

Android Gym Application Utilizing Near Field Communications (NFC) CM0343

FINAL REPORT
BRETT STEVENS (0931855)

Contents

Abstract	3
Table of Figures	4
Introduction	5
Project Goals revisited	5
Specification	6
Specification summary revisited from interim report.....	6
Technologies that I will be using to implement my application.....	6
Processes that I intend the application to be able to carry out.....	6
Initial Database design	6
Design.....	7
Initial Design.....	7
Application Design	9
Package Explorer for the Gym Diary Application	9
Key areas utilized in development	10
Approaching the Interface	11
Defining objects that are used in Gym Diary xml's	12
What Objects are in each XML page	12
Implementation	16
Test device.....	16
LogCat	17
NFC	18
SQLite database	20
Calendar	21
Stop watch.....	23
Results and evaluation	24
Key Features of Gym Diary.....	24
Evaluation.....	29
Limitations of NFC.....	30
Problems during the project	30
Error Checking.....	31
Testing Gym Diary Objects Function as intended	37
Home screen –Activity_main.xml	37

Scan tag - Activity_nfc.xml	37
Add entry to Diary - Activity_db_add_entry.xml	37
Manual add to diary - Activity_db_add_entry_manual.xml	37
Go to database page - Activity_db.xml	38
Go to calendar view - Activity_calendar.xml	38
Selecting stop watch - Activity_stopwatch.xml	38
Selecting help page - Activity_help.xml	38
Activity_db_add_exercise.xml	38
Activity_db_edit_exercise.xml	38
Activity_db_view_exercise.xml	38
Testing Continued	39
Critical appraisal of results	42
-Adding records to the database from an NFC Tag	43
-Editing / deleting/ viewing exercise records	43
- Stop/start timer	43
Potential testing in a Gym environment	44
How has the project matched initial designs?	44
Future Work	45
Possible improvements	45
Conclusions	46
Reflection of learning	46
Table of Abbreviations	48
Appendices	49
References	49

Abstract

This report focuses on transferring initial designs into a real template which will be used to implement the building of the Android gym application utilizing NFC technology. This will also include extensive documentation and testing/evaluating the finalized application. Investigation will be made into potential future work in where the application could be extended to fit more complex scenarios.

Table of Figures

- Fig [1] - NFC Tags -own photo
- Fig [2] - <http://www.digitaltrends.com/mobile/samsung-galaxy-s3-impressions-maybe-it-is-designed-by-human-emotion/>
- Fig [3] process Flow of the system
- Fig [4] – Modular design android development environment
- Fig [5] – Xml layout
- Fig [6] – Package explorer
- Fig [7] – Manifest extras
- Fig [8] - Initial design for main page
- Fig [9] – Final design of application home page
- Fig [10] – Scan tag page XML- <http://www-static.se-mc.com/blogs.dir/0/files/2012/02/xperia-smarttag-touch-tag-smartphone2.png>- image for scan tag page
- Fig [11] – Add exercise XML
- Fig [12] – Add exercise manual XML
- Fig [13] – Database XML
- Fig [14] – Add exercise XML
- Fig [15] – Delete exercise XML
- Fig [16] – View exercises XML
- Fig [17] – Calendar View XML
- Fig [18] – Stopwatch XML- <http://www.clker.com/clipart-stopwatch-1.html> image for clock design
- Fig [19] – Help page XML
- Fig [20] – Screenshot of Eclipse IDE
- Fig [21] – LogCat debugger
- Fig [22] – Table of Initial requirements
- Fig [23] – Scanning tag process
- Fig [24] – Storing NFC data to the database
- Fig [25] – Adding exercise completed
- Fig [26] – Editing Database
- Fig [27] - Manual entry for exercises
- Fig [28] – Deleting exercise from database
- Fig [29] – Deleting entry's from calendar view
- Fig [30] – Adding entry's from calendar view
- Fig [31] – Stop watch States start/stop
- Fig [32] – Scan tag page empty tag
- Fig [33] - – Scan tag page malformed tag
- Fig [34] - – Scan tag page no NFC
- Fig [35] – Delete entry calendar view
- Fig [36] – Delete exercise from database
- Fig [37] -Pop up message deleting exercise
- Fig [38] – Initial design with final design.

Introduction

Project Goals revisited

The project is based on creating a more stream line approach to people recording their gym workouts by creating an android gym application utilizing Near Field Communications (NFC) technology.

A current problem that many gym users find is that when they wish to record their gym workout regime to a phone application they often find themselves spending more time looking for a particular exercise to add to the day's diary then they do focusing on the actual workout. If users have to manually search for exercises each time they wish to add to their diary this can work to a lot of wasted time in the gym. More often than not many exercises are not in a particular gym application.

The aim is to solve this problem is to create a versatile android application that has the ability to add and change database entries so users can also add their own exercises. By utilizing NFC technology you would be able to simply swipe your phone near an NFC tag that would be located on or near exercise equipment already be loaded with the required information the application would then add that exercise to the phone. Users would be able to go around the gym from one machine or exercise to another and with a simple swipe of the phone add that workout to the day's diary. This will make a gym users experience more streamlined as they will no longer be searching for exercises to add to their phone as they can now simply swipe their phone near an NFC chip to add the data to the diary.

The technology's that will be used are NFC tags, which are essentially very small memory devices with antennas attached this allows the transmission of data wirelessly. NFC tags have no internal power source to run, the power comes from the NFC reading device such as a mobile phone being in close proximity this comes from the radio signal which powers the chip inside the NFC tag. The tags can come in a variety of different shapes and sizes. More often than not a sticker as shown below. But can also be hidden in key fobs or plastic casings.



Fig [1]



Fig[2]

The device that the Android Gym application is being developed for is the Samsung Galaxy s3(as shown above).

The development Platform will be on windows using Eclipse IDE which has built in (ADT) Android Developer Tools, also needed will be the Android Software Development kit (SDK) which acts as a plugin to the Eclipse IDE.

Specification

Specification summary revisited from interim report

The specification is to produce a working android gym application that has the ability to read in NFC tags that contain data of exercise equipment into the application's SQLite database. Without the need for the user to type which exercise to add. Deliverables are a competent app featuring NFC technology and maintain key functional requirements.

The application will be navigated through a series of large buttons. Each menu button has series of sub commands which will interact with the integrated NFC technology or the SQLite database. Users will be able to store and retrieve planned workouts, edit the workouts as they see fit. Upon selected exercises they can select a timer in which to time the exercise they are currently doing.

Technologies that I will be using to implement my application

- Eclipse IDE + Android SDK.
- Java – Will be my primary language I will be using to code my application as this is the language that android uses.
- XML- Will be heavily used to implement the design and layout of the application.
- SQLite database - will be used to store all the records contained within the application.

Processes that I intend the application to be able to carry out.

- Scan AN NFC tag retrieve the text stored and parse into an SQLite database
- Make all database records accessible to the calendar.
- Manually add records without the need to use NFC.
- Stop/ Start timer to record time.
- Dynamic database that is easily editable.
- Error Checking systems in place

Initial Database design

I will need two tables, one labeled Exercises and another called Exercises table added.

The Exercises table will need to store ID, Name, Tag id.

This exercises table will be used to hold the data read in from the NFC tags. A unique ID is used so the application can distinguish between multiple records.

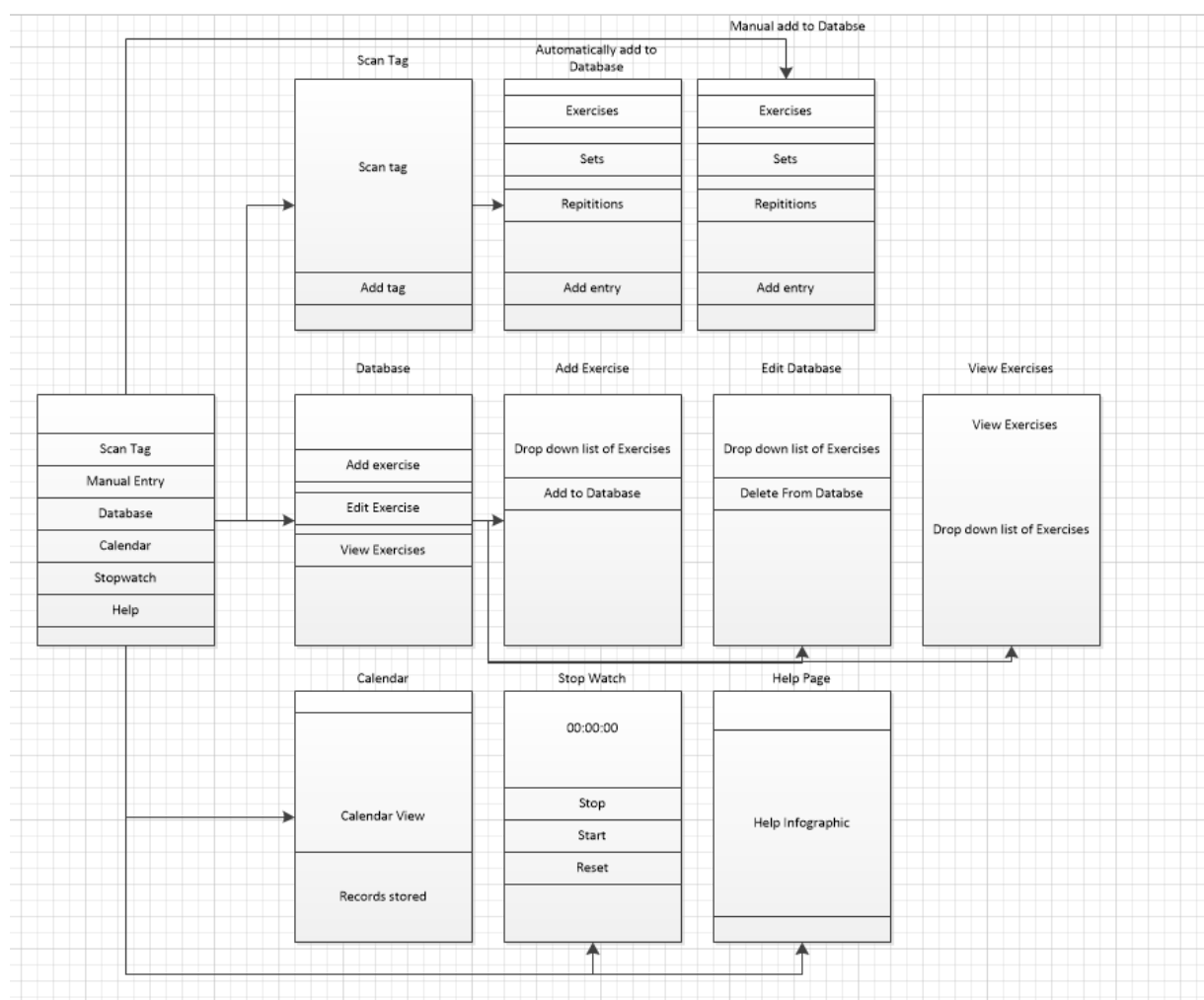
The Exercises table added will need to store , ID, exercise_id , sets, repetitions, weight and date added.

The exercises table added will store all additional records associated with the original Exercise stored. It will add new entry data to existing entries in the database.

Design

Initial Design

Taking into consideration feedback received from the questionnaires in the research phase. I chose to go with large on screen buttons, this will allow the application to be easily navigable for users with large hands and small hands alike. Before moving ahead with Implementation I have created an initial design data process of how the application could look. Fig [3] shown below is the process Flow of the system.



First was establishing how many different screens would be needed in order to keep the application as simple and intuitive as possible. I came up initially with the need for 11 screens that would be the minimum required to cover the functionality I intend to put into the application.

First the main page to hold the center of the application that would feature buttons/ shortcuts to key functionality within the application. They would feature a Scan tag button- this would navigate to the scanning of the NFC tag page. From this page it will then link in to adding of the exercise scanned into the database. This will feature the option for the user to enter additional information to be stored with the exercise scanned such as repetitions, sets and weight of the exercise.

Another Page I found would also need to be linked to the scanning of the exercise was the Manual add entries. This is because going back to the results from the questionnaires in the Interim report a certain number of android users did not have an NFC capable phone. So rather than exclude a section of the market just because their device is not NFC capable I decided to incorporate the manual entry also.

Another button on the main page would be Databases, this would link to the Databases page that would cover Adding/deleting exercises and also viewing exercises stored within the database. This page will also feature drop down menus called (spinners) this we be covered in the implementation section.

Calendar was another page that would need to be made as this would tell the user important information such as what exercises are on what day, and also allow the user to see what exercises were scanned that day. This page will feature a calendar view along with a list of records stored.

Stopwatch is another page that will be needed as feedback from the background research found that many users record time between sets of their exercises in order to achieve the best results possible. This page will feature a start, stop and reset button.

