National College of Ireland

BSHC2

2014

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colin**  **Allen**    **15540607**  Github username: **Colin303** | **Keith Feeney**    **15015556**  Github username:  **keithfeeneyNCI** | **Patrick Lawlor**    **15014606**  Github username:  **Paddyyyyyyyy** | **Fearghal McMorrow**    **15014584**  Github username:  **Fearg6** | **Cedric Vecchionacce**  **15011071**  Github username:  **CedricVz** |

Audio Acoustic Assistant

Technical Report



Table of Contents

[Executive Summary 4](#_Toc480374831)

[Definitions, Acronyms, and Abbreviations 4](#_Toc480374832)

[1 Introduction 5](#_Toc480374833)

[1.1 Background 5](#_Toc480374834)

[1.2 Motivation 5](#_Toc480374835)

[1.3 Project Overview 5](#_Toc480374836)

[1.4 Target group 6](#_Toc480374837)

[1.5 Technologies 6](#_Toc480374838)

[1.6 Distribution of tasks 6](#_Toc480374839)

[2 System 8](#_Toc480374840)

[2.1 Requirements 8](#_Toc480374841)

[2.2 User Requirements Definition 8](#_Toc480374842)

[2.3 Requirements Specification 9](#_Toc480374843)

[2.4 Functional requirements 9](#_Toc480374844)

[2.5 Use Case Diagram 9](#_Toc480374845)

[2.5.2 Requirement Type of room (ft2 or mt2) 11](#_Toc480374846)

[2.5.3 Requirement Select User Role 12](#_Toc480374847)

[2.5.4 Requirement Input for calculations 14](#_Toc480374848)

[2.5.5 Requirement Output from calculations. 15](#_Toc480374849)

[2.5.6 Requirement Comparing Results (req 3 and 4) 16](#_Toc480374850)

[2.6 Non-Functional Requirements 17](#_Toc480374851)

[2.6.1 Database Requirement 17](#_Toc480374852)

[2.6.2 Navigation Requirements 17](#_Toc480374853)

[2.6.3 Performance/Response time requirement 18](#_Toc480374854)

[2.6.4 Availability requirement 18](#_Toc480374855)

[2.6.5 Recover requirement 18](#_Toc480374856)

[2.6.6 Robustness requirement 18](#_Toc480374857)

[2.6.7 Maintainability requirement 18](#_Toc480374858)

[2.6.8 Portability requirement 18](#_Toc480374859)

[2.6.9 Extendibility requirement 19](#_Toc480374860)

[2.6.10 Reusability requirement 19](#_Toc480374861)

[2.7 Design and Architecture 19](#_Toc480374862)

[2.8 Implementation 21](#_Toc480374863)

[2.9 Graphical User Interface (GUI) Layout 22](#_Toc480374864)

[3 Discussions and reflections 23](#_Toc480374865)

[4 Conclusions 25](#_Toc480374866)

[5 Further Development 26](#_Toc480374867)

[6 References 28](#_Toc480374868)

# Executive Summary

This Project is about creating an app that provides the user the amount of acoustic materials they’ll need to perform an acoustic treatment to any room. With many people wanting to become content creators on YouTube, produce their own media or organise small events, there is no app on the Google Play Store that can help with the acoustics of a room or area in a “user-friendly” way. There are apps to measure noise levels but none of them address the above issue.

The solution to this is the creation of the AAA app. The app addresses this niche in the market. The app gets the user type; whether YouTuber, home artist, or event organiser. This Boolean value is transferred to the Advice page. It uses a Login System. The login values are sent and stored in a database. (Potential future use; Users can login to website and see all saved results, email marketing), a room area calculator, room selected; whether bedroom or empty room. In the Recording page, a value is taken from the phone’s microphone. Calculations are done to get the dB value from this input and the amount of acoustic materials needed improve acoustics. The Advice page shows the results; the user type and how much acoustic foam is recommended (based on calculations from previous intents). An image link is also provided, so that users can tap it and purchase the acoustic foam. There’s also a Facebook link that opens the Facebook application if installed (The link opens in the default browser if Facebook is not installed)

## Definitions, Acronyms, and Abbreviations

AAA – Audio Acoustic Assistant

Intent – A page in an Android Application

dB – decibels

RT– Reverberation Time.

# Introduction

The project is for the likes of YouTubers, home-made recording studios and for novice people in presenting an event such as party in a small venue. It is not necessary for the user to have prior knowledge. Sound engineering itself is rather complicated and can require expensive equipment and knowledge to get the acoustics of a room just right.

The aim is to simplify this process by providing an Android app. The app will contain an extensive amount of default figures and calculations, allowing the user to measure the sound pressure level and obtain the right amount of acoustic materials needed based on their acoustic abortion factor, size and dB drop capabilities. Then they will be able to get the required amount and size of materials in their room, such as their carpet or walls. Steps for this will be included.

## Background

The original idea came from Cedric who is an associate sound engineer. In early 2016, Cedric saw there was a niche in the market or such an app. He wanted to create something that would help simplify the process, but was lacking knowledge for technologies and was unsure about how he would go about it.

## Motivation

The motivation stemmed from when Cedric and Keith created a website for the Web Application Development module in our previous semester. It was decided by the team that we would continue this and develop this into an Android application.

