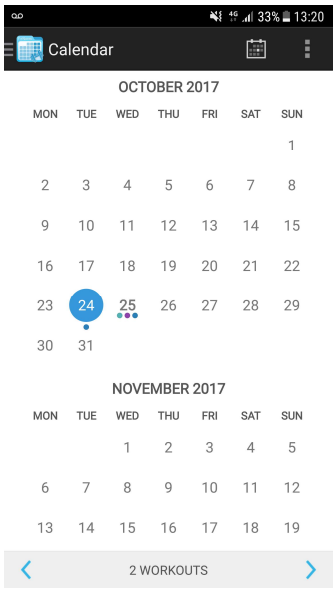
Improvement could be made regarding utilisation of space, for example the calendar only uses half of the screen, which leaves a large area of wasted space. Whilst white space is often an aspect of an app's style and design, it is not aesthetic in this instance. The 'Videos' tab is unnecessary, considering the application is primarily used by individuals who already have a good knowledge of these exercises. This feature should not be one of the top-level navigation destinations. The navigation could be improved as the Calendar/History/Graph tabs could be merged into one. Doing so would make the navigation bar more streamlined.

A similar design was adopted for the Workout, Track and Graph activities in FitTracks. StrongLifts has also inspired the navigation of FitTracks, which uses a bottom navigation. FitTracks improves upon StrongLifts by allowing users to select from a range of workouts, as well as to share their own.

FitNotes

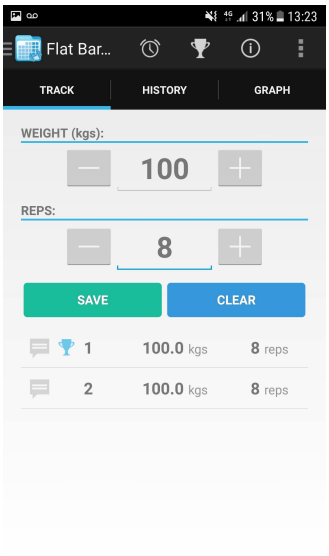
FitNotes is a customisable workout planner that acts like a notepad. It tracks important data such as bodyweight and personal records that can be displayed graphically (FitNotes, 2018).

Layout and Activities



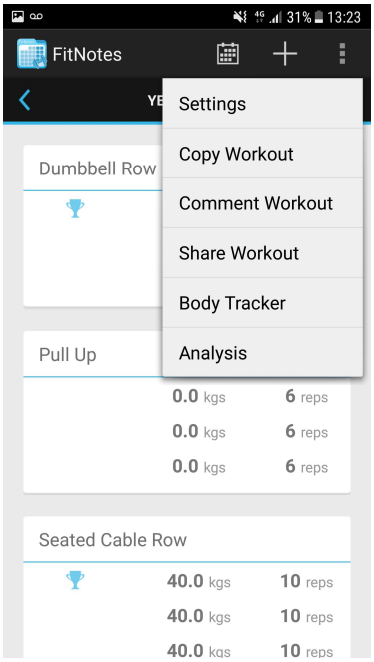
Calendar

Viewing completed workouts on a calendar.



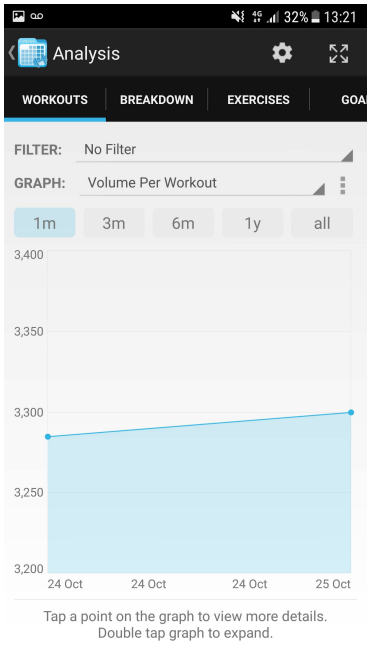
Exercise

Adding an exercise to a workout.



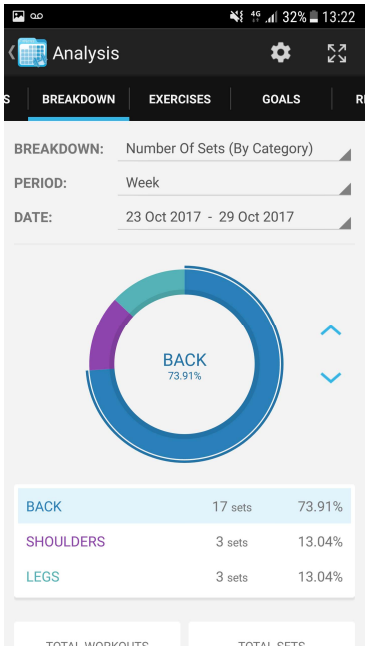
Navigation Menu

Allows for the navigation of the application.

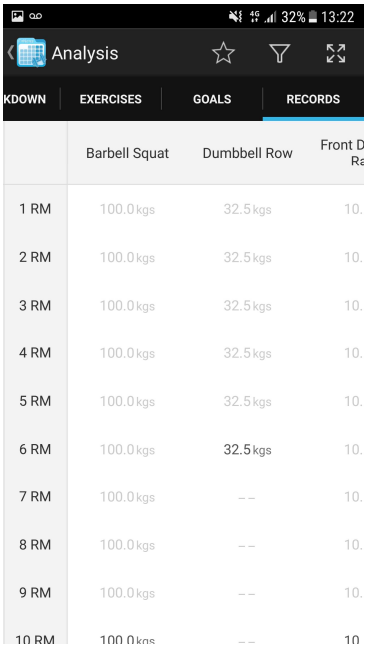


Analysis

Viewing graphical analysis of workouts.



Analysis
Viewing graphical analysis of Workouts.



Analysis
Personal Records.

Analysis

FitNotes is good at tracking progress using graphical representation, possessing many in-depth analysis tools useful for a professional athlete or personal trainer.

Drop-down lists are used for parts of the navigation, making some areas of the application awkward to use. There is too much information displayed on some activities, making the screen look cluttered. The app may also have too much functionality, which can make navigation difficult. It seems like FitNotes was designed for desktop rather than mobile, although this may be because it was tested on a smaller device. The UI is not user-friendly, especially for a user without much knowledge of fitness terminology and this detracts from the overall user experience.

FitTracks incorporates a 'share' feature, inspired by the one found in FitNotes, as well as using a similar Calendar activity to display progress.

Design Rationale

According to the material design guidelines, a bottom navigation bar should be used for between three to five top-level destinations that require direct access. The height of the navigation should be 56dp, and each section should be equally sized (Material, 2018). Given that FitTracks performs three distinct tasks (create and share workouts, interactively complete a workout and track progress), the bottom navigation bar is the most suitable navigation option.

The other navigation method considered was a top navigation bar, which was not chosen due to the ergonomics of using a device one-handed. A report on user experience found that 75 per cent of mobile use is done using the thumb and 49 per cent is done using one hand (Hoober, 2013). FitTracks has been designed for use when exercising, so it is important that it can be used one-handed. A navigation at the top of the screen can be difficult to reach with one hand as shown in Fig 1, therefore it is logical to place the main navigation at the bottom of the screen.

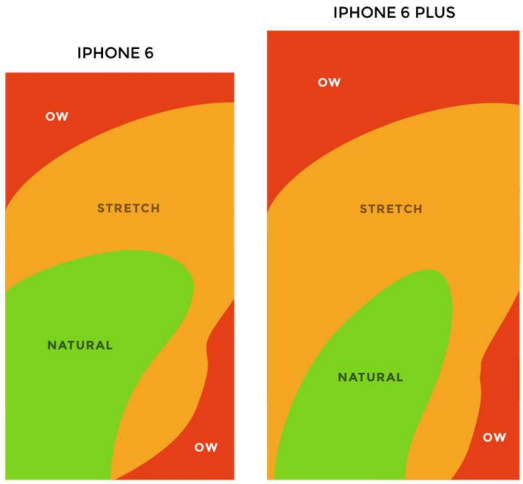


Figure 1: A heat map showing areas of use on common screen display sizes (Hurf, 2014)

Although many applications use fragments when designing a bottom navigation, their use can be detrimental to performance. Using activities, rather than fragments arguably gives greater flexibility through improved encapsulation, testability and reusability (Ableson *et al.*, 2011).

Activities were used instead of fragments as each top-level navigation destination within FitTracks is conceptually unique. This allowed for the separate development of each activity without breaking encapsulation. This is also true for testing and debugging; it is much easier to find the cause of a bug when the source can be traced to an activity. In terms of reusability, all top-level destinations extend from a BaseActivity, and whilst they are conceptually different, they use similar implementations to achieve different goals.

The decision was made to limit FitTracks to run in portrait mode only in all activities apart from the Graph activity, ensuring that the UI could not be warped by different screen sizes. This also improves its functionality as a one-handed application as it is difficult to use a device one-handed in landscape mode.