Last but not least I will include a help button. This page will show an info graphic of the technologies used and how they work, the reason for this is because not every user will have heard of NFC as it is still an emerging technology.

The application will be called Gym Diary as this encapsulates its intention within the android application environment as its primary design is to record gym exercises that day and keep a track of progress.

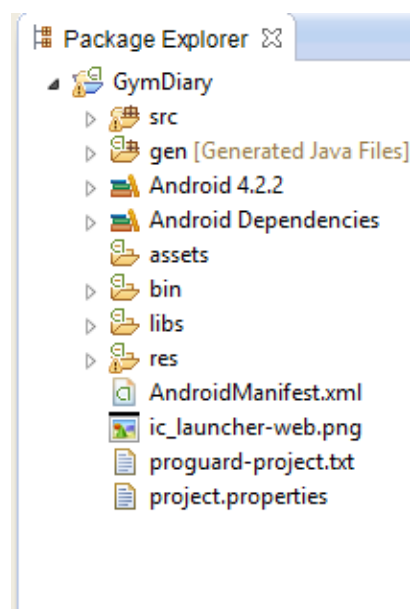
Application Design

Package Explorer for the Gym Diary Application

Before approaching a more in depth design, rules have to be adhered to in order to create your application within the android development platform. Android features a modular design architecture. This is due to if all the information (code) of an application were inside one file it could end up very messy for a developer to accurately determine what was going on, as when you start adding more pages and features the code used can be quite extensive. The modular nature divides development to a few keys sections. Fig [4] The image below shows some of the various modules.

All of these sections are integral for the development of the application, the src folder contains the java packages and class files. These are the “action” pages. This is where actions are applied to objects found in the xml pages, these are behind the scenes actions which will run in the background such as initiating a database. For example a button will be on the xml page, but it's on click action a listener will be found within the java file.

The android 4.2.2 is the current version of android that you are developing on this provides the current features found within that operating system and allows your application to access the application programming interface (API) of that platform. APIS can be accessed for a whole host of reasons such as Google maps integration and many others but I will be using the API access to enable NFC technology.



The bin folder will contain any images that are used for Icons throughout the application. More often than not there will be several sub folders contain various sizes of the same image, this is because if the application is opened on a different android phone other than the Galaxy s3 it will automatically use the best size images for different phones.

The res folder will house all the other resources used within the application such as Images that may be found on various XML pages. One of the main sub folders here is the Layout folder, inside this are the many XML files that will be used to determine the look and feel of the application.

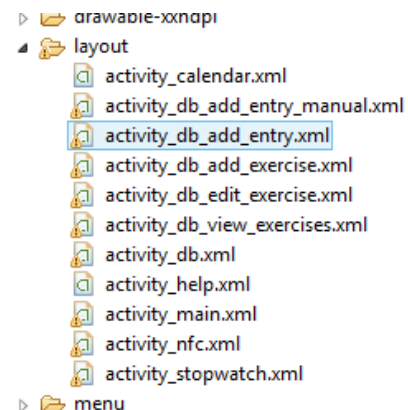
Finally there is an android manifest which every application must have. The manifest file holds important information such as the name of the java package for the application. This is needed to ensure that the application has a unique identifier. It will describe the components of the application, activities, receivers and capabilities the application has. This will also set the minimum API level of the application. This is used to determine how far backwards compatible the application will be to older generation phones.

Key areas utilized in development

In order to approach the build of the application properly key areas will need to be considered before any code is written.

First there will be xml pages which stand for EXtensible Markup Language. These are the pages that the user will see and navigate. They will feature the images and buttons of the application. These are typically called the layout pages. As these will dictate the look and feel of the application.

In the image shown to the right Fig[5] an XML has been created for each of the pages that will be used in the application, 11 in total.

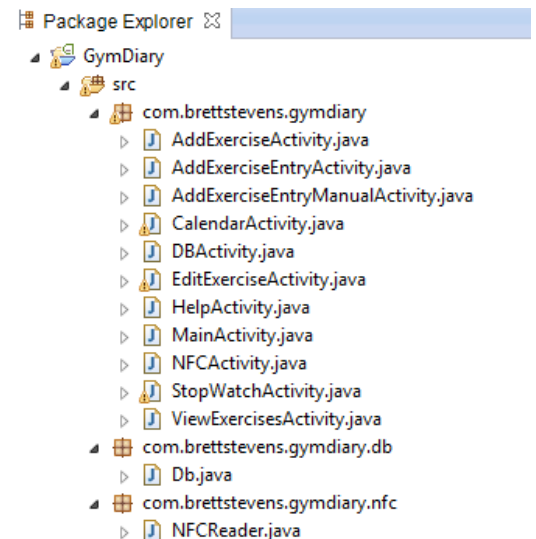


Next will be java class packages, these will house the Actions of each sub class java file; Fig [6]

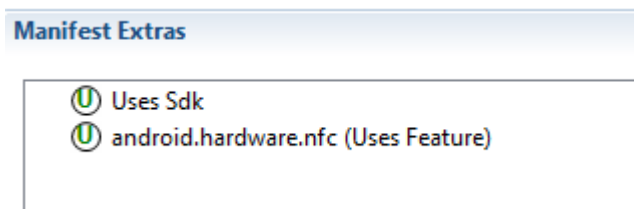
Com.brettstevens.gymdiary will hold the java files for each of the XML pages actions.

But there are also two separate Java packages which are Com.brettstevens.gymdiary.db which contains DB.java this is where the central database for the application is found.

Com.brettstevens.gymdiary.nfc contains NFCReader.java here is the code that utilizes the NFC technology and allows the phones NFC reading chip to be accessed from the Gym Diary Application and read NFC Tags.



The android manifest is another key area utilized in development as every XML page that is used within the application will need to be declared in the android manifest, and also NFC permissions will need to be enabled. This is to ensure that when the application is run on an NFC enabled device the NFC chip will allow permission to the app. The image below shows that the NFC is selected as extra to the standard SDK used. Fig[7]



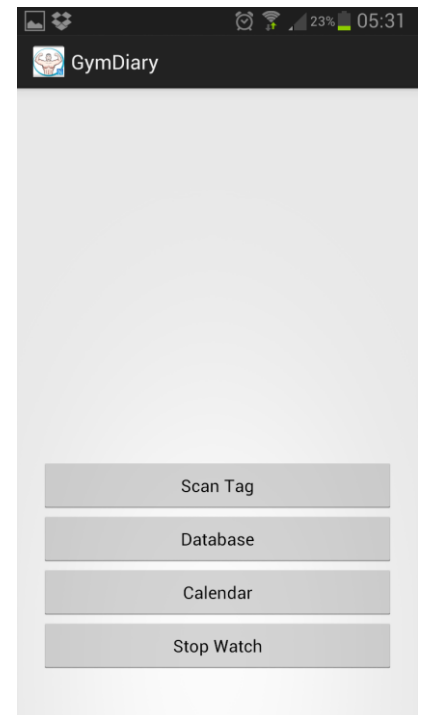
Approaching the Interface

Initial designs were creating all 11 pages into xml pages using the process flow diagram of the system as a guide.

This would allow the application a base in which to build from. As you can see in the image to the right the initial designs were very bland and not very appealing. And also lacking in other factors such as Help pages and manual entry pages that were added later on in the building phase of the application. Fig[8]

Each XML page in the application will need to handle different types of data. Some pages will be dealing with Scanning NFC tags, others will be accessing the SQLite database others will be dealing with drop down menus of text.

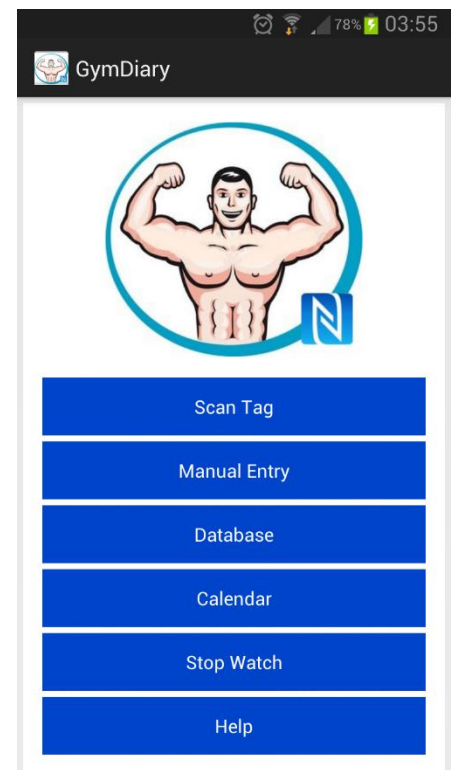
The activity_main.xml will need to contain 6 buttons and ImageView (to enable images to show within the XML file).



Based on user preferences I have opted to use large, screen wide buttons which will use a blue background with white text so it clearly stands out to users. The application will also have a white background to provide adequate contrast of colours on the screen so the app is easily navigable. This page also features an Image of the Gym diary's Logo. Fig [9]

This design is simple yet effective as it provides the correct functionality will still retaining a simple interface to access the features.

The image to the right is the Final design for the main page.xml of my application. This page will feature 6 buttons which will link to all subsequent pages. This xml page will be linked to a mainactivity.java file which will have the on click listener's so that each button found inside the main page will be clickable through to the corresponding pages.



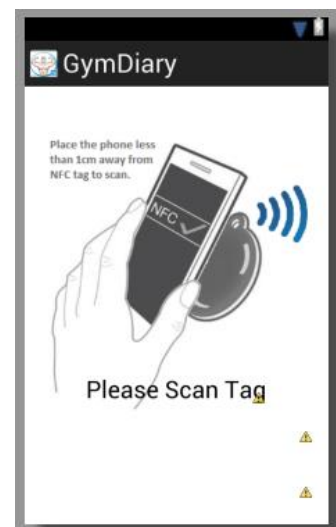
Defining objects that are used in Gym Diary xml's

- Button –Simple button
- TextView- Text Label
- ImageView- Inserting an image into the xml
- Spinner – Drop down list populated by SQLite database
- ListView- List populated by SQLite Database
- Chronometer – Implements a simple timer
- Calendar View – Creates a monthly view calendar
- Edit Text –Creates a editable Text field.

What Objects are in each XML page

The scan NFC tag XML page will need to include ImageView , TextView and a button. Image to right Fig [10]

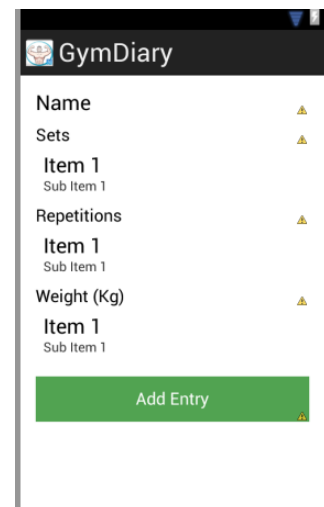
This is to show the User what to do with the image, and the TextView will change depending on what action is taken. First the text will display Please scan text but once an NFC tag is scanned the text will change to the Exercise contained on the tag. When the correct Tag is scanned it will be added to the database and then a button will appear to let the user decide if they wish to add that exercise to their workout. This will in turn link to the Addentry.XML.



This screen will be used for when automatically scanning and NFC tag
Image to right Fig [11].

The activity_db_add_entry_.xml will feature four TextViews for the labels (Name, sets, repetitions and weight) and three spinner objects (drop down menus) that will be populated with Integers so that the user can select the number of sets, reps and weight used and there will also be a button which will be used to add all of the selected entry's to the SQLite database.

This page is also linked to AddExerciseEntryActivity.java to provide the OnClick listener's for the button and population of the spinner objects.

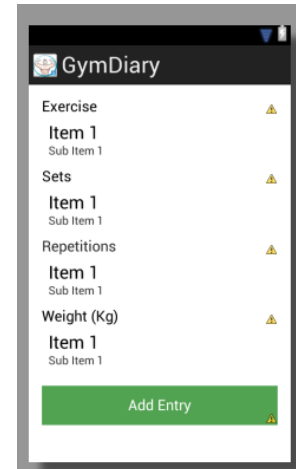


There will also be an identical page as the addentry.xml, but this page will be used for manually adding entries to the database, without the use of NFC scanned Tags. This is the activity_db_add_entry_manual.xml page.

Image to right Fig [12]

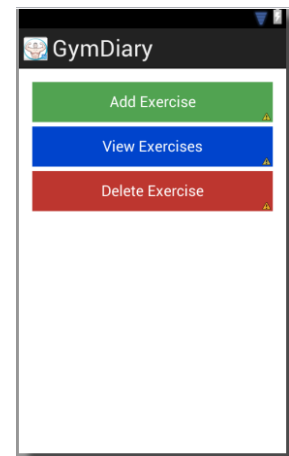
This page also features four TextViews for the labels (Name, sets, repetitions and weight) and three spinner objects (drop down menus) that will be populated with Integers so that the user can select the number of sets, reps and weight used and there will also be a button which will be used to add all of the selected entry's to the SQLite database.