## Project Overview

The Project itself is run over 13 weeks. The first few weeks, we were dealing with the Proposal and the Requirements Specification. From Week 4 – 10 we were learning how to use Android Studio as well as learning specific coding relevant to the Project as well as dealing with errors, etc. This was in preparation for writing coding for the app. On Week 9 & 10, a prototype was developed to show our app in the Mid-Point Presentation. After this, we were building on the prototype to develop the app further to the end result.

## Target group

The target group is both men and women, 18 – 40, who run Android versions 4.1 (Jelly Bean) to 7.1 (Nougat). 98% of Android users use these version, with just 2% of users using a previous version. 2.8% are currently using versions 7.0 and 7.1, but this has been included this version to allow for future proofing. (Google, n.d.)

## Technologies

Android Studio

Android Studio uses Java, XML and C++ embedded into the software. Android Studio is used to create apps specifically for Android devices. In this Project, Android Studio is used to create our application.

SqlLite - holds values of login system

MySql,

Java, - programming in Android Studio

XML, - values for layouts, design, manifest

C++, - C-Binding in Android Studio

Adobe Fireworks & GIMP – image editing

## Distribution of tasks

Colin:

* SQL Database creation for login and registration system (Design and Code).
* Internal memory on phone SQLite (minimum usage).
* External Hosted on a 000Host user MySQL.

Paddy:

* General Layouts,
* Facebook and Google APIs for registration and login System. (Design and Code).

Fearghal:

* Created original layout (buttons, TextViews, EditTexts).
* Created different versions of the app with different layouts and designs for testing
* Worked with Paddy to help with the integration of Facebook login and Google sign-in.
* Created Area Calculator.

Cedric & Keith:

* Worked on calculations, Recording page
  + Tried different implements of coding to get the decibel value from device’s microphone.
  + Recording, stop Recording, Playback, Stop Playback
  + Brought variables to Advice Page for calculating for user advice (Kit) and acoustic foam required.
* Fixed coding from other members, started references

# System

## Requirements

## User Requirements Definition

The Customers are people who needs a tool that will help and teach them step by step on how to perform an acoustic treatment to a room for audio production.

Most YouTuber's will work in environments where they will need noise control, they will receive instructions in how to easily and cheaply improve the acoustics and sound quality and use our app to find absorbing materials, e.g. absorption for Chroma screen (green screen) as an example, which is important to CGI (Computer-Generated Imagery) environments, which a lot of up-to-date YouTuber's use for their videos.

Home Artists (producers, singers, musicians) will use our app to acquaint themselves with how to change any room in their house for a perfect acoustic quality when rehearsing or recording their music.

Event Organizers will have many things to do, one of the most important things when they are organizing a place that will have a lot of people in it (like a conference, concert, wedding reception), is to make sure people can be heard and clarity in the voice is optimal, our app will show them where they can put the speaker for best sound propagation and the right materials that should be used in this kind of situations.

The App has a multipurpose design, it easy to use, uses a minimum of space and will have a calculation and recording audio capabilities specifies it to the user's location of use.

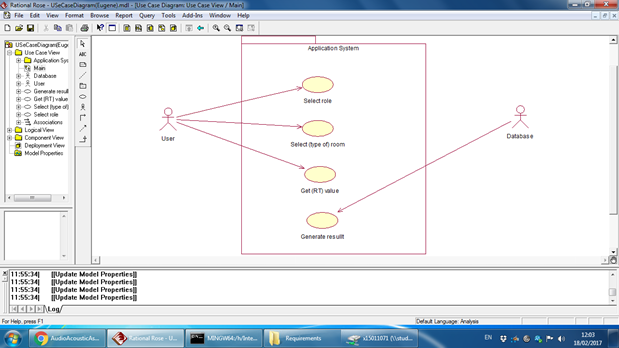
## Requirements Specification

## Functional requirements

This section lists the functional requirements in **ranked order**. Functional requirements describe the possible effects of a software system, in other words, what the system must accomplish. Other kinds of requirements (such as interface requirements, performance requirements, or reliability requirements) describe how the system accomplishes its functional requirements. Each functional requirement should be specified in a format similar to the following:

Short, imperative sentence stating highest ranked functional requirement.

## Use Case Diagram



* + 1. **Requirement Sound input (Digital Sound Processing)**

The application will be able to record an input from the phone’s microphone and read the sound sample to measure the “maximum amplitude” value and convert it into a dB value.

The sound input is essential for the application as the application will generate an initial value for the processes and calculations. The sound input function will make the application easier and more user friendly to those who are not familiar with sound engineering, which make up a large percentage of our target audience.

* + - 1. **Use Case**

**Scope**

The scope of this use case is to perform a “digital sound processing” from a microphone input and turn that sample into a value in dB that can be used by the app.

**Description**

This use case describes how the recorded sample will be measured by the system to generate initial value.

**Flow Description**

**Precondition**

User grants **permission** to use device microphone.

Microphone turned on.

App records sound.

App takes the maximum amplitude value of the sample and turn it into a value in dB, this value represents the initial sound pressure level.

**Activation**

This use case starts when the user presses the sound input button.

**Main flow**

1. The system identifies that the user has selected sound input.
2. The system begins recording.
3. The system reads max amplitude values.
4. The system converts into dB values
5. The system gives the user a dB value in real time.
6. The user stops recording by pressing button again and value is stored in a variable.

**Exceptional flow**

E1: Failed recording.

1. The system identifies that the user has selected sound input.
2. The system begins recording.
3. The system