Login Activity

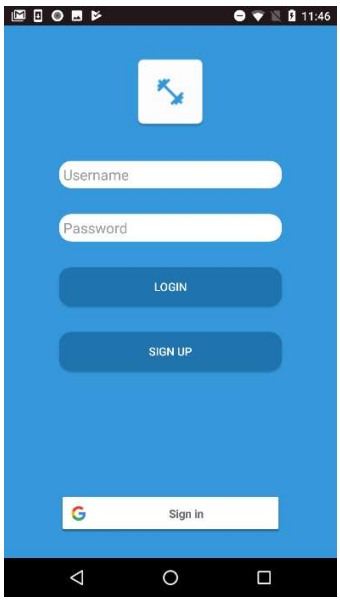
The Login activity is used for user authentication, this is necessary as all users must have an account to use FitTracks. It also links to the New User activity.

Entering a valid email address and password will log a user into the application, and authenticate with Firebase using email and password.

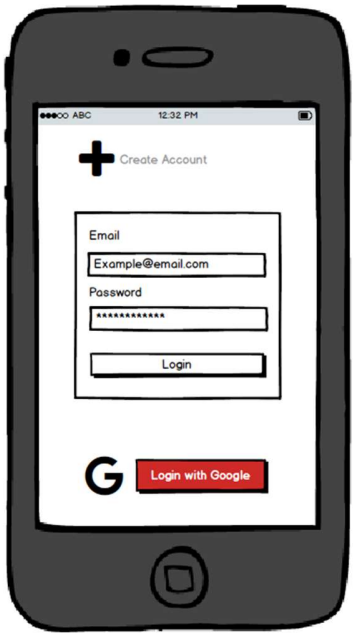
If a user does not have an account, they can create one using the Sign-up button. This takes any information from the two EditText fields, and passes it to the New User activity.

Users can also sign in using their Google account, using the button at the bottom of the activity. This creates a new Firebase user so that there is no change in functionality.

The username and password fields are high up on the screen so that they are not blocked by the keyboard on text entry.



Actual Implementation



UI Draft

New User

The New User activity is used to create a new user on the Firebase database using email and password validation.

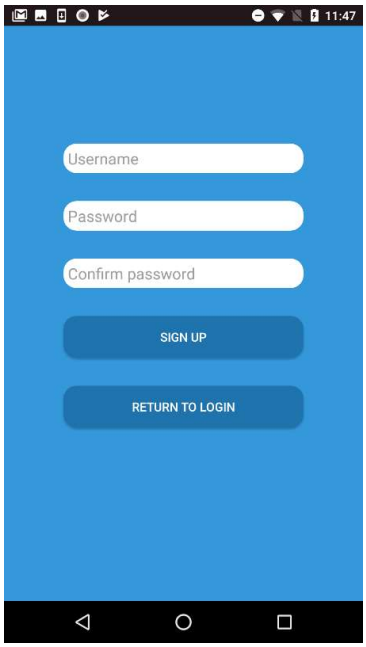
Any information entered on the Login activity will be used in the sign-up form.

Tapping the sign-up button creates a new user on Firebase using the username and password entered on this activity.

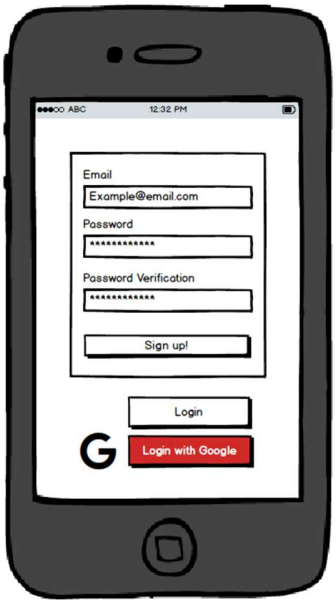
For validation, the password must be entered twice before a new account is created.

The Return to Login button returns the user to the login screen.

As with the Login activity, the username and password fields are high up on the screen so that they are not blocked by the keyboard.



Actual Implementation



UI Draft

Hub Activity

The Hub activity is the first destination the user will see after login, it is one of three top-level destinations. It contains two tabs: My Workouts and Browse Workouts. The My Workouts tab contains all workouts the current user is subscribed to, and the Browse Workouts tab shows all workouts created by all users, which can be subscribed to.

There is a tab navigation bar to switch between the users subscribed workouts, and the workouts available for subscription on the database.

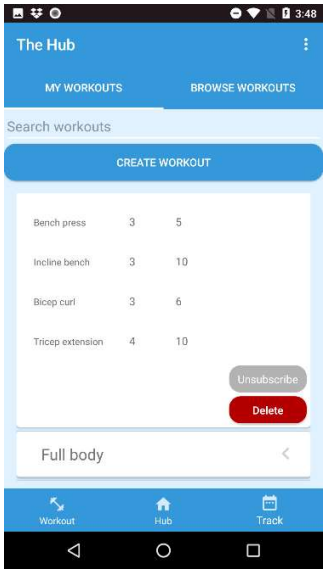
Tapping the Create Workout button will start a new activity to create a workout.

Each workout is displayed as an expandable CardView. Tapping the arrow on the right-hand side will expand the view to display the exercises with their set and repetition scheme.

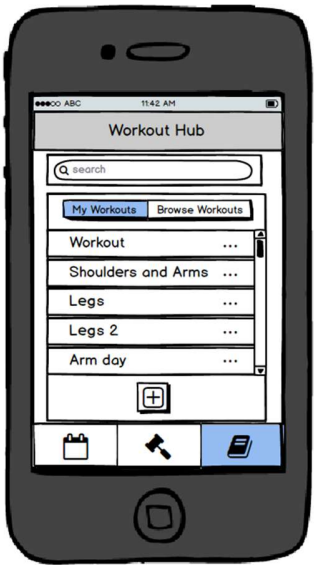
Subscribing to a workout will add it to the My Workouts tab and change the colour of the subscribe button to grey, indicating that the workout has been subscribed to. It also adds the workout to the list associated to the user in the database. This means that the user can select the workout, and complete it in the Workout Activity.

The Unsubscribe button removes the workout from the list of user workouts on the database. This removes the workout from the My Workouts tab and changes the button back to blue in the Browse Workouts tab. The user can no longer complete the workout on the Workout tab.

If the user owns the workout, they have access to the Delete button, which removes the workout from the database for all users.



Actual Implementation



UI Draft

Create Workout

The Create Workout activity is used to create a new workout, all workouts created are publicly available to all users via the Browse Workouts tab on the Hub activity.

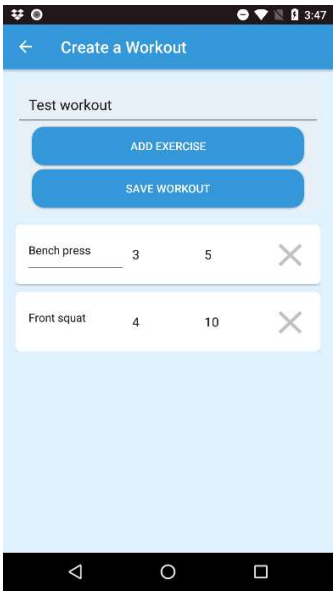
The name of the workout can be changed using the EditText at the top of the activity.

New exercises can be added to the workout by tapping the Add Exercise button. The name of the exercise can be changed using the EditText. This uses an autocomplete feature from a list of well-known exercises.

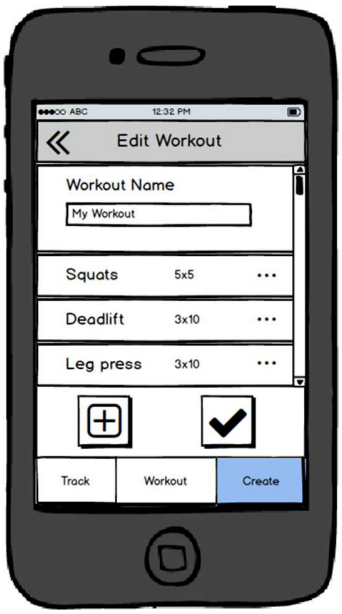
The sets and repetitions can be changed using the two drop-down boxes.

Tapping the grey cross deletes the exercise from the workout.

The Save Workout button saves the workout to the database. The current user is the owner of the workout, and can therefore delete it.



Actual Implementation



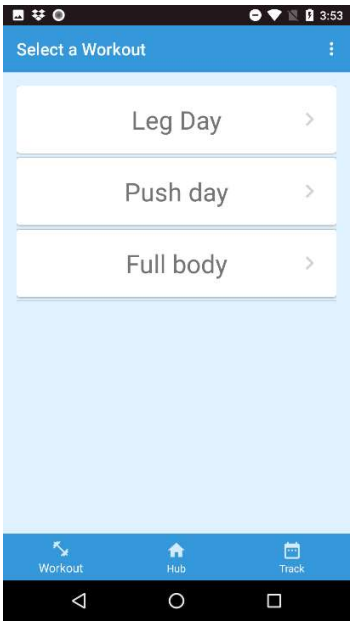
UI Draft

Workout Activity

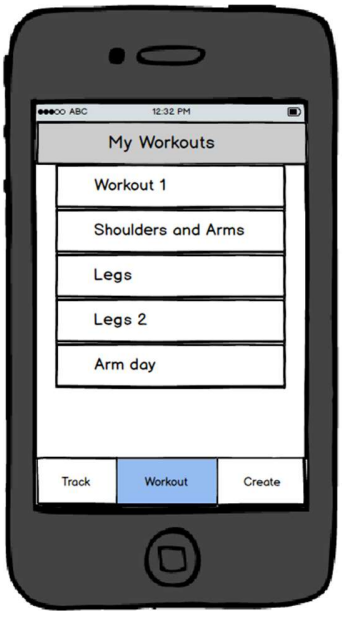
The workout activity is the left-most top-level destination, it is used to select a workout to complete.