This page is also linked to AddExerciseEntryActivity.java to provide the on click listener on the button and populate the spinners with integers.



This is the Database homepage, from here the user will be able to add/ delete are view exercises found within the SQLite database. This XML page features three buttons which link to 3 additional pages. This is also linked to the DBActivity.java file which will provide clickable functionality to the additional XML pages. Image to right Fig [13]

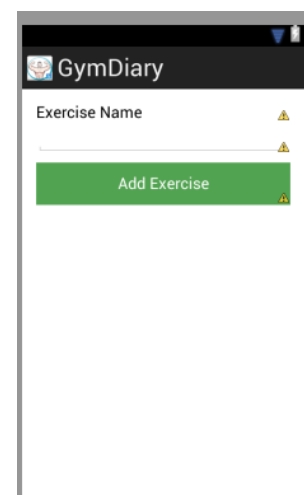
This page is also linked to DB.activity.java and DB.java. This is to enable total control of the database creation and dynamic editing.



This is the Add exercise activity page which is called the activity_db_add_exercise.xml page it contains a TextView label, TextField and a button. Image to right Fig [14]

The pages TextField will let the user type their own exercise in to the field and upon clicking the add exercise button will add to the SQLite database.

This page is also linked to AddExerciseActivity.java, which provides the OnClick listener for the button and sending the text entered into the text field to a record within the SQLite database.

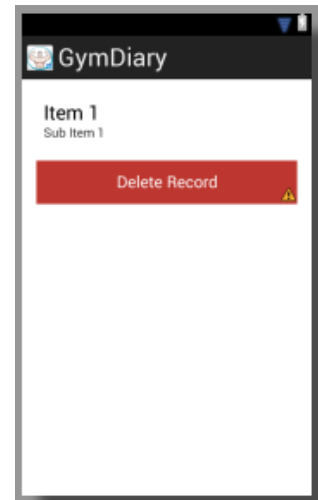


This page allows the user to delete entries from the SQLite database.

This is the activity_db_edit_exercise.xml page which contains a spinner and a button. Image to right Fig [15]

The spinner will create a drop down menu populated by all exercises contained within the SQLite database. From here users will be able to remove entire exercises from the Gym Diary.

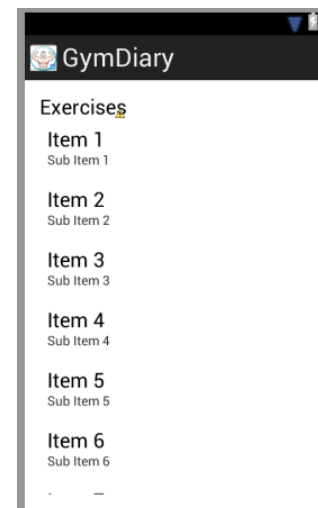
This XML page is also linked to EditExerciseActivity.java which has the OnClick Listeners for the button and populating the spinner with entries found within the database.



This is the activity_db_view_exercices.xml. This features A TextView and a Spinner. Image to right Fig [16]

This page will features all exercises that are currently found within the SQLite database and list them as a scrollable list.

This page is linked to ViewExercisesActivity.java which provides populating the spinner with the exercise entries from the SQLite Database.

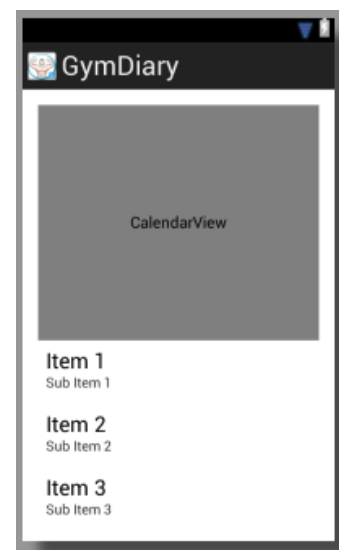


This is the activity_calendar.xml page. It features a CalendarView and a Spinner. Image to right Fig [17]

This CalendarView object will show the current month view of dates and allow the user to select a date and see all records associated with that date. By default todays date will be selected upon opening. The spinner below will be populated by the selected dates exercise entries.

This page will also allow adding and deleting of records in the database from CalendarView.

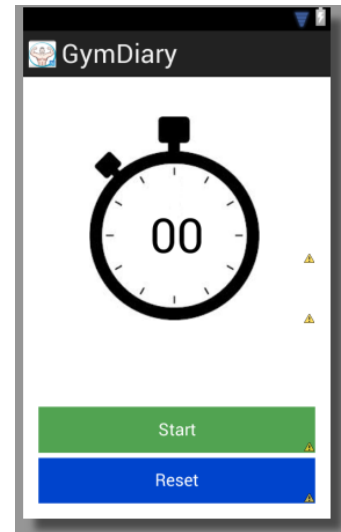
This XML is linked to CalendarActivity.java which will provide the current month is populated in the CalendarView and that the spinner below is populated by the selected date's records.



This is the activity_stopwatch.XML. This page Features two buttons an ImageView and Chronometer. Image to right Fig [18]

The buttons contained will be start/stop merged into one (it will change on click) and also change the colour of the button to match the action. A reset button. The chronometer will display Hours, minutes and seconds.

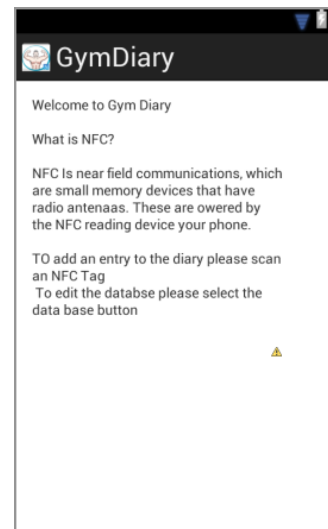
This is linked to StopwatchActivity.java which will provide the OnClick Listeners for the buttons and the enabling of the Chronometer.



This is the help page which features an ImageView. This is called activity_help.xml. Image to right Fig [19]

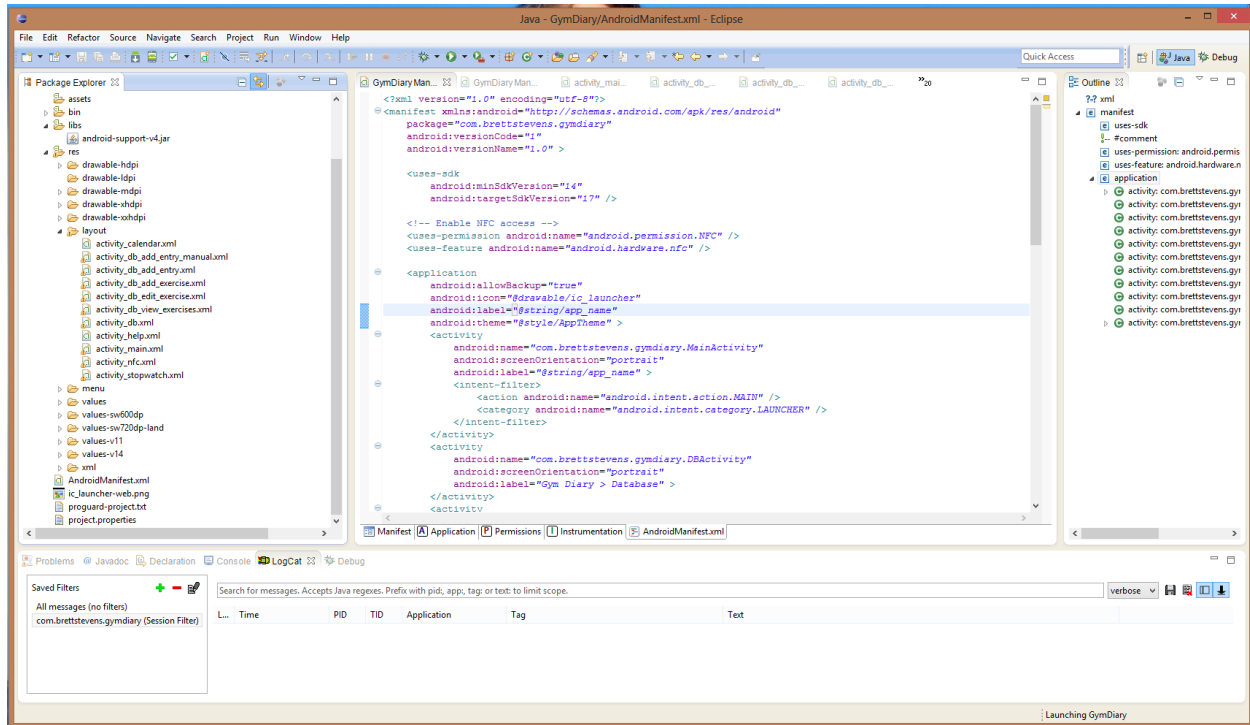
This page shows an info graphic on what NFC does and how it works in the application.

This XML page is linked to HelpActivity.java.



Implementation

Now was the task of transforming my designs into reality, first off running eclipse IDE with Android SDK plugin. This creates a dynamic environment in which to build my application. Below is a screen shot of the program running. Image below Fig [20] .



The Items to the left of the screenshot is where the Package explorer are located, here is where the Gym Diary's attributes and files will be stored. The center screen is where the code is implemented, this can be switched to different views to see the application running with a graphical layout (XML). The lower part of the screen is used for debugging which I will discuss later.

Before testing of the code can begin I created an android virtual device (AVD). This creates a virtual Android phone device. This is used to test if the application works on a simulated virtual phone. With the AVD you can set parameters for any off the current Android handsets available, you could even test the application on a tablet device. This is a very useful tool as you can run code instantly on the computer and iron out any errors that may arise. There was one caveat, my intended application utilizes NFC technology and this cannot be simulated in the virtual environment as the NFC tags that I am using are a physical entity. I would need to test my work in progress on a real device. The AVD virtual phones that will test the application will be used to bench mark the application on a non NFC enabled device. This will allow me to see how the application will respond on a non NFC enabled device.

Test device

The test device for my project is a Samsung Galaxy S3. The phone is set to allow USB debugging, this enables the ability to install applications on the device without a notification and to read log data. This is very useful as the application can be running in real time and you can see a live log feed of the application.

So any errors that take place you can access where and when they are occurring. The test device is much faster than the AVD test phones, despite the AVDs running on a much more powerful machine.

LogCat

The logCat is the inbuilt debugging tool found inside Eclipse IDE. This allows the user to see every process output of the application that is currently running. In both the AVD and on a real device.

Below is a code snippet of the code used to call up the LogCat during operation of the test application.

```
//If 3 numbers are parsed it returns in simple date format
protected String intToDate(int year, int month, int day) {

    Calendar date = Calendar.getInstance();

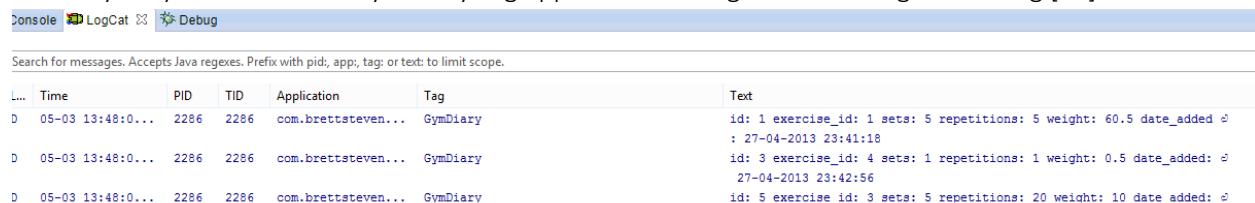
    date.set(Calendar.YEAR, year);
    date.set(Calendar.MONTH, month);
    date.set(Calendar.DAY_OF_MONTH, day);

    String selectedDate = new
SimpleDateFormat("dd/MM/yyyy").format(date.getTime());

    //debug messages
    Log.d("GymDiary", String.valueOf(year));
    Log.d("GymDiary", String.valueOf(month));
    Log.d("GymDiary", String.valueOf(day));
    Log.d("GymDiary", selectedDate);

    return selectedDate;
}
```

The Log.d command calls up variables from the code and outputs them to the Logcat view as shown below. This allows you to determine where an application is crashing if the log commands are used correctly. As you can see the Gym diary Tag appears in the LogCat view. Image below Fig [21]



...	Time	PID	TID	Application	Tag	Text
D	05-03 13:48:0...	2286	2286	com.brettstev...	GymDiary	id: 1 exercise_id: 1 sets: 5 repetitions: 5 weight: 60.5 date_added: 27-04-2013 23:41:18
D	05-03 13:48:0...	2286	2286	com.brettstev...	GymDiary	id: 3 exercise_id: 4 sets: 1 repetitions: 1 weight: 0.5 date_added: 27-04-2013 23:42:56
D	05-03 13:48:0...	2286	2286	com.brettstev...	GymDiary	id: 5 exercise id: 3 sets: 5 repetitions: 20 weight: 10 date added:

NFC

Before NFC technology can be utilized within an application the required packages need to be imported to enable the use of the NFC. The code snippet below shows the packages required.

```
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
```

These are to ensure that the device is first checked to see if it NFC capable.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_nfc);

    //Set textView to our label
    textView = (TextView) findViewById(R.id.checkNfcAdapterTextMain);

    //Get and initialise NFC adapter from phone
    NfcAdapter = NfcAdapter.getDefaultAdapter(this);

    //Check if we have got an adapter and set text to indicate
    if(NfcAdapter == null) {
        textView.setText("Unable to enable NFC adapter");
    }
}
```

The code initializes the NFC adapter and if no NFC adapter is found display the message Unable to enable NFC adapter. This is so users who use the Gym Diary application will know if there device has NFC or not. My NfcReader.java file contains the code that enable the application to read data off the NFC tag. The code extract below shows pulling the data from the NFC Tag. I used NFC Programming [2013] as a guide for integrating NFC.

```
//Initialise string
String tagText = "";

//Get array of data to parse
Parcelable[] data =
intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);

//Check if we have data payload, if not it is an empty but formatted
tag
if (data != null) {
    try {
        //Loop through data and get stored records
        for (int i = 0; i < data.length; i++) {
            NdefRecord [] NDEFRecords =
            ((NdefMessage) data[i]).getRecords();

            //Loop through record and get
            for (int j = 0; j < NDEFRecords.length; j++) {
                if (NDEFRecords[j].getTnf() ==
                NdefRecord.TNF_WELL_KNOWN && Arrays.equals(NDEFRecords[j].getType(),
                NdefRecord.RTD_TEXT)) {

                    byte[] tagPayload = NDEFRecords[j].getPayload();
                }
            }
        }
    }
}
```

This reads in the NFC tag that gets scanned. This is done by first enabling the NFC adapter which will read in the NFC Tag. This will then loop through the data stored on the NFC Tag.

The bitwise operators 0777 gets the length of the country code identifier from the control byte. And the bitwise operator 0200 gets the bit identifying whether it is stored as UTF-8 or UTF-16 from the control byte.

This information is used to determine what data needs to be extracted from the chip. Once the language is known it can be stripped off the tag as this is not needed as it will start each entry with EN (for English).

The code snippet below shows the converting the binary data from the NFC Tag into a string form. The payload is to start after the language and extract all information contained within and convert to string.

```
tagText = new String(tagPayload, (tagPayload[0] & 0077) + 1,
tagPayload.length - (tagPayload[0] & 0077) - 1, ((tagPayload[0] & 0200) == 0)
? "UTF-8" : "UTF-16");
    }
}
```

Now that the NFC tag information can be stored as text, The NFCActivity.java class will be used to determine what to do with the converted text. When the text is read in it is stored like this "1;Bench press" The separator ";" will be used to see if the Tag being read in is designed to be added to the database.

```
if(result.get("payload") == "-1") {
    textView.setText("Empty or unrecognisable tag scanned -- Please
try again");
    addActivityFromTagButton.setVisibility(View.INVISIBLE);
} else {
    String[] values = result.get("payload").split(";");
    if(values.length == 2) {
        textView.setText(values[1]);
        List<String> checkForExercise =
gymDiaryDb.checkForTagId(Integer.valueOf(values[0]));
        Log.d("GymDiary-Size",
String.valueOf(checkForExercise.size()));

        if(checkForExercise.size() == 0) {
            toast("New exercise scanned, adding " + values[1] + " to
database");
        }
    }
}
```

If the tag read in contained the wrong information format, the option to add to database will not be applied and resulting error messages will appear. Please see error checking section of the report. Random Thoughts: Read/Write NFC tag in Android. [2013]. was a strong influence in deciding how to implement this within my application.

SQLite database

The SQLite database will allow records to be stored within the application its self. This will contain NFC scanned Tag data, as well additional data added to the entry such as repetitions , sets and weight. Each time the scan data is read in to the database it will also have a time stamp attached. This is to ensure later when putting results to calendar view they appear in the correct place. The code snippet below shows the creation of the tables found in the SQLite database.

```
@Override
public void onCreate(SQLiteDatabase gymDiaryDatabase) {
    String query;
    query = "CREATE TABLE \"exercises\" (\"id\" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL UNIQUE , \"name\" TEXT NOT NULL UNIQUE , \"tag_id\"
INTEGER UNIQUE )";
    gymDiaryDatabase.execSQL(query);
    Log.d(GymDiary, "Exercises table added");
    query = "CREATE TABLE \"entries\" (\"id\" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL UNIQUE , \"exercise_id\" INTEGER NOT NULL , \"sets\"
INTEGER NOT NULL , \"repetitions\" INTEGER NOT NULL , \"weight\" DOUBLE NOT
NULL , \"date_added\" DATETIME NOT NULL )";
    gymDiaryDatabase.execSQL(query);

    ContentValues values = new ContentValues();
```

The database is populated by two entries by default as shown in the code snippet below. This is to make sure that the database will not be deleted by the user by mistake. Please see error checking section of the report for more information.

```
ContentValues values = new ContentValues();

values.put("name", "Bench press");
values.put("tag_id", "1");
gymDiaryDatabase.insert("exercises", null, values);

values = new ContentValues();

values.put("name", "Deadlift");
values.put("tag_id", "2");
gymDiaryDatabase.insert("exercises", null, values);
```

Adding the data into the database can be done from number of screens from the application, this can be done through scanning a tag at the scan tag page which will add an entry to the Database or through the dedicated database management. Below is the code snippet showing how additional data such as repetitions sets and weights are added to records already within the database.

```
public Boolean addEntry(String exerciseId, String sets, String reps,
String weight, String dateAdded) {
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    Boolean success = false;

    values.put("exercise_id", exerciseId);
    values.put("sets", sets);
    values.put("repetitions", reps);
    values.put("weight", weight);
    values.put("date_added", dateAdded);

    if(database.insert("entries", null, values) != -1) {
        success = true;
    }
    database.close();
    Log.d("GymDiary", "Added entry");

    return success;
}
```

Calendar

The calendar view plays an integral role with in the Gym diary application as it's the most use section of the application due to being able to see scanned record's on any specific day in list view but also due to the ability to add and delete records form this view as well. The design guide for this is found at, Stack Overflow. Calendar [2013] please see references.

The code snippet below shows creating the calendar View object. And populating the list view with records added for that day.

```
//create calendar
CalendarView calendar = (CalendarView)
findViewById(R.id.calendarView);
Calendar currentDate = Calendar.getInstance();

//popluate list with todays exercises
populateList(currentDate.get(currentDate.DAY_OF_MONTH),
currentDate.get(currentDate.MONTH), currentDate.get(currentDate.YEAR),
false);
```

This code snippet below creates an on click listener inside calendar view so if there is a long click detected it will use the current selected date and give the option to add a new entry to the database. It also covers if the date changes to repopulate the list with current selected date. This is so if the user selects a different date only records from the selected date are shown.

```

//creates listener on long click(in calendar view)
calendar.setOnLongClickListener(new OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        //On long click display dialog included selected date
        addEntryDialog.setMessage("Do you want to add an
entry for " + intToDate(currentlySelectedYear, currentlySelectedMonth,
currentlySelectedDay) + "?");
        addEntryDialog.show();
        return true;
    }
});

//whens date changes re-populate list with current selected date
calendar.setOnDateChangeListener(new OnDateChangeListener() {
    public void onSelectedDayChange(CalendarView view, int year, int
month, int dayOfMonth) {
        populateList(dayOfMonth, month, year, false);
    }
});

```

Spinners

Are used to create dropdown menus within java they can either be populated with preset records or can dynamically retrieve records stored from a SQLite Database. There was extensive work done by Android Hive [2013] that involved populating a spinner class with entries from the SQLite database. The code snippet below shows creation of the spinners used and the Values that are used to populate the spinners.

```

//Containers that hold the values of the spinners
ArrayAdapter<String> setsAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, sets);
ArrayAdapter<String> repsAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, repetitions);
ArrayAdapter<String> weightAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, weight);

//Setting the drop down view of the spinner

setsAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);

repsAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);

weightAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdow
n_item);

//Adding the values to the spinners
setsSpinner.setAdapter(setsAdapter);
repsSpinner.setAdapter(repsAdapter);
weightSpinner.setAdapter(weightAdapter);

```

The code snippet below shows the code for deleting a record entry from the calendar view. It will show a message that gives the user the option to not delete the currently selected record.

```
//Sets dialog if row is deleted returns message "exercise deleted" if
not "failed to delete exercise"
    final AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setTitle("Delete Entry");
    dialog.setMessage("Are you sure you want to delete the selected
entry?");
    dialog.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            if(gymDiaryDb.removeEntry(idToDelete) > 0) {
                toast("Exercise deleted");
                populateList(currentlySelectedDay,
currentlySelectedMonth, currentlySelectedYear, true);
            } else {
                toast("Failed to delete exercise");
            }
        }
    });
```

If the record is unable to be deleted a Toast pop up message will be used to show the user that their selected action was not carried out successfully.

Stop watch

The stop watch features a chronometer which is used to implement a timer in the stopwatch.xml. Android chronometer- Stack Overflow. [2013] featured guidance for the chronometer. On this page I was looking to merge both the start and stop buttons, so that once the user would see the start button it would be green, then once the timer was started the button would change to red and change the text to stop. The code snippet below shows how this was implemented.

```
//Sets listener for start / stop button
startStopButton.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            String button =
startStopButton.getText().toString();
            if(button.equals("Start")) {
                stopwatch.setBase(SystemClock.elapsedRealtime());
                stopwatch.start();
//On button press changes colour of button to red

                startStopButton.setBackgroundColor(Color.parseColor("#bd362f"));
                startStopButton.setText("Stop");
            } else {
                stopwatch.stop();
//On button press changes colour of button to green