The Workout activity uses a RecyclerView to display the current user's workouts. It is populated using data gathered from the database.

Tapping an item in the list selects the workout and starts the Complete Workout activity. As all the data is loaded in this activity, the correct workout is passed to the Complete Workout activity when it is started. This saves bandwidth as only one retrieval is done from the database.



Actual Implementation



UI Draft

Complete Workout Activity

The Complete Workout activity is started by selecting a list item in the Workout activity. It is used to interactively progress through a workout. The user taps the circular icon after completing each set, indicating that it has been completed.

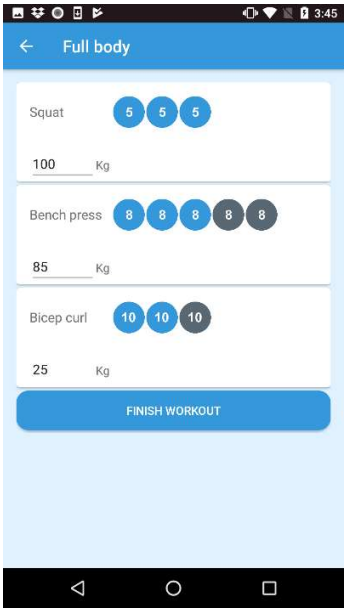
The title of the activity in the action bar changes based on the workout selected.

The number within the circular icons represent the number of repetitions in a set, and the number of circles represents the number of sets. For example, the repetition range for the Bench Press exercise is 5 sets of 8 repetitions.

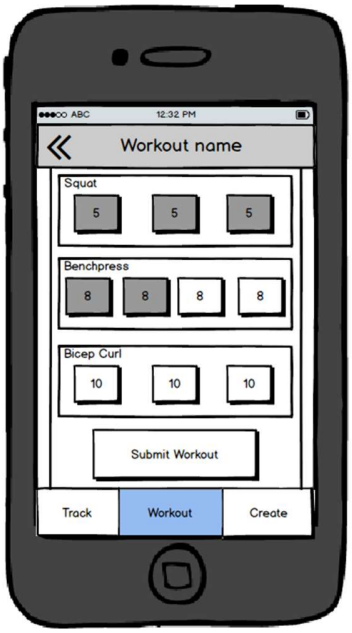
Tapping on a circular icon marks the set as complete by turning its colour from grey to blue.

The weight used in the exercise can be adjusted to suit the capabilities of the user, creating a personalised workout.

Once all icons have been selected, tapping on the Finish Workout button submits the workout to the database, end the current activity and start the Track Activity. The workout does not need to be completed in order to be submitted, allowing users to finish workouts prematurely. This is reflected on the database and the Track activity calendar.



Actual Implementation



UI Draft

Track Activity

The Track activity is a calendar, showing all workouts completed by the user, allowing them to visually track their progress. It is populated by completing workouts in the Complete Workout activity.

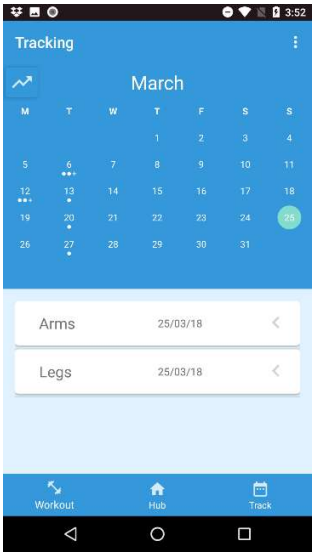
The small white dots on the calendar represent completed workouts.

The dark blue circle represents what day it is today.

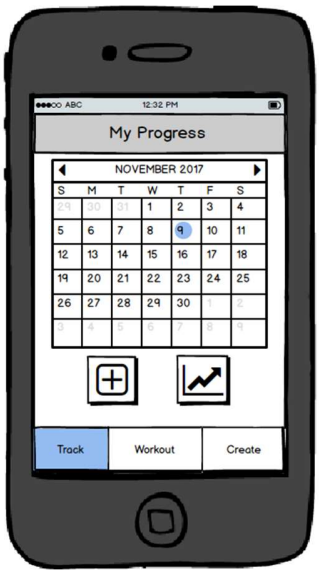
The light blue circle represents what day is currently selected.

When selecting a day, the workouts completed on that day are displayed in expandable CardViews below the calendar. Tapping on the grey arrow on the right expands the view, displaying additional information about that workout.

The arrow button below the activity title starts the Graph activity.



Actual Implementation



UI Draft

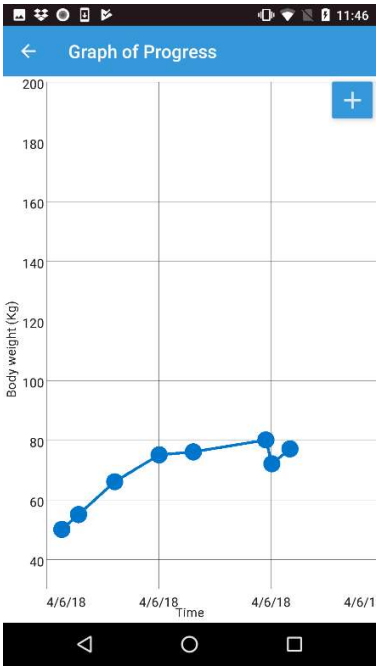
Graph Activity

The Graph activity allows for a user to track their body weight. There must be at least three entries for a graph to show.

Tapping the Add button in the top corner allows the creation of a new event. A dialog input is shown to take the weight input as kilograms.

Tapping the data points on the graph prompts the user to delete them.

The view of the graph can be changed by pinching to zoom on the horizontal axis. This means that a graph can continue to be plotted for long periods of time.



Actual Implementation

UI Draft

Technical Architecture

The architecture of the application can be broken into three layers: Presentation, Business and Data.

Presentation Layer

The presentation layer is formed of the XML files created in Android Studio. It provides a specification for the user interface (Meir, 2012). The presentation layer receives user input, as well as providing feedback.

Business Layer

The business layer handles the internal logic of the application, and uses the interactions given by the user in the presentation layer. It also interacts with the data layer to send and retrieve data from the database. It is made up of the Java files. The business layer is also responsible for interacting with the application framework layer of the android platform (Rogers *et al.,* 2009).

Data Layer

As recommended in the feedback for the proposal document, Firebase was used for persistent cloud storage. Firebase is a web and mobile application development platform owned by Google (Tamplin, 2012). It provides a real-time online database and user authentication almost entirely through using client-side code. Firebase stores data in a JSON format, which reduces the time and effort that would be required to create a traditional relational database using other means. As both Firebase and the Android platform are heavily supported by Google, they are subsequently designed to be compatible with each other. As a result of this, there is a vast pool of online resources available to assist with the creation of a Firebase database for Android.

Firebase has been used for both Google authentication as well as email and password authentication. By using Firebase, all usernames and passwords are safely stored automatically, complying with data protection laws (Gov, 2018). As a result of this, it was not necessary to manually salt and hash passwords.

Mobile-Cloud Ecosystem

Firebase eliminates the need to host a server as all external communication is done using the Firebase API.

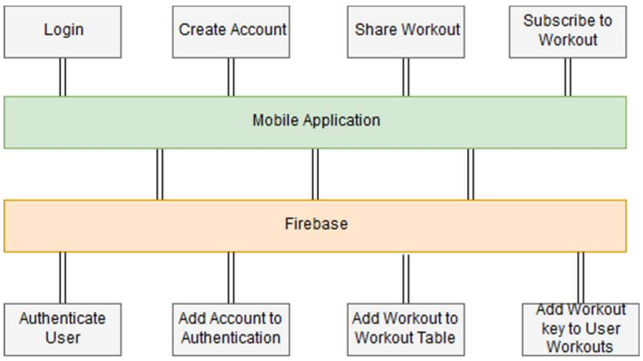


Figure 2: High level architecture diagram showing how the application layers interact.

Deployment Diagram

The application has been deployed in a simple client-server model. The business and presentation layers are stored on the client device, user data is retrieved remotely from the database.

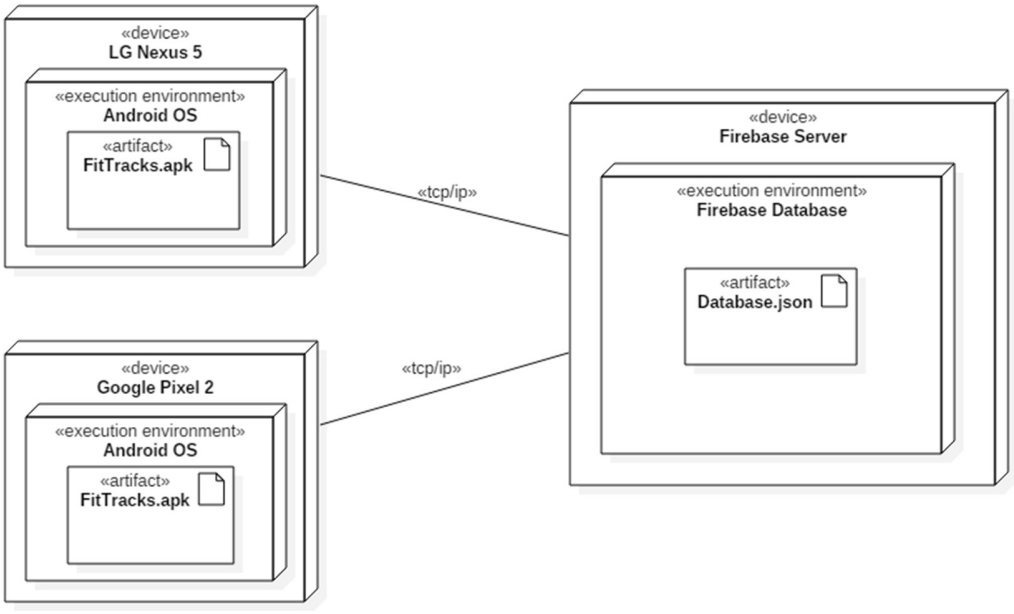


Figure 3: UML deployment diagram of the application. It shows two clients communicating with the Firebase database. FitTracks is not limited to two concurrent users.