                startStopButton.setBackgroundColor(Color.parseColor("#51a351"));
                startStopButton.setText("Start");
            }
        }
    });
```


Results and evaluation

The Gym Diary application is now finished. Here are the initial Requirements from the interim report that I set out to achieve from this project, paired with the functionality that the final application offers. Table below is Fig [22].

Features	My Android Gym Application using NFC Initial requirements	Finished Gym Diary Application
NFC reading capability	√	√
Dynamic SQLite database	√	√
Ability To create workouts	√	√
Workout guides	√	√
calendar	√	√
Ability to read data in from the NFC tags and store into the database	√	√
Interval timer	√	√
Error Checking systems	-	√

As you can see I have achieved all of the key tasks that I set out to do, and the project has evolved into a more developed system as I now have refined error checking systems in place to ensure the integrity of the application. This was not something that I considered in the early design stages but something that I felt needed to be added to the refining of the application.

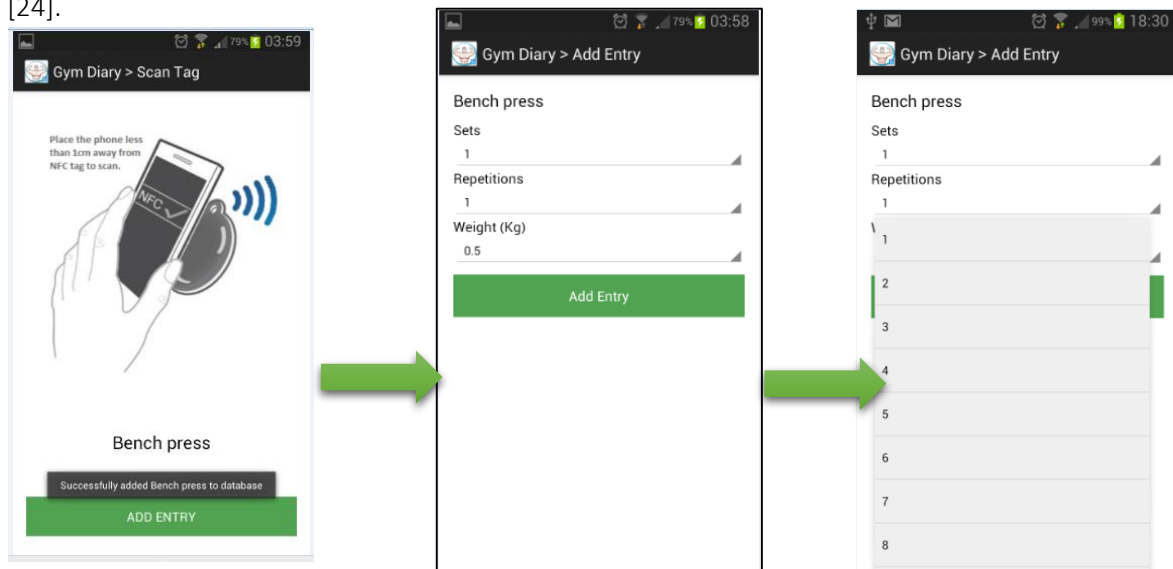
Key Features of Gym Diary

The Gym Diary NFC enabled applications main features are the ability to read in exercises stored on to an NFC Tag dynamically check to see if the record has already been stored within the database, if it has not the application will now add the newly scanned exercise to the SQLite database. The user will then be taken to a screen in which they can add extra information about the exercise they have scanned. Such as

the amounts of sets they have done, how many repetitions and the total weight used. Below are screenshots of this process being carried out. Image below Fig [23].

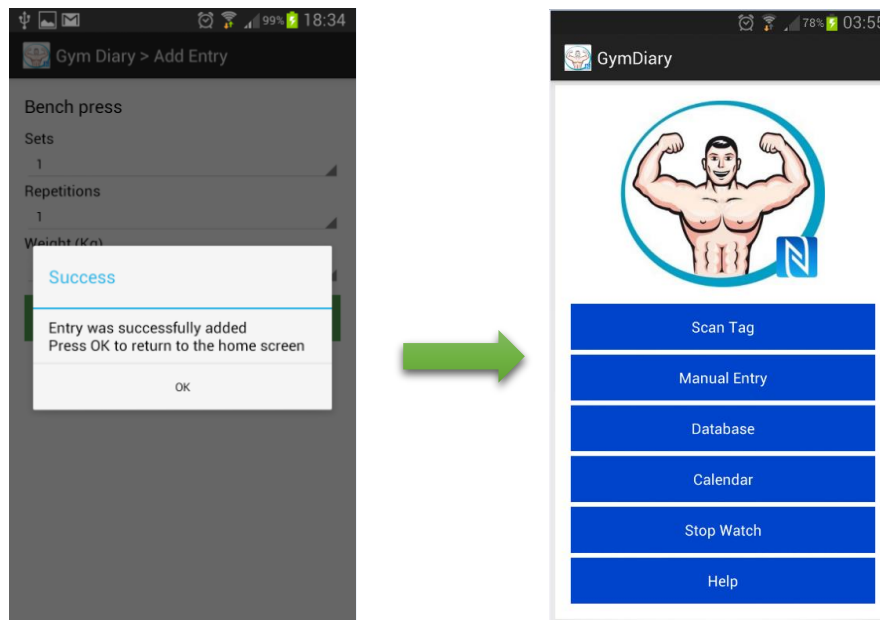


The first page shown is the main page of the application, the user will now select the scan tag button, and this takes the application to the scan tag page. Here is where the user is asked to scan the NFC Tag exercise they wish to add to the application. When the tag has been scanned the application determines that this exercise is not found within the database. So a toast pop up message appears letting the user know that the “Bench Press” is being added to the database. Because the NFC Tag is in the correct format a button now appears giving the user the option to add the entry into their gym workout. Image below Fig [24].



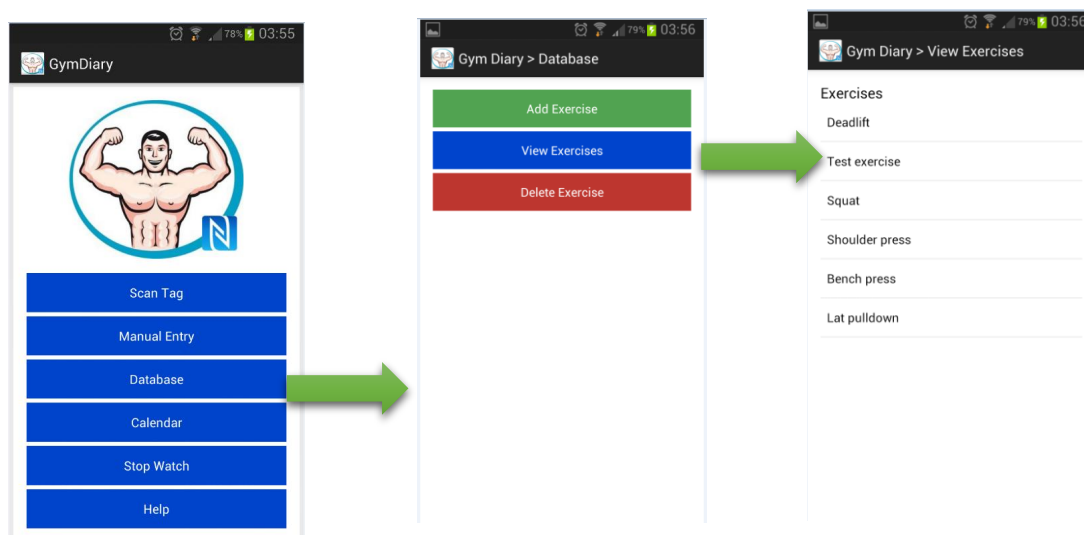
Now the user sees the message that the bench press exercise has successfully been added to the database. They can now select Add entry which provides spinners (Drop down menu) for the amount of

Reps, sets and weight used. Once the required information has been selected the user clicks the Add entry button. And they will see this message. Success entry was added. This will now put the user back at the home screen, so that they can scan more Tags if needed. Image below Fig [25]



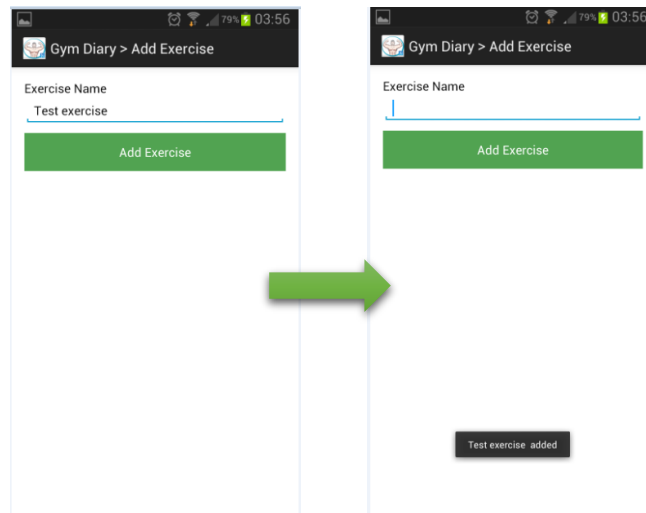
Please see Error Checking systems for how the system copes with scanning an empty Tag and also a Malformed Tag with the incorrect information stored.

Another key feature for Gym Diary is the SQLite database. This is accessed again from the main page by clicking on the database button. This will present the user with actions that they can engage with the database. Adding exercises, deleting exercises and viewing all the exercises currently stored within the application. Selecting View exercises will show the user all current exercises stored within the SQLite database. Image below Fig [26]



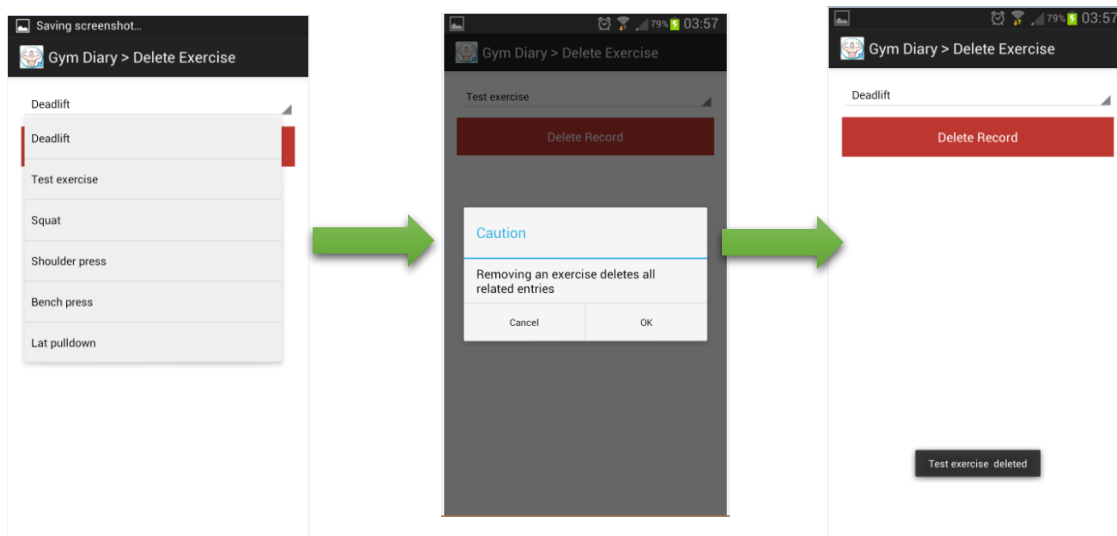
This is designed so if the Gym did not have an NFC Tag for a particular exercise the user can simply create their own in the database. Which would then be selectable to add entries within the database. Now if the user selects Add exercise they will be presented with a page where they can type in the name of the exercise they wish to add to the database. Shown below. Once the exercise is entered then the add exercise is selected, the user will be greeted with a pop up message that the exercise has been added.

Image below Fig [27]



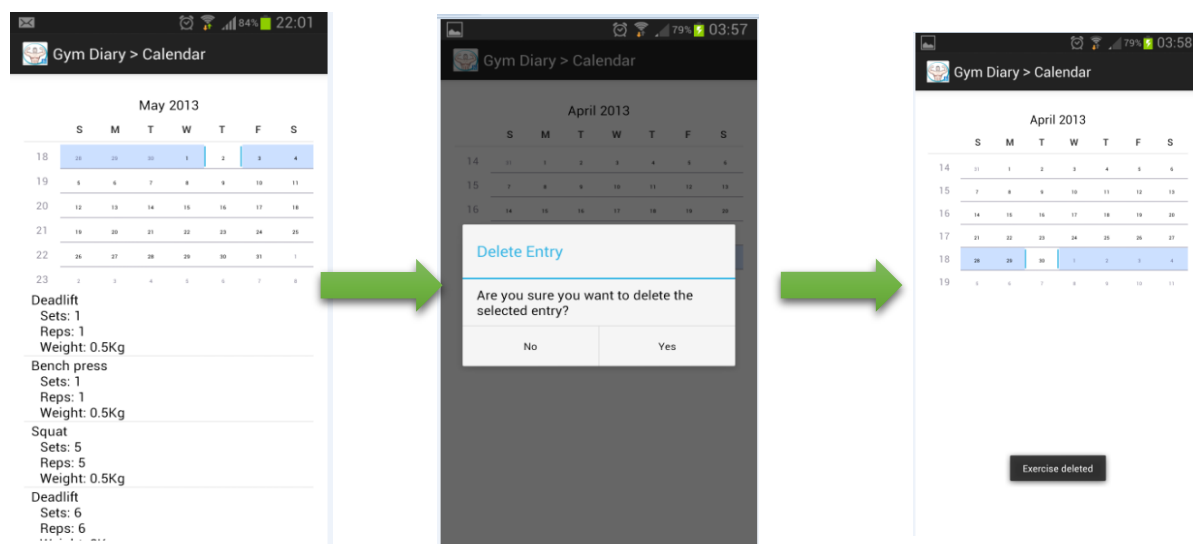
Now if the user wishes to delete an exercise they can select Delete exercise button from the Database page. This page will provide a drop down menu (spinner) which is populated by all the exercises currently stored in the SQLite database. The screenshots below show the process of deleting a record.

Image below Fig [28]

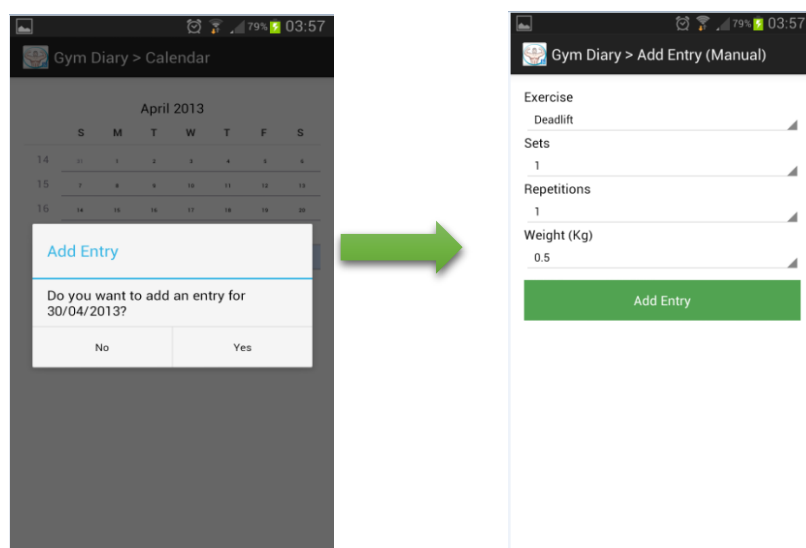


When the user has selected the record they wish to remove they simply click the delete record button, they will be shown a caution message. They can either carry on with deletion or select cancel and return to the previous screen. When deletion has been completed a message will pop up confirming exercise deletion.

The calendar is also a prominent feature of my application as it ties both the scanning of NFC tags and the database pages together nicely into one simple page. The role of the Calendar view is to show a scrollable Month calendar with a list view situated underneath. The nested list below the calendar shows all entries that have been added to the day that is selected. The screenshot below shows an example of a selected day. Image below Fig [29]

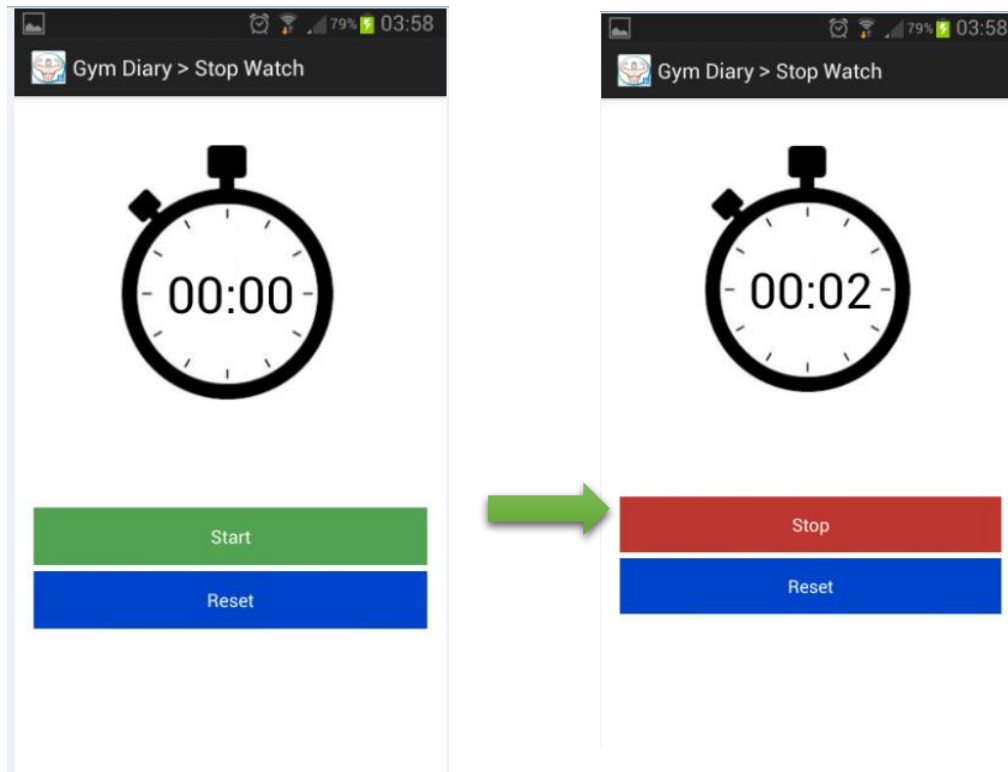


The calendar view is design to be able to edit the database from this view as well. If a user wishes to remove an entry for a selected day they simply long press on the exercise they would like to remove, and a message will pop up asking if they wish to remove that entry, selecting yes will make the entry disappear followed with a confirmation message that the entry has been removed. Image below Fig [30]



Also added into the calendar view is that ability to add new records for any past or future date as well as the present. This is done by first selecting the day you wish to add a record to, then long click on the month name in calendar view, this will take the user to manually add an entry page, where they can use the drop down menus to select a new record to be created.

The stop watch has also been included to allow gym user to time themselves between exercises. The screenshots below shows that the start /stop button will change colour depending on what action is going on. Start will be green and the stop will be red. Image below Fig [31]



Evaluation-

I feel that the application runs very smoothly, it has a slick clean interface that is easily navigable. The colours used are to aid its simplistic design and to guide the intended users. The colour scheme used are Green buttons which are to add entries/ exercisers , Blue buttons are to view pages/entries and Red buttons are for deleting/ removing exercises and entries. The Gym Diary has incorporated all of my intended primary functions.

From the ability to read in data stored on NFC tags and store this information into its own SQLite database. Check new scanned entries in the existing database to determine whether to add the new entry or not. A Fully fledged Calendar that is populated from the SQLite database and fully editable from the calendar view.

A Dynamic SQLite database that is fully editable to control Present, past and future records. Also included in the final application is a Stop watch. The features of the Gym Diary are designed to enhance a user's experience in the gym environment. As no longer do they need to manually search for the exercise and add it to their application they can simple swipe an NFC chip located in the gym to store the exercise for that session.

Limitations of NFC

NFC technology is becoming more and more frequent as the technology is becoming more robust. All of the latest Android handsets feature an integrated NFC chip, with rumors circling around that the iPhone 6 will also feature an NFC chip to match with the Passbook. But at the moment NFC is not widely accepted due to a large percentage of the market having non NFC enabled phones. I believe as soon as Apple gets on board with NFC integration it will really take off as both of the two largest Phone handset sellers will support the NFC integration.

NFC Tags offer a wide range of shapes and sizes the ones I used in the project ranged from the NTAG203 Tags all the way up to the 4k Desfire. Using the different size chips allowed me to test the various affects that distance has on the scanning of the Tag. The larger the NFC chip the further away the phone can be from the Tag.

Even though you could be further away from the tag and still read the data, in practice this only offered a few centimeters of distance from the tag. So the choice of NFC tag used wasn't a major factor when considering design implementation. NFC is just an extension of RFID technology, the maximum reading distance for NFC Tags is 4 inches.

Also during the project I broke one of the NFC tags by rolling my office chair over it. This is another issue to consider is the placement of the NFC tags as they can easily be damaged especially in the gym environment.

Problems during the project

The longest phase during the entire project was implementing an NFC reader as the documentation is very poor for new android developers. Many of the online "guides" were merely snippets of code that were useless to the new android developer. I found that many of the android development sites were heavily geared towards people with extensive prior knowledge of the android development sector. Fortunately after many hours of searching I was able to find NFC integration samples. Which were a great tool to teach me how I could implement my own NFC reader into the application.

Linked with this was getting the NFC chip read data to store in an SQLite database. As I needed the addition to get stored along with date and time of scanning the chip. So I could show this data in a relevant place from calendar view.

Time was also another constant issue because I felt that I wasn't following my original Gantt chart well enough to start with, so I felt like I was majorly behind on the project mid-way through the third year. After accessing the situation and re prioritizing my goals I was soon back on track for the scheduled project completion.

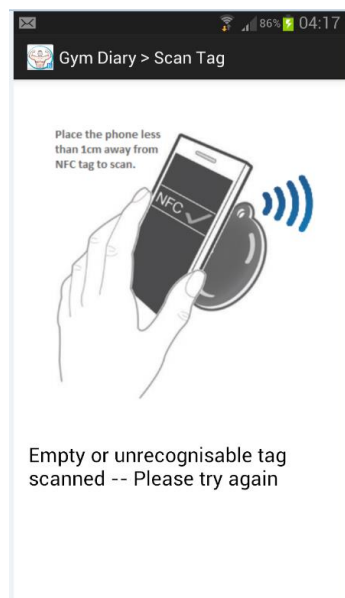
Error Checking

During development of the application I discovered that there were areas that could be exploited by errors and could compromise the integrity of the app. So to counter this I have put in several error checking systems. To help keep the application running as smooth as possible and prevent it from errors which it once had.

Reading blank/Malformed NFC Tags

First the application would come to an error if it scanned an empty tag, as it was originally set to find a unique ID and exercise listed like this 1;Bench Press. So the application would run into an error. To solve this if the payload read in from the NFC tag is -1, return the message "Empty or unrecognizable tag scanned – please try again". The code snippet is below. Example shown in the screenshot below. Image below Fig [32]