Sequence Diagram to Create a Workout

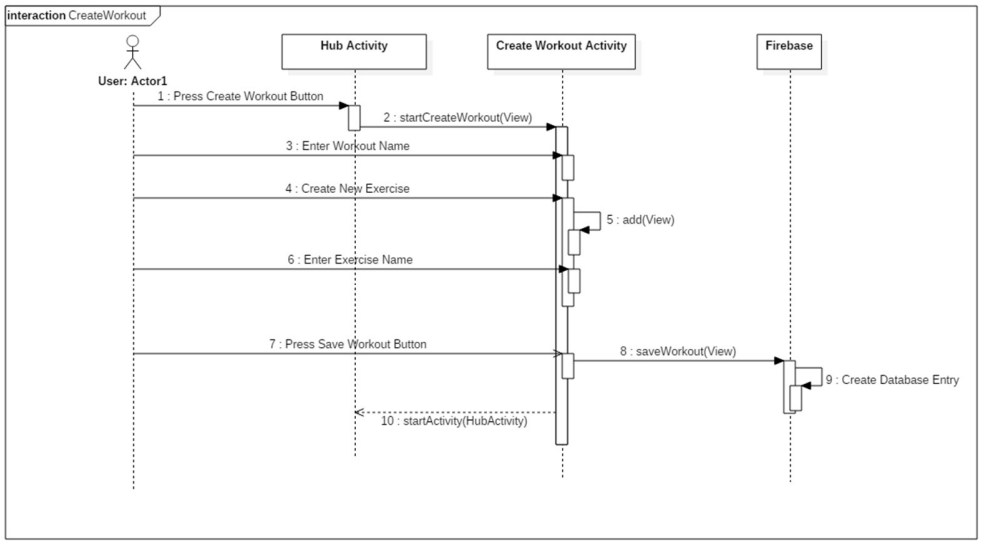


Figure 4: UML sequence diagram showing how a workout is created. It shows how the layers of the architecture work together to Achieve a goal.

Technical Challenges

Optimisation

Whilst the performance of mobile devices has increased in recent years, they are still generally slower than desktop and laptop computers. The largest factor limiting their performance is their size, which has a negative effect on battery life (Balasubramanian *et al.*, 2009). This means that modern mobile applications need to be highly optimised to use as little processing power, and therefore as little battery life as possible (Carrol and Heiser, 2010).

To ensure that these standards have been met, the code was optimised by removing any unused library importations and unnecessary object declarations. Removing object declarations will free up application memory, and reduce the amount of garbage collection required (Jones and Lins, 1996) (Agesen *et al.*, 1998).

Bugs and Solutions

Slow UI

Towards the end of development, a bug was found whereby the UI would dramatically slow down, causing jank after approximately 1 minute of use. This could be seen in the Android monitor with the CPU usage at between 40-50 per cent and memory usage at 130mb-160mb when the application was idle. This is shown in Fig 3 and Fig 4. CPU usage should be close to 0 per cent when idle, unless a background task is executing.

The cause of the problem was a database listener in the HubAdapter class, which was connecting to the database each time a new CardView was created when populating the Hub activity. This was causing unnecessary method calls to retrieve from the database, slowing down the UI.

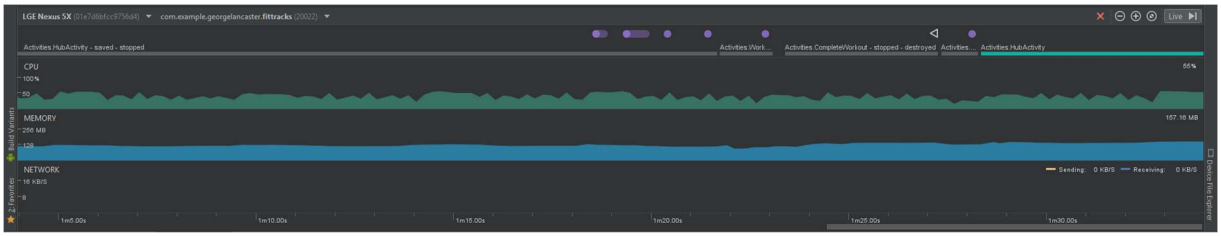


Figure 5: System use with the bug when the application is idle. CPU usage is constantly high.

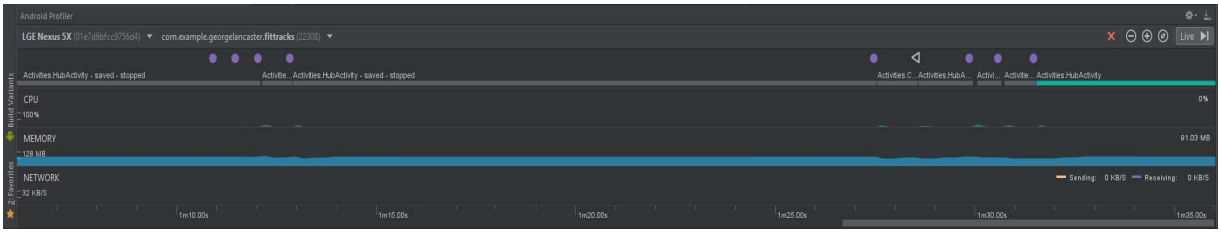


Figure 6: System use when fixed. CPU usage is 0, or close to 0 when the application is idle.

Table 1: CPU use, memory use and network use before and after fixing the bug.

Metric	Bug	Fixed
CPU use	56%	0%
Memory use	157.16mb	91.03mb
Network use	0Kb/s	0Kb/s

The bug was fixed by removing the database listener from the HubAdapter class, and placing it in the Hub activity, so that the data was only retrieved once.

Race Condition

Another bug occurred in the Graph activity. It was caused by a race condition between adding a new body weight event to Firebase, and retrieving the event at the same time in a different thread. This was caused by the use of an addValueEventListener, which runs asynchronously to the main thread and is called when any changes are made to the database.

An event is made up of two pieces of data; the body weight and the upload time. When added to the database, it makes two changes, and therefore the addValueEventListener is called twice. As the upload time was submitted first, the first call was causing a null pointer exception when receiving the weight, which had not yet been uploaded in the other thread.

This was solved by assigning the weight variable a value if it is null, before retrieving the true value from Firebase in the second method call. This meant that the second call of addValueEventListener could retrieve both correct values.

Bugs caused by concurrency are notoriously difficult to fix, and the solution to this bug was found purely by chance; only after it was solved did the cause of the problem present itself.

Testing and Evaluation

Requirements Testing

Table 2: contains descriptions of the tests carried out, stating whether the user requirements defined in the proposal document have been met, and gives evidence that the test has been passed.

Requirement Tested	Requirement Description	Test Description	Success Criteria	Pass/Fail	Evidence
1	The application must have three sections: Hub, Track and Workout	Tapping the navigation.	Tapping the navigation will navigate to the specified tab.	Pass	Appendix a
2	A user must be able to sign-in to the application.	Entering a valid email address and password into the login activity.	The hub activity starts and the user is authenticated.	Pass	Appendix b
2.1	A user must be able to sign in with a Google account.	Tapping the Google Sign-in button.	Tapping the Google Sign-in button will authenticate using Google.	Pass	Appendix h

Requirement Tested	Requirement Description	Test Description	Success Criteria	Pass/Fail	Evidence
3	Users must be able to sign out of the application.	Tapping the logout menu item in the app bar menu.	Tapping the logout menu item in the app bar menu will log the current user out, and navigate to the login screen.	Pass	Appendix c
4	Users must be able to track their bodyweight in the Track tab.	Tapping the Create Activity button in the Track Graph activity.	A new body weight event is created in the Track Graph activity.	Pass	Appendix i
5.1	Users must be able to view their previous workouts in the Track activity.	Tapping on the Track activity to show the calendar.	Completed workouts will appear in the Track tab.	Pass	Appendix d
5.2	Users must be able to track their personal records in Track activity.	N/A	N/A	Fail	None
5.3	Users must be able to track bodyweight.	Tapping the plus button in the Graph activity.	Tapping the plus button in the Graph activity creates a new body weight event.	Pass	Appendix i
6	The Graph activity must create a graph.	Tapping the graph button on the Track activity.	A new activity is started showing the Graph activity.	Pass	Appendix i
7.1	Users must be able to create workouts.	Enter in details of a workout in the Create Workout Activity and press Create Workout.	A workout is created and added to the database. This is shown in the My Workouts tab and can be subscribed to on the Browse Workouts tab.	Pass	Appendix e
7.2	Users must be able to delete workouts.	Press the delete button on the Hub Activity on a workout owned by the user.	The workout is removed from the database.	Pass	Appendix f
7.3	Users must be able to modify workouts.	N/A	N/A	Fail	None