```
if(result.get("payload") == "-1") {
    textView.setText("Empty or unrecognisable tag scanned -- Please try again");
    addActivityFromTagButton.setVisibility(View.INVISIBLE);
} else {
    String[] values = result.get("payload").split(";");
```



This code snippet also shows that it prevents the add button becoming visible on this XML page, so an empty NFC Tag cannot be added to the database.

The next error check to solve was, what happens if an NFC tag has data on it, which is not applicable to the application. For example a malformed tag with "Ndjxd54". This was solved by creating an additional else statement to redirect incorrectly formatted NFC tags. The code snippet below from NFCActivity.java shows this. Image below Fig [33]

```
else {
    showButton = false;
    textView.setText(result.get("payload"));
    toast("Malformed Gym Diary tag detected.\nPayload: " +
result.get("payload"));
}
```

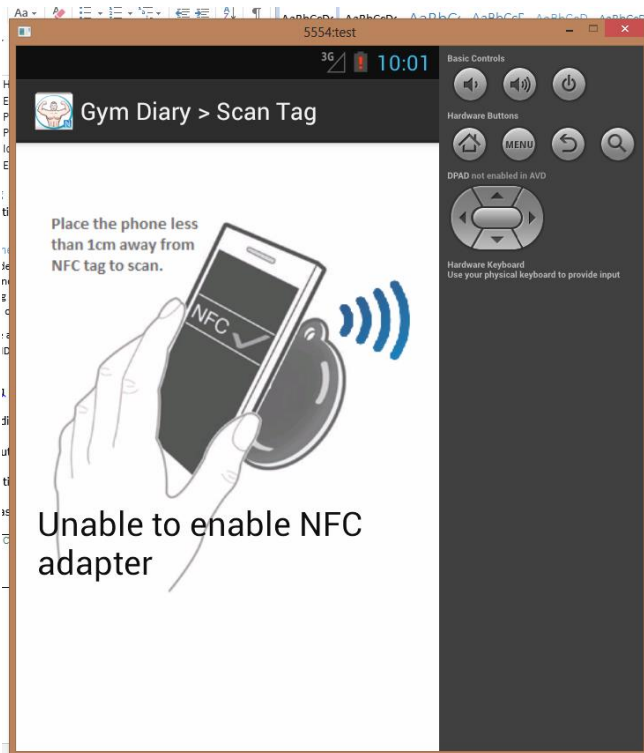


Here is a screenshot showing the error message that pops up this is a toast message. Where the text will be displayed for a few seconds then disappear. It will display the payload (content) of the tag but will not add it to the SQLite database as the add exercise button is set to be invisible due to a malformed Tag being read.

Will the Application work on a non NFC enabled phone?

This was solved by having the NFC activity.java class checking to see if the device has an NFC chip. Here is the code snippet below. Image below Fig [34]

```
//Check if we have got an adapter and set text to indicate
if(NFCAdapter == null) {
    textView.setText("Unable to enable NFC adapter");
}
```

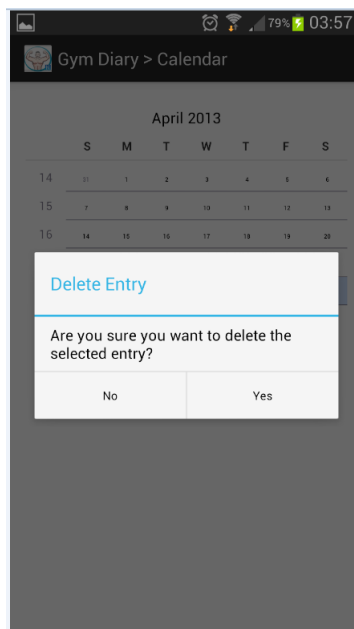


As you can see from the First Image on the left, this is when the application is run through the (AVD) Android Virtual Device manager. The computer has no NFC device so the error message Unable to enable NFC adapter pops up. But when ran on the Samsung galaxy s3 shown in the image on the right the user will be presented with please scan tag.

Deleting Records daily records from the SQLite Database

If the user wishes to delete a record a warning message will pop up on screen warning the user of their actions. This allows users to cancel the action if it was clicked by mistake. The code snippet below shows this. Below is a code snippet where a user removes a gym entry for a particular day in calendar view. Image below Fig [35].

```
//Sets dialog if row is deleted returns message "exercise deleted" if not
"failed to delete exercise"
    final AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setTitle("Delete Entry");
    dialog.setMessage("Are you sure you want to delete the selected
entry?");
    dialog.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            if(gymDiaryDb.removeEntry(idToDelete) > 0) {
                toast("Exercise deleted");
                populateList(currentlySelectedDay,
currentlySelectedMonth, currentlySelectedYear, true);
            } else {
                toast("Failed to delete exercise");
            }
        }
    })
}
```



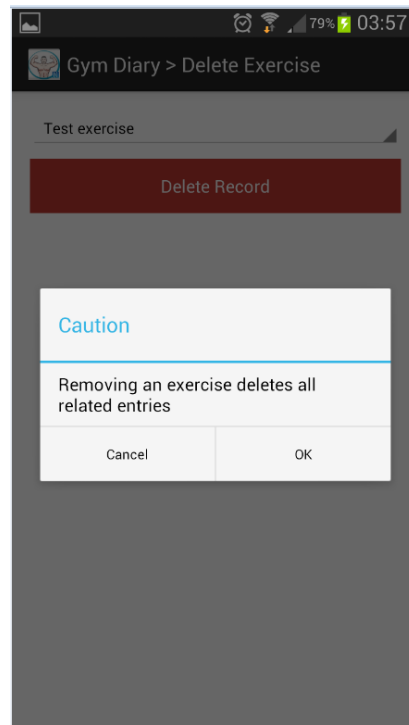
The screenshot above shows the error message as it appears when deleting a record from within calendar view. It also checks to see if there is a record, if not return "Failed to delete exercise".

Deleting entire exercise's records from database

A user can choose to delete an entire record from the database from the activity_database.xml, when they click on delete record. They are presented with a caution message of "Removing an exercise deletes all related entries". The code snippet is shown below. Image below Fig [36].