Requirement Tested	Requirement Description	Test Description	Success Criteria	Pass/Fail	Evidence
8	Users must be able to create exercises.	Any new exercise can be entered when creating a workout.	The user can create a new workout using any new exercises.	Pass	Appendix e
9	Users must be able to share workouts.	Creating a workout will make it publicly available for other users.	Workouts created by one user will show in the 'Browse' tab for other users.	Pass	Appendix g
10	Users must be able to download workouts.	Tapping on the subscribe button of a workout in the Browse Workouts tab.	Tapping on the subscribe button will cause a workout to appear in My Workouts.	Pass	Appendix g
11	Users must be able to log workouts in real-time.	Tapping on a circle set icon in the Complete Workout tab marks it as complete.	When completing a workout, tapping the sets circle will mark it as complete.	Pass	Appendix d
12	Users must be able to submit completed workouts.	Tapping the Finish Workout button in the Complete Workout tab will add the workout to the calendar.	Tapping the Finish Workout button in the Complete Workout tab adds the workout to the calendar.	Pass	Appendix d

Some of the user requirements were not met due to the scope of the application being too large for a one-person project. The most important requirements have been met, ensuring that FitTracks functions as a fitness tracker. Workouts can be created, deleted, shared and tracked and a user's bodyweight can be plotted in a graph. In total, 15/17 of the user requirements specified in the proposal document have been met.

User requirement 5.2 has failed because a user cannot track their personal records in the Track activity. This was not possible to implement as there was not enough time in the project. A possible implementation could have been done using tabs to show a graph for each type of exercise.

User requirement 7.3 has failed because a user cannot currently modify an existing workout. However, they can delete the workout and create a new one. This would have involved creating a new activity similar to the Create Workout activity, and populating it using data from an existing workout.

Performance

To test performance, the Android Software Development Kit (SDK) comes with a command-line tool called the Android Debug Bridge, which can be used to measure application performance. A summary of the frames used after full path coverage and with an up-time of 10 minutes are shown.

```
** Graphics info for pid 29211 [com.example.georgelancaster.fittracks] **

Stats since: 77554043132609ns
Total frames rendered: 3729
Janky frames: 619 (16.60%)
50th percentile: 10ms
90th percentile: 18ms
95th percentile: 19ms
99th percentile: 29ms
Number Missed Vsync: 29
Number High input latency: 0
Number Slow UI thread: 46
Number Slow bitmap uploads: 0
Number Slow issue draw commands: 567
```

Figure 7: A summary of the frames used after full path coverage and with an up-time of 10 minutes are shown using the command:
adb shell dumpsys gfxinfo com.example.gerogelancaster.fittracks

This shows that from the start of the program, 16% of frames have been rendered with some jank. Although this number seems high, the UI was not noticeably slow.

Test Lab

Firebase gives access to Test Lab, an automated cloud-based testing suite (Firebase, 2018). Test Lab uses an intelligent algorithm to explore the full path coverage of the application. During the test, it measures the CPU, memory and network usage.

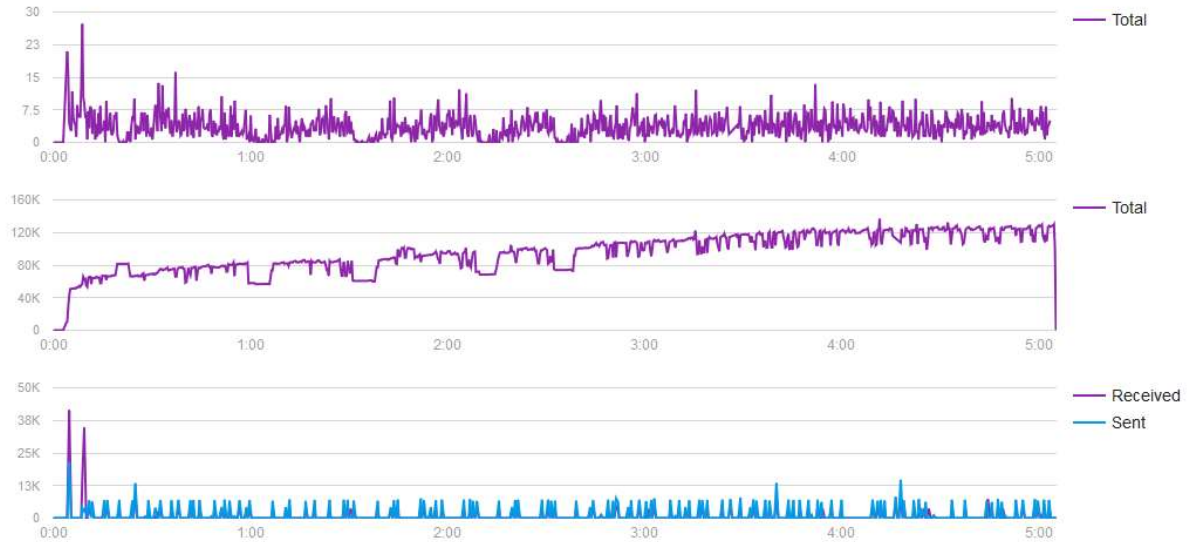


Figure 8: CPU usage, Memory usage and Network usage from the Test Lab.

Table 3: peak CPU and memory use from the Test Lab test.

Peak CPU Use (%)	Peak Memory Use (Kb)
28	130008

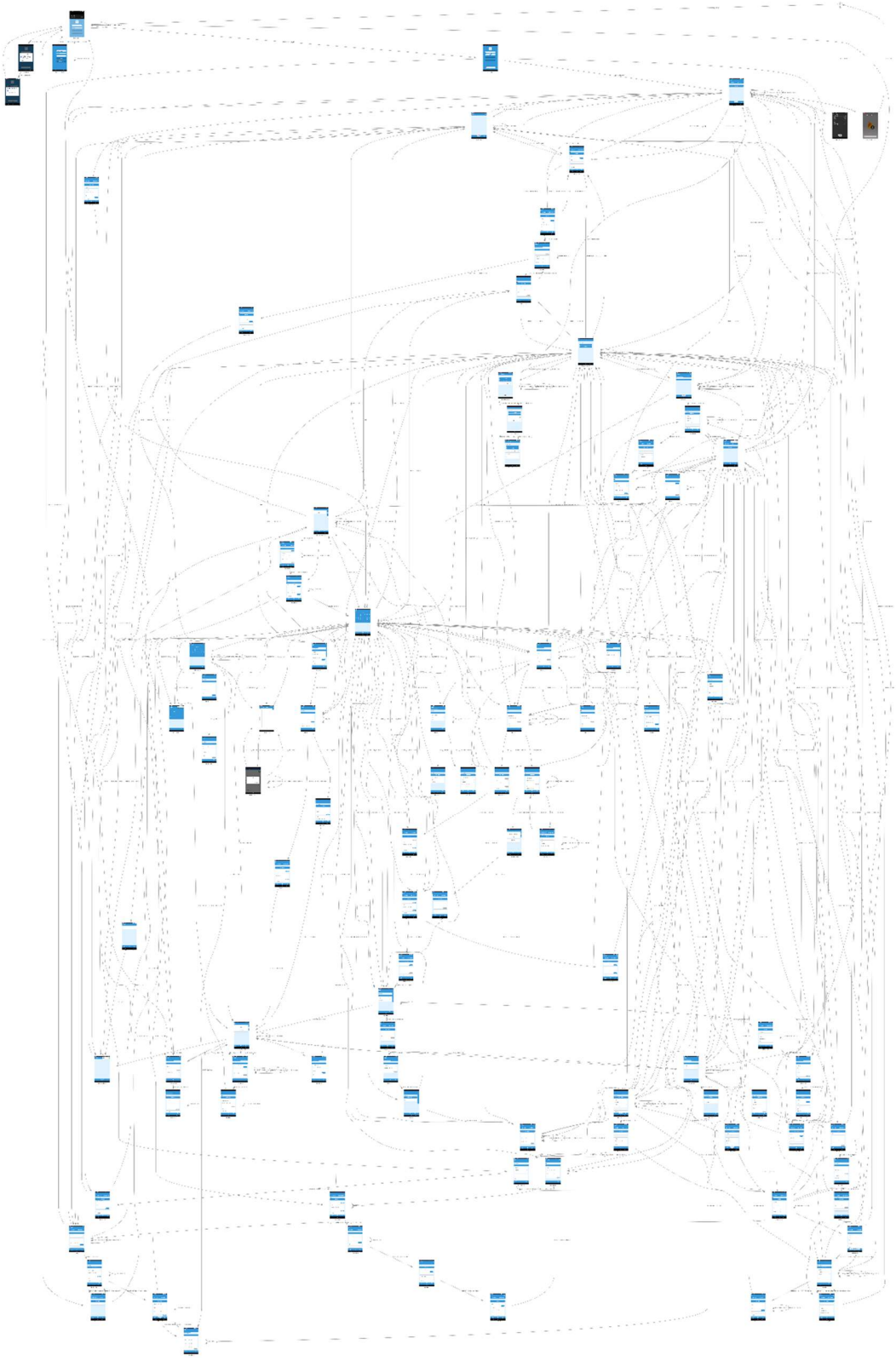


Figure 9: Shows the path coverage of the Test Lab test.

Development Methodology and Project Management

The working schedule allowed at least one full day a week working on the project, which was usually every Tuesday before, during and after the code clinic. Although this was the regular schedule, development was not limited to these days and times.

This project was completed using the agile methodology. Each working day was tracked in a diary, which allowed for weekly planning of activities. A copy of the diary can be found in appendix j. Although the diary gives a good indication into the days worked on the project, it is not an exhaustive list as a diary entry was not always made. The diary ensured that there was always a plan for work to be completed and it gave a good indication as to how the project was progressing.

The progress of development was not linear as activities were not completed in any order. This meant that improvements were constantly being made to previous implementations. Whilst this made it difficult to plan the project, it meant that the code was written to a higher standard using guidelines provided online by the developers of Android.

Gantt Chart

A Gantt chart showing the development process can be found in appendix k.

Milestones and Deliverables

Aside from the two compulsory deadlines, several other milestones were set to ensure that work on the project was consistent throughout the year.

The table below shows the predicted date of task completion against the actual date the task was completed. The table indicates that the project was completed faster than expected as all milestones were met before their predicted date.

Table 4: Predicted completion dates of tasks, and the actual date they were completed.

Predicted Date	Actual Date	Milestone
17/11/17	17/11/17	Hand in Proposal
10/01/18	07/12/17	Have a simple prototype working. This includes having three main views and a bottom navigation bar.
04/04/18	13/02/18	Have the application in a state ready for release.
04/04/18	03/03/18	Begin work on the final report
03/05/18	30/04/18	Finish report.
04/05/18	04/05/18	Hand in final version with report.

GitHub

GitHub was used for backup, and for version control when working on different computers. The use of GitHub ensured that there was no loss of work, and that the correct version was consistently being worked on.

Discussion

Learning Points

Technical Skills

A lot of development time was spent learning how to use the Android SDK effectively. Because of this, a lot of the code written in early development had to be re-written to use the classes and libraries the Android SDK provided. An example of this is creating a list of items using the RecyclerView, as has been done in the Hub, Workout, Complete Workout and Track activities. The original implementation was done by appending objects to a vertical linear layout. Whilst this worked, it is not the proper method supported by the developer documentation, and had to be changed.

During the development process, skills had to be improved in the usage of the mark-up languages XML and JSON, having only had limited experience of them in the past. It was important to learn how to write XML as directly editing layouts using XML gives greater flexibility than using the "What You See Is What You Get" editor. Having an understanding of JSON meant that the database could be designed effectively.

It was impossible to learn the entirety of Android programming in the amount of time given to complete this project. Online resources such as the Android Developer website and Stack Overflow proved invaluable to the success of the project, providing answers to very specific questions unique to Android programming.

Designing an application for mobile is vastly different to designing an application for larger screens. It is important to make the most of the space available whilst keeping the user interface de-cluttered. A balance must be found to show the right amount of information for a given activity, which was a learning point in itself.

Project Management

The experience of working on this project has been valuable in the development of time-management skills, as well as learning the importance of documenting progress throughout a project. Given the time that was available to complete this project whilst also being required to complete other assignments, it was important to divide time equally amongst tasks and to prioritise the most important requirements.

Strengths

FitTracks is a very robust application, there are currently no known bugs which cause it to crash. Due to its design, all user data is secure and will not be lost once submitted to the database in the case that the app does crash.

The user interface remains simple through using icons to represent the functionality of some buttons, which provide feedback when pressed by changing colour. FitTracks is not restricted to any demographic, and has been designed to be suitable for technologically inexperienced users. However, some familiarity of fitness terminology is required, such as the use of 'sets' and 'reps'.

FitTracks makes use of the limited screen size of the mobile platform. The views within activities have space between them to show that they are separate entities, without wasting areas of the screen on blank space. An example of this being the space between CardViews.

Weaknesses

There is a limited use of animations, they are currently only used to switch between the three top-level destinations. Animations do not affect the performance or functionality; however they can be used to give a feeling of 'polish' to a user.

Although previously mentioned in this report, fragments could have been used in the bottom-navigation. This could improve the UI by fixing the bottom-navigation in place as well as reducing the loading time when switching between activities.

The size of the text throughout FitTracks cannot be changed; this could affect the accessibility of the application for users with visual impairment.

FitTracks measures weights in kilograms. This does not have any bearing on how the application performs, but some users may wish to measure weight using pounds.

Further Improvements

Record Tracking

Further developments could be made to the Graph activity. Currently, only the bodyweight of the user is tracked. This could be further improved by tracking personal records of exercises. A possible implementation could be done through using fragments on the Track Graph activity to switch between graphs of different exercises. A user's records would be saved the Firebase database in an Events branch of the JSON, much like how it is done for tracking body weight.

Search Bar

Currently, the search bar on the Hub activity is not working. It was intended to be used to search for workouts on the current tab by the workout name. This could have been implemented through using the Query class from the Firebase API to filter out searches using the String provided in the search bar EditText.

Conclusion

From a project management viewpoint, this project could be considered to be successful. Most of the user requirements have been met on time, and many goals were completed before their deadlines. There are very few bugs, and the ones that are present are only cosmetic.

FitTracks could be released to the public, but it is more likened to an application in its beta stage of development. Most of the issues holding it back are not in its design, but additional functionality and 'polish' which could have been added given more development time.

The UI of FitTracks was heavily influenced by other applications, and the current standards for mobile application design. This influence meant that there were no risks taken in terms of its usability.

References

Ableson, F., Sen, R., King, C. and Ortiz, C.E., 2011. *Android in action*. Manning Publications Co..

Agesen, O., Detlefs, D. and Moss, J.E., 1998, May. Garbage collection and local variable type-precision and liveness in Java virtual machines. In *ACM SIGPLAN Notices* (Vol. 33, No. 5, pp. 269-279). ACM.

Balasubramanian, N., Balasubramanian, A. and Venkataramani, A., 2009, November. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement* (pp. 280-293). ACM.

Carroll, A. and Heiser, G., 2010, June. An Analysis of Power Consumption in a Smartphone. In *USENIX annual technical conference* (Vol. 14, pp. 21-21).

Carroll, J., Howard, S., Vetere, F., Peck, J. and Murphy, J., 2002, January. Just what do the youth of today want? Technology appropriation by young people. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* (pp. 1777-1785). IEEE.

Developer. 2018. *Testing UI Performance*. [ONLINE] Available at: https://developer.android.com/training/testing/performance.html. [Accessed 5 April 2018].

Firebase. 2018. *Firebase Test Lab for Android*. [ONLINE] Available at: https://firebase.google.com/docs/test-lab/. [Accessed 27 April 2018].

FitNotes. 2018. *Apps.* Retrieved from FitNotes [ONLINE] Available at: http://www.fitnotesapp.com/ [Accessed 27 April 2018].

Gehring, J., 2018, GitHub repository, https://github.com/jjoe64/GraphView

GOV.UK. 2018. data protection. Retrieved from gov.uk [ONLINE] Available at: https://www.gov.uk/data-protection

Hoober, S., 2013. *How do Users Really Hold Mobile Devices?*. [ONLINE] Available at: https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php. [Accessed 5 April 2018].

Hurff, S., 2014. *How To Design For Thumbs In The Era Of Huge Screens*. [ONLINE] Available at: https://www.gizmodo.com.au/author/scott-hurff. [Accessed 27 April 2018].

Jones, R. and Lins, R., 1996. *Garbage collection: algorithms for automatic dynamic memory management* (Vol. 208). Chichester: Wiley.

Material. 2018. *Bottom Navigation*. [ONLINE] Available at: https://material.io/guidelines/components/bottom-navigation.html. [Accessed 5 April 2018].

Meier, R., 2012. *Professional Android 4 application development*. John Wiley & Sons.

Pirc, J. 2012. lets get physical and digital. Available at: http://blog.lab42.com/lets-get-physical-and-digital/. [Accessed 5 April 2018].

Rogers, R., Lombardo, J., Mednieks, Z. and Meike, B., 2009. *Android application development: Programming with the Google SDK*. O'Reilly Media, Inc..

Stronglifts 5x5. 2018. *Apps*. Retrieved from Stronglifts [ONLINE]
Available: https://stronglifts.com/apps/ [Accessed 27 April 2018].

Sundeep, K., 2018, GitHub repository, https://github.com/SundeepK/CompactCalendarView

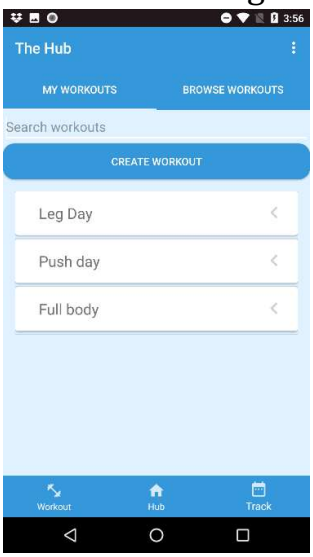
Tamplin J. 2016. *Firebase expands to become a unified app platform*. [ONLINE] Available at: https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html. [Accessed 11 April 2018].