```
//Set dialog to show caution message when deleting exercise entry else if not
"failed to delete"
    final AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setTitle("Caution");
    dialog.setMessage("Removing an exercise deletes all related
entries");
    dialog.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            Spinner exerciseName = (Spinner)
findViewById(R.id.editExerciseSpinner);
            String exerciseNameText =
exerciseName.getSelectedItem().toString();
            if(exerciseName.getAdapter().getCount() > 2) {

                if(exerciseDb.removeExercise(Integer.parseInt(listOfAllExercises.get(ex
erciseNameText))) > 0) {
                    toast(exerciseNameText + "
deleted");
                    populateSpinner();
                }
            }
        }
    });
```



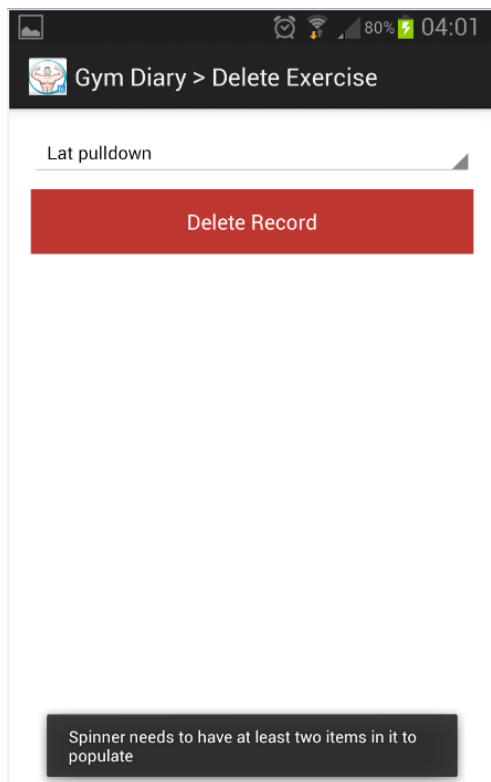
Also included within this error check is that the user can never completely delete the entire database by mistake. The fail safe mechanism is that there must always be at least a minimum of 2 entries in the database. If the user tries to delete the last two exercises they are presented with the error message shown below in the screenshot on the right hand side. Image below Fig [37]