Valenzuela T, O. Y. 2010. Adherence to Technology-Based Exercise Programs in Older Adults: A Systematic Review. Journal of geriatric physical therapy, 1.

Voth EC, O. N. 2016. A Theory-Based Exercise App to Enhance Exercise Adherence: A Pilot Study. JMIR Mhealth Uhealth, 4(2):e62.

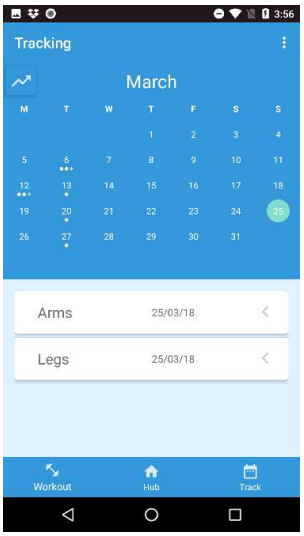
Appendix

Appendix a: Bottom Navigation



Before

User is located on the Hub Activity.



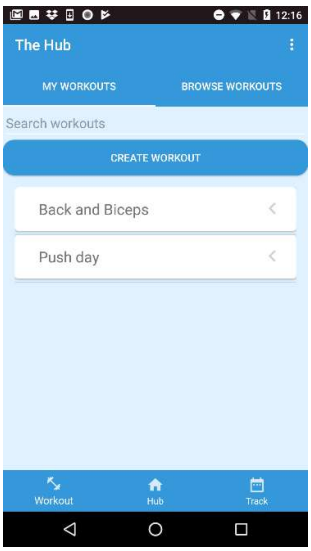
After

Navigation is changed to the track tab.

Appendix b: Login

Before

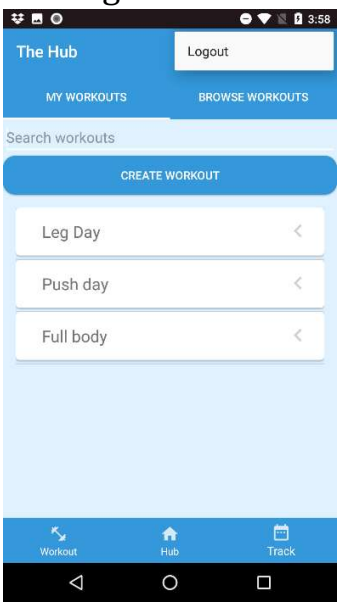
Valid username and password combination have been entered.



After

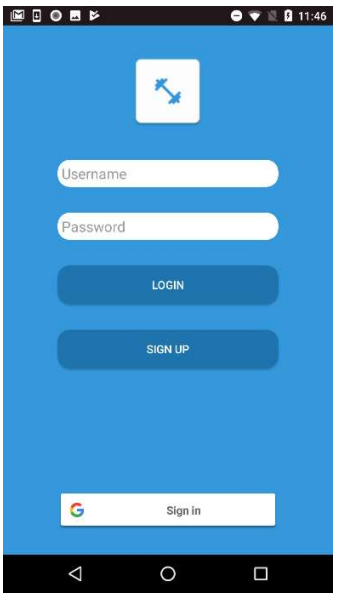
Hub Activity is shown and user is authenticated. This is shown by the correct workout being displayed.

Appendix c: Logout



Before

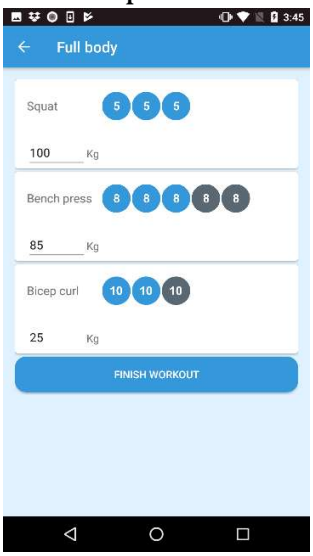
Tapping the logout button.



After

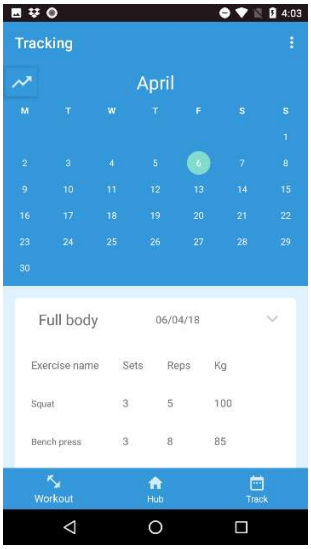
User is de-authenticated and returned to the Login Activity

Appendix d: Complete a Workout



Before

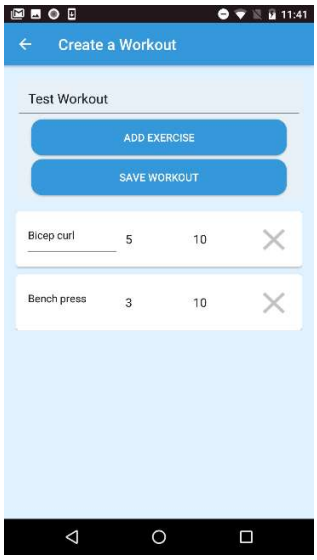
Tapping the Finish Workout Button.



After

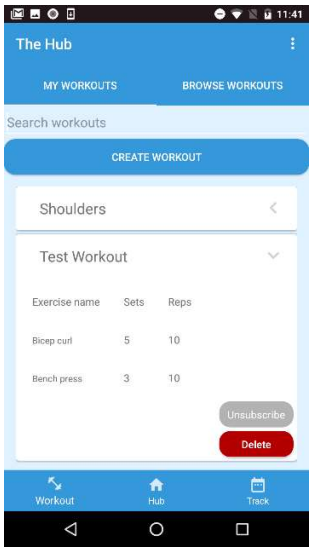
The completed workout can be seen in the Track Activity.

Appendix e: Create a Workout



Before

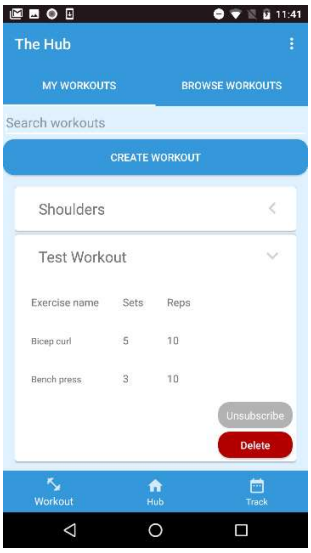
Enter details of the workout.



After

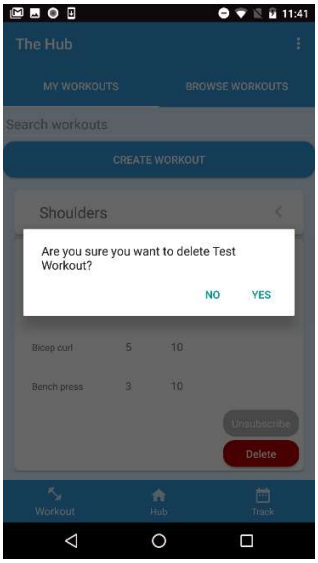
Workout is saved to My Workouts.

Appendix f: Delete a Workout



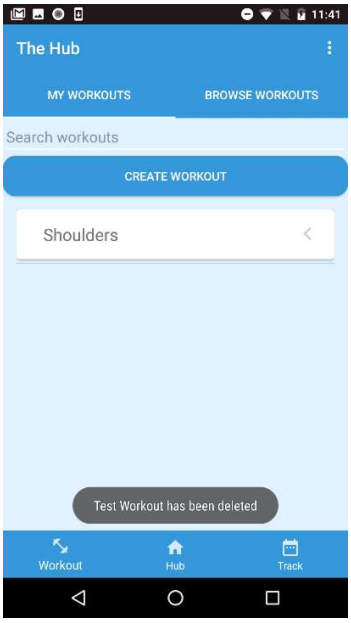
Before

Delete button can only be seen if the user owns the workout.



During

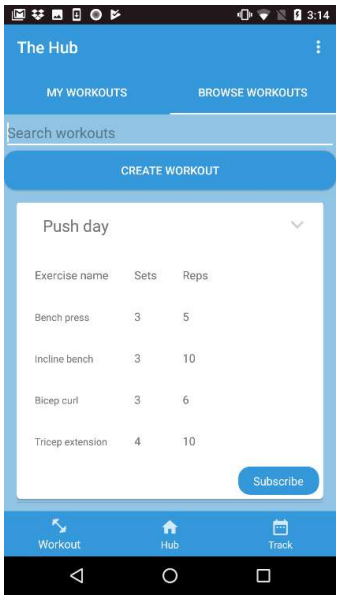
On the button press, a confirmation dialog box is shown.



After

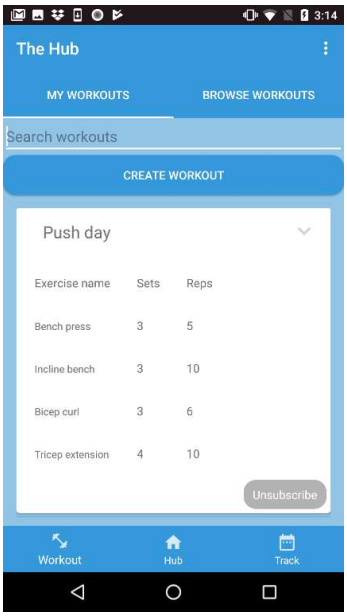
The workout has been deleted.

Appendix g: Subscribe to Workout



Before

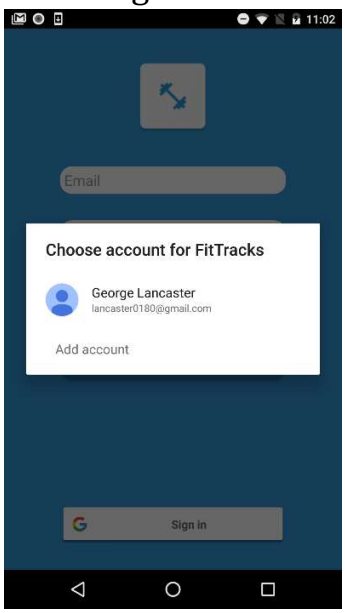
Press the subscribe button.



After

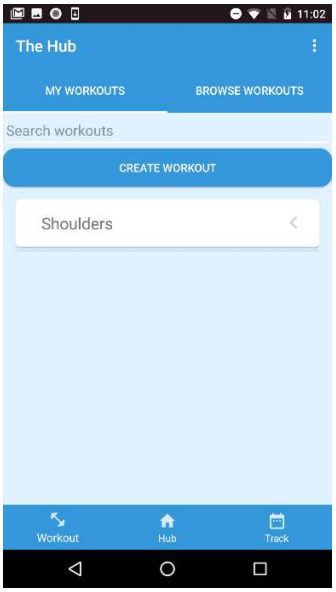
Workout appear in the My Workouts tab.

Appendix h: Google Authentication



Before

Select a Google account for authentication.



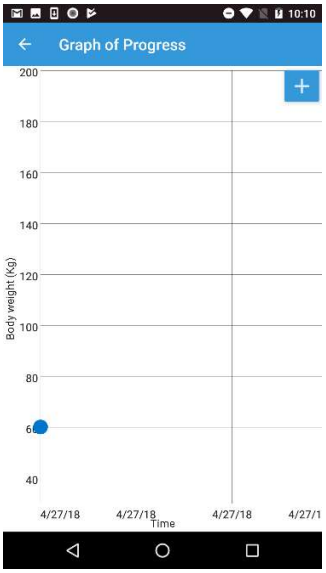
After

Workout has been saved to the calendar.

Appendix i: Graph Tracking

Before

Enter weight in kilograms.



After

Weight has been saved to the database and displays in a graph.

Appendix j: Work Diary

#Work Diary

07/12/17 ----------

Created the project file. 4 Activities and a simple dropdown menu to navigate between them. Login view, Workout hub, Tracker view, Graph view

14/12/17 ----------

Connected the project to firebase. I can add and modify simple data in the database.

24/12/17 ----------

Have done basic email authentication. Can login with email and password. Still quite buggy, not sure if it is authenticating properly. Cannot create users using the app - will do this at a later date.

#TODO

Need to integrate sign-in for google etc.

03/01/18 ----------

I have the ability to create workouts and save them to the database. I have done this by creating a HubView which shows the users workouts and the list of workouts online. This implementation is crude and will need updating in the future.

04/01/18 ----------

I need to add the ability to sign into the application.

This includes:

1. Google sign-in
2. Email sign-in
3. Application is personalised for each user.

At the end of the day, Users can now login with email address and save workouts and take ownership of them.

Some of the exercises can be retrieved from the database, however this is still not working 100%.

08/01/18 ----------

Fixed the errors I had on 04/01/18. The WorkoutHub now displays the workouts from the database in a ListView. Looks fairly neat, however I will change this at a later date.

I need to separate workouts into MyWorkouts and Browse Workouts as it is currently displaying all workouts, when it should only display the Users workouts. I will have to implement some kind of query to acheive this.

11/01/18 ----------

Added a RecyclerView to the WorkoutView to display some workouts. This involved the implementation of a WorkoutAdapter class to display the workouts linearly. Currently, the RecyclerView is displaying dummy data, however i'm sure it will not be difficult to display the real data as I have done it in a listview for the WorkoutHub.

To improve the WorkoutHub I may change the afforementioned ListView into a RecyclerView as it does look a lot neater.

Very happy with progress so far and I seem to be way ahead of schedule.

15/01/18 ----------

Look at WorkoutView and started creating a dynamic/intereactive workout session based on the workout selected.

21/01/18 ----------

With a lot of the functionality there, I now need to begin improving the aesthetic of the application.

Today I am working in GIMP to create some custom graphics for the app.
I have created custom checkboxes for the workout view. These have a number on them corresponding with the numbers of reps to perform for the set.
I have added the graphics to the complete workout activity.
I have also made some changes to the create workout activity, using cards rather than my previous crude implementation.

27/01/18 ----------

I have updated the login view and created a graphic for the login view. It is the word 'FitTracks' with a dropshadow.
It does look rather crude, so I will need to change this at a later date.
The rest of the login view now looks a bit better, howver it is still not completely finished.
I need to create a sign-up view.

29/01/18 ----------

I have created another cardview for the selection of a workout. It is not finished yet. One problem I am having with the selectworkout view is that the theme is dark, I cannot change it, even in the manifest. This will require further investiagtion.

06/02/18 ----------

Animated the transitions between activities. I still need to implement a bottom navigation.

07/02/18 ----------

I have created a bottom navigation. I have not decided wether or not to use fragments or stick with my current plan of using activities for all areas of the app.
It may take up too much time to re-write large parts of the app to cater for fragments

rather than activities.

I have also made some changes to the back-stack, so that a user cannot logout then continue to use the app.

12/02/18 ----------

Made some changes to the hub view. Using a cardview and with recyclerview rather than a listview. Looks much better. Need to finish the cardview to make it look neater.

Also changed the database to make each exercise have multiple subscribers, rather than a user have muliple workouts. It seems to work better that way as users are found when getting a workout anyway so we only need to contact the database once, thus speeding up the app.

17/02/18 ----------

Continued work on the cardview on the hubview. The cardview is now collapsable and displays entire workouts.

20/02/18 ----------

Have implemented the ability to subscribe and unsubscribe from workouts in the hub. Subscribed workouts now show up in the My Workouts tab on the hub view. They also show up as subscribed when on the browse view. A user can unsubscribe from a workout from either view.

I have also found a bug whereby the application slows down after a few minutes of use. I am not completely sure what is causing this. CPU usage does not exceed 60%, however memory usage can climb when interacting with the track view. Maybe a different implementation is required.

03/02/18 ----------

Finally found the cause of the bug. In my HubAdapter, I was accessing the database for each item in the list to see if the workout had been subscribed to or not.
This caused a massive amount of jank in the UI. I will need to find another way of doing this.

04/02/18 ----------

Created a sign-up activity. Can add new users to the database using this activity.
It is accessible from the

06/02/18 ----------

Fixed the hub activity so can now subscribe and unsubscribe from workouts.

12/03/18 ----------

Made an adapter for the track view, rather than inflating layouts as done in the previous implementation. I also wrote a little about user experience in the report.

Need to start wrapping up the app and moving on to the report.

13/03/18 ----------

Added in a few cosmetic changes to some of the card views.

17/03/18 ----------

Worked on testing and evaluation. I have found firebases Test Lab which uses AI to navigate the application. I have also continued work on the report which is now at 20 pages and 2000 words.
Although the app is not 100% complete. Most of the required features are there, any changes I now make are a bonus.
Things I can add are:

1. Search workouts using search bar.
2. Show placeholders for new users with no workouts.

20/03/18 ----------

Continued work on report

25/03/18 ----------

Continued work on report as well as making some minor changes to application.

Removed the app-bar from the sign-up activity.

Added the ability to track weight. Should have done this a while ago, but did not take that long.

Changed some of the buttons to red rather than blue to signify that they are deleting.

Users can delete a workout is they have ownership of it, no one elses.

27/03/18 ----------

Continued work on report, also made some small changes to app, login screen etc.

30/03/18 ----------

Continued work on report, mostly finished, needs some more on mobile architecture and references.

03/04/18 ----------

Added the graph to the Track activity, still a little bit buggy. Can finish this later.

Also added Google sign in.

Re-wrote some parts of the report. It is now at 4266 words excluding appendix.

Most user requirements are now met, once I fix the bug on the Track graph activity I will be happy with to submit the application.

06/04/18 ----------

Fixed the graph activity bug. Can upload and delete body weight events.
Application is finished. Will not make any more changes unless I notice a bug.
Report is 95% finished, just need to update screenshots.
report is now 5496 words.

24/04/18 ----------

Made some changes to UI. When loaded, the Hub activity now displays a message to say that the workouts are loading, this message disappears when they have finished loading. Have also rearranged the classes into new packages. Will create a package diagram for the report.

27/04/18 ----------

Made some final tweaks to the application and read through the report. Made some corrections and re-wrote parts that needed it.

30/04/18 ----------

Finalised everything. Am going to go to the android session tomorrow to ensure I haven't missed anything and submit in the evening.

01/04/18 ----------

A little bit last minute but have found an authentication bug. Not sure why it is happening now, will have to fix tomorrow.

02/05/18 ----------

Have fixed the bug, was a small problem with users not being correctly logged out. Will submit tomorrow.

03/05/18 ----------

Generated signed APK and submitted.

Appendix k: Gantt Chart