```

    } else {
        toast("Failed to delete " + exerciseNameText);
    }

//Error message if user attempts to delete all the records(there must be two
minimum
    } else {
        toast("Spinner needs to have at least two items in it to
populate");
    }
}
}
});

```



The error message that pops up is a toast message and will disappear after a few seconds.

Testing Gym Diary Objects Function as intended

This testing stage will involve checking that every object used on all the XML pages is working correctly, this will also check all of the java class files that are associated with each XML page as they provide the actions of the objects.

Home screen –Activity_main.xml

Objects	Action	Success/Fail
Button 1	Got to Activity_nfc.xml	Fully working
Button 2	Got to Activity_db_add_entry_manual.xml	Fully working
Button 3	Got to Activity_db.xml	Fully working
Button 4	Got to Activity_calendar.xml	Fully working
Button 5	Got to Activity_stopwatch.xml	Fully working
Button 6	Got to Activity_help.xml	Fully working
ImageView	Load Gym Diary Logo	Fully working

Scan tag- Activity_nfc.xml

Objects	Action	Success/Fail
ImageView	Load scan tag image	Fully working
TextView	Display scan tag text	Fully working
Button 1	Open Activity_db_add_entry.xml	Fully working
Text view 2	Change displayed text when reading blank tag	Fully working
Text view 3	Change displayed text when reading in exercise	Fully working
Text view 4	Change displayed text to read malformed text and not add to database	Fully working

Add entry to Diary- Activity_db_add_entry.xml

Objects	Action	Success/Fail
Button 1	Add entry to database	Fully working
Spinner 1	Populate with sets 1- 9	Fully working
Spinner 2	Populate with reps 1-25	Fully working
Spinner 3	Populate with weight 0.5-300	Fully working
TextView	Display exercise name	Fully working
TextView	Display sets	Fully working
TextView	Display reps	Fully working
TextView	Display Weight(kg)	Fully working

Manual add to diary- Activity_db_add_entry_manual.xml

Objects	Action	Success/Fail
Button 1	Add entry to database	Fully working
Spinner 1	Populate with sets 1- 9	Fully working
Spinner 2	Populate with reps 1-25	Fully working

Spinner 3	Populate with weight 0.5-300	Fully working
TextView	Display exercise name	Fully working
TextView	Display sets	Fully working
TextView	Display reps	Fully working
TextView	Display Weight(kg)	Fully working

Go to database page- Activity_db.xml

Objects	Action	Success/Fail
Button 1	Got to Activity_db_add_exercise.xml	Fully working
Button 2	Got to Activity_db_edit_exercise.xml	Fully working
Button 3	Got to Activity_db_view_exercise.xml	Fully working

Go to calendar view - Activity_calendar.xml

Objects	Action	Success/Fail
CalendarView	Show Monthly calendar	Fully working
ListView	Populate selected dates records	Fully working

Selecting stop watch - Activity_stopwatch.xml

Objects	Action	Success/Fail
Button 1	Start timer	Fully working
Button 2	Stop timer	Fully working
Button 3	Reset Timer	Fully working
Chronometer	Display simple Timer	Fully working

Selecting help page- Activity_help.xml

Objects	Action	Success/Fail
ImageView	Display help page Image	Fully working

Activity_db_add_exercise.xml

Objects	Action	Success/Fail
Button 1	Add exercise to SQLite Database	Fully working
TextView	Label exercise name	Fully working
EditText	Editable field of text	Fully working

Activity_db_edit_exercise.xml

Objects	Action	Success/Fail
Spinner	Show list of exercises from SQLite Database	Fully working
Button 1	Delete exercise selected	Fully working

Activity_db_view_exercise.xml

Objects	Action	Success/Fail
ListView	Show list of exercises from SQLite Database	Fully working
TextView	Label exercise	Fully working

Testing Continued

The second phase of testing will be to carry out tasks the application was intended to do and record whether completion was achievable.

Adding a new Exercise to the database with an NFC TAG

Task	Success/Fail
Open Application	Fully working
Select scan tag	Fully working
Swipe NFC Tag	Fully working
Tag added to Database	Fully working
Return to add exercise view	Fully working

Adding a new Exercise to the database with an empty NFC tag

Task	Success/Fail
Open Application	Fully working
Select scan tag	Fully working
Swipe empty Tag	Fully working
Message display Empty Tag	Fully working
Please scan a new tag	Fully working

Adding a new Exercise to the database with Malformed NFC chip

Task	Success/Fail
Open Application	Fully working
Select scan tag	Fully working
Swipe malformed NFC Tag	Fully working
Output payload	Fully working
Display Malformed Tag message	Fully working

Adding a new Exercise to the database with manual entry

Task	Success/Fail
Open Application	Fully working
Select Manual entry	Fully working
Enter in records	Fully working
Message display added to DB	Fully working
Added confirmation message	Fully working

Add exercise to the database manually

Task	Success/Fail
Open Application	Fully working
Select database	Fully working
Select add exercise button	Fully working
Type in new exercise	Fully working
Click add exercise	Fully working
Message pop up saying new exercise adding to database	Fully working
Confirmation message pops up saying exercise added to Database	Fully working

View all exercises within the application

Task	Success/Fail
Open Application	Fully working
Select database	Fully working
Select view exercise button	Fully working
See list view of exercises	Fully working

Delete an exercise from the database

Task	Success/Fail
Open Application	Fully working
Select database	Fully working
Select delete exercise button	Fully working
Select exercise from drop down spinner	Fully working
Caution message to delete exercise	Fully working
Confirm Delete Message	Fully working

Accessing Calendar view to see records stored

Task	Success/Fail
Open Application	Fully working
Select calendar	Fully working
View list of exercises for that day	Fully working

Delete an entry from today's date from the Calendar view

Task	Success/Fail
Open Application	Fully working
Select calendar	Fully working
View list of exercises for that day	Fully working
Select record from today's date	Fully working
Long click on the record	Fully working
Display message do you wish to remove entry	Fully working
Select yes	Fully working
Confirmation message entry deleted	Fully working

Add an entry to a month before today's date

Task	Success/Fail
Open Application	Fully working
Select calendar	Fully working
View list of exercises for that day	Fully working
Select record one month behind today's date	Fully working
Long click on month name in calendar view	Fully working
Display message do you wish to add entry	Fully working
Select yes	Fully working
Confirmation message entry has been added	Fully working

Start the timer and stop after 1 minute

Task	Success/Fail
Open Application	Fully working
Select stopwatch button	Fully working
Select start button	Fully working
Wait 1 minute	Fully working
Timer is counting down	Fully working
Select stop	Fully working

Open the help page

Task	Success/Fail
Open Application	Fully working
Select help page	Fully working
Help page image shown	Fully working

Testing for NFC Tag placement

Testing to see if the NFC tag can still be scanned under certain conditions.

Scenario	Success/Fail
NFC tag under 1cm of wood	Fully working
NFC tag on a metal frame	Fully working
NFC tag under a metal frame	FAIL
Scan NFC tag through phone case (thin)	Fully working
Scan NFC tag through Thick case	Partially working

This testing phase was to assess possible placement of the NFC tags if there were to be put inside a gym. My best recommendation based on the tests is to have the stickers near the equipment and not placed directly on the machines unless a safe location is found as the Tags could easily be damaged by a falling weight.

Critical appraisal of results

Referring back to my initial plan I had listed a set of goals that I aim to achieve by the time the final report is due.

- Creation Of application.
- Database management.
- Adding new exercises.
- Creating features.
- Error checking.
- NFC integration.
- NFC tag programming.
- Production of working Prototype.
- Showcasing final product.

I am pleased to say that I achieved all of the goals that I had set out to do. The application is now a very robust Gym application utilizing an NFC chip reader. It can distinguish between different types of NFC tag. It has a set format for reading in correct tags and if empty/malformed tags are scanned it will return the correct error message to the user. This is handily linked to a dynamic SQLite database, which stores all of the NFC tags scanned information. The application even allows a user who does not have an NFC enabled phone to use the application as they can manually add records to the database which will then populate the drop down menus to allow quick easy record keeping on the go.

There is advanced error checking in place, this was added into the functionality of the project as time went on. This addition in turn made the Gym Diary application more secure and reduced the chances of user error.

Looking at the test results of the application I can see it functions well on the tasks it is intended for such as and will discuss strengths and weaknesses. ;

-Adding records to the database from an NFC Tag

This functions well, its strength is the speed at which it reads in the exercise and adds it to the database, and there is also code in place so that if the Tag is scanned twice by mistake the Gym Diary will not add a duplicate exercise in.

A possible weakness for this is if a NFC tag was in the correct format "1;Bench press " but actually said "23;lasdkihfbap" The application would add this into the database as its current filter options are to look for a ";" and check to see if there is an ID first integer if true add to the database. This would need to be better refined before a commercial release of the app.

-Editing / deleting/ viewing exercise records

The system performs these tasks with ease and even has several ways of doing this, its strength is that they can be edited from the database tab or from inside the calendar view. The calendar view offers a more streamlined process as the users can see all records associated for a particular date and edit the list on the fly by removing entries or adding new ones.

- Stop/start timer

Is a simple chronometer that enables the time to be started and stopped from the same button. Its weakness is that given more time could have been made in to a more complex aspect of the program. Extra features such as interval timing for laps could have been added, also the functionality to add times to say cardio exercises would be a nice addition.

-calendar view with list populated for the SQLite Database

The calendar view page functions well it has a list view that is populated by records stored in the database. This is very quick to navigate. The one crux if this page is the adding a new record to the database. For example deleting an entry is no problem simply select the record you wish to remove and press and long click to bring up the do you wish to delete message. To add a new record from the calendar view the user needs to select the date then click and hold on the date of the month. I think that with more time I would have set the on click action to add new entries to be on the selected day. Rather than making the user having to click elsewhere in calendar view just to add a new record.

-Manual entry to the database/Dynamic SQLite Database

This process is as simple as it gets, as shown in the testing, the user simple selects manual entry and types in the exercise that they wish to store. The strength here is simplicity easy to see what you're doing. A weakness could also be limited functionality as the ability to add a generic text field would be a great attribute as this could allow a user to enter any additional information with the exercise they wish to record.

Potential testing in a Gym environment

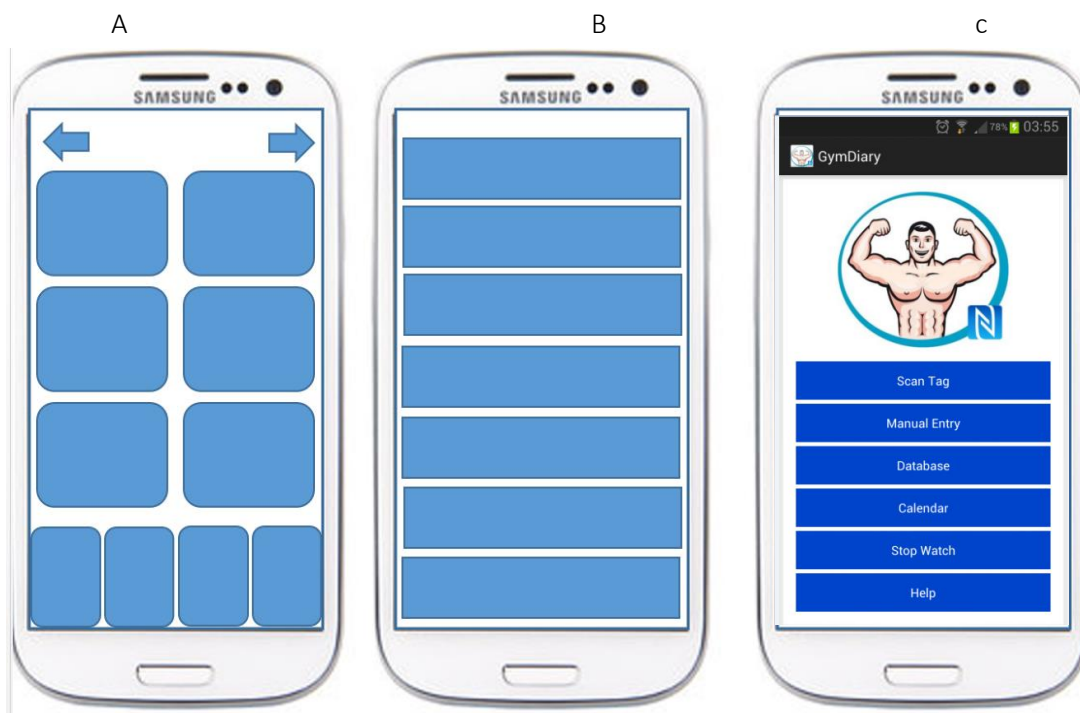
I believe more rigorous testing would have provided better results to work from. I would have liked to roll out the application prototype in to a real life gym. I.e. placing NFC tags strategically around the gym pre labeled with the correct exercises for the equipment they are placed near and release the application on the Google play store for free. This would allow anyone with an android phone to download the application.

From this initial testing ground I would have been able to measure the interest in having a simple Gym Diary application in a gym environment. I would be able to collect statistics for the gym on what users are using what equipment. Also access the durability of the NFC chips as many people would be using them.

This would also allow the application to be thoroughly tested as all the users would be new to the app. So it would be interesting to see if any errors would come up and what shortfalls people may feel the Gym diary has.

How has the project matched initial designs?

Below are the Initial Designs for the Gym Diary application. From the focus groups they selected sample A as the design they would prefer most for the application but upon actual implementation of the application. B turned out to be the far better fit. So I decided to use B as my base design. Shown at C is a mockup of how the real application looks running on a Samsung galaxy s3. Image below Fig [38]



Another area that was changed from the initial plan is the version of android that I was developing for. I had originally proposed 2.3.3 Gingerbread as my base level to develop for. But seeing as my test device was the Samsung Galaxy s3 it made sense to develop for the latest firmware on offer 4.2.2 Jelly Bean.

Using a use case study methodology in the early stages really helped get my idea of the ground, this combined with focus groups allowed me to capture key important areas that should be found with in a Gym Diary application. The questionnaires helped determine a look and feel for the application. From here is when designs began to take place.

Using the Eclipse IDE with the android SDK was a wise choice as the program creates a simple interface which mixes multiple technologies such as java class files and XML files all in one. Due to the SDK plug I was able to drag and drop images directly to XML pages without the need to manually type in the code for each object used.

I am very confident that my application Gym Diary could be further developed with a little extra functionality and be rolled out in to gyms within a matter of weeks.

Future Work

One of the ideas for potential future work for The Gym Diary application could be franchising out the application to individual gyms. Where they could have their own brand and logos live twitter feeds and updates direct to their users. This could be used to harness information from its users and about the gym. The applications would tell the gym owners which exercise equipment was used the most. For the users they could get updates of classes that are upcoming from the gym if there are spaces available. Due to the design of the application this could easily be implemented without too much extra work.

Similar to this idea is having a computer from the gym acting as a server that hosts all the user's information, users could log in and out and access there recorded workouts live from the system.

Adding additional pages that would link to diet management tools and calorie counters. I feel that the additional of these two aspects could make the application much more diverse in its target audience as not only will it capture users who wish to record their workouts but also people who are looking to implement a healthy routine though the use of exercise and diet.

Possible improvements

I would like to add extra functionality to cater for cardio exercises, more specifically recording time as a unit of measurement and distance covered for these exercises.

Also the ability to link the app to Facebook, so users could set when they have been, and the potential for Facebook integration so they could edit their workouts through Facebook. Obviously this would require a plug in to do this but this is a potential improvement linking exercise with the mainstream social media.

I would have also liked to include demonstration videos for first time gym users. The application would have guide workouts preset for beginners. This would allow for new gym users to get into good routines in the gym and if they were unsure of an exercise they could select the exercise and see a video of how to perform the exercise on their phone.

An additional feature I will look to implement is my own NFC Tag write from within the application.

Conclusions

The summary of aims for this project was to create a Gym Diary application that incorporated the use of NFC technology. This would be managed using an SQLite Database and the aim was to use this technology to streamline a user's experience when using the gym as they would not have to sit and type in each exercise that they wish to add to the application because with the Gym Diary app you can simply scan an NFC chip to store exercises.

This report has shown that with adequate planning for the development of an application you can create a useful tool that will save time in real world application in the gym. Through the use of use case design, focus groups and extensive research into the NFC technology I have been able to create a fully working Android Gym application utilizing NFC technology.

Looking at the design phase you can see that by choosing a bright colour scheme that is easily identifiable i.e. Green buttons for adding action, red buttons for deleting action and blue buttons to view this gives the application continuity on all its screens. Extensive testing has shown that the application is very robust. It carries out all of its intended tasks with error checking systems in place to prevent users from making a mistake while navigating the app.

Key lessons learned from this project phase are to never underestimate the size of the project, as I fell behind in the development stages and I made it hard for myself to get back on track. I feel this has given me better skills in approaching a problem. Take a step back analyze do you research construct your solution as opposed to diving straight in, As this can often lead to disappointment and missed deadlines.

The process of this project has given me a greater understanding of java and programming in the android environment and I have reached a point where I am really behind my idea of the Gym Diary and I definitely think there could be a release of this application to the public.

In conclusion I feel that I have delivered on all of the goals that I set out to achieve and have provide scope in which to build the application on and take it to the next level.

Reflection of learning

The Initial problem was learning to integrate the NFC capability within my application, first the research was to learn about the technology and determine whether this could be used to enhance my application. This process of research first allowed me to gain a thorough understanding of how NFC technology works and what factors I would need to consider during the development of the application. This allowed me to take a direct approach during my implementation as I already understood what needed to be incorporated for the application to be built to specification.

The application building process has taught me that even a small scale project can turn out a lot more complex than you had ever imagined. For example when I began coding the Gym Diary app it started off as a few files and folders. By the end of the application I had ended up with over 140 files in 30 different folders. I believe I took the right steps in building the application by coming up with a design plan first, I think that had I tried to jump into the project I would have easily got lost in the size and scope of the project.

By keeping the application development tailed by user feedback, kept the project true to its original goals. The result of this is a simple yet effective application that fits its intended purpose.

A side bonus from this process is the new skills developed in programming for the android mobile phone operating system. These skills can now be transferred into new projects to create more adventurous applications. This will decrease the time in which complete future applications due to not having the learning curve of coding in a new language.

Due to the nature of the design this project can easily be built upon it further enhance existing features or implement entirely new functionality.

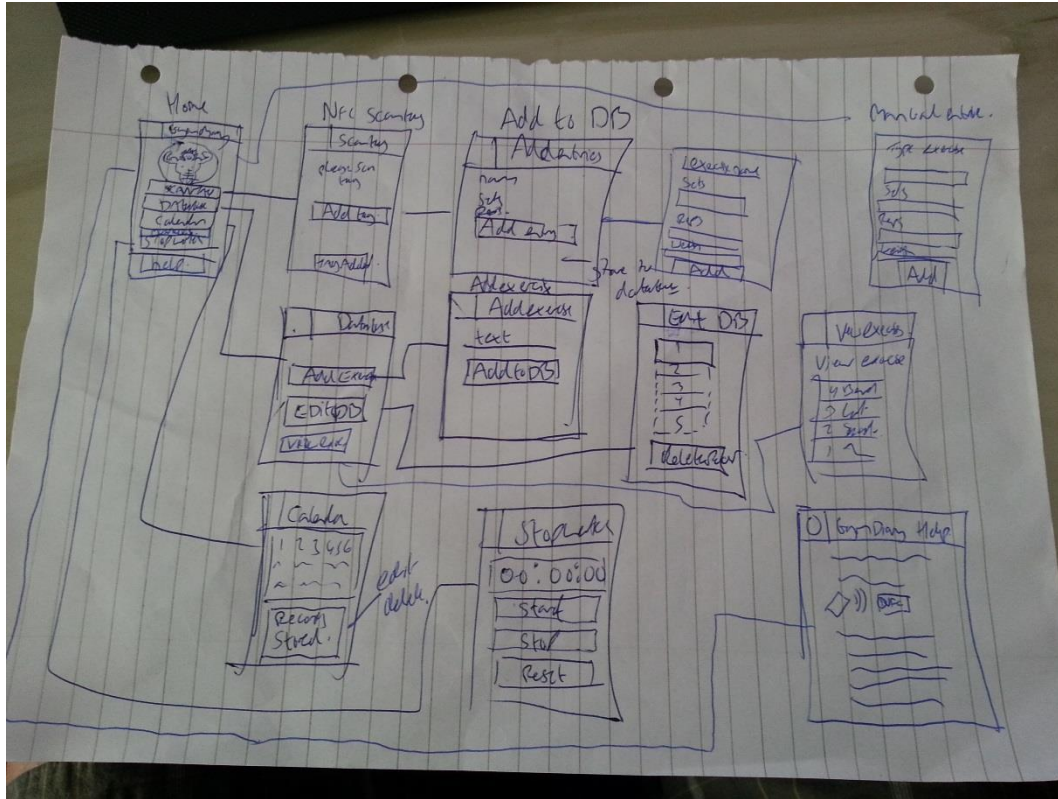
Table of Abbreviations

NFC- Near field Communication
OS- Operating system
UI- User Interface
SQLite- ACID-compliant embedded relational database management system.
QR codes- is a matrix barcode (or two-dimensional code), readable by QR scanners.
SDK- Software Development Kits

Appendices

Please see Additional appendix Brett Stevens- scm9bs code appendix.pdf

Initial design Plan on paper



References

Android- how to change format of chronometer?- Stack Overflow. [2013]. [ONLINE] Available at: <http://stackoverflow.com/questions/4152569/how-to-change-format-of-chronometer>. [Accessed 03 May 2013].

Android Hive. [2013]. Android Populating Spinner data from SQLite Database. [ONLINE] Available at: <http://www.androidhive.info/2012/06/android-populating-spinner-data-from-sqlite-database/>. [Accessed 03 May 2013].

NFC Programming [2013]. NFC Programming in Android- Developer.com. [ONLINE] Available at: <http://www.developer.com/ws/android/nfc-programming-in-android.html>. [Accessed 03 May 2013].

Random Thoughts: Read/Write NFC tag in Android. [2013]. Random Thoughts: Read/Write NFC tag in Android. [ONLINE] Available at: <http://elsoufy.blogspot.co.uk/2012/10/readwrite-nfc-tag-in-android.html>. [Accessed 03 May 2013].

Stack Overflow. [2013]. java- Can't add listeners to CalendarView events- Stack Overflow. [ONLINE] Available at: <http://stackoverflow.com/questions/12969944/cant-add-listeners-to-calendarview-events>. [Accessed 03 May 2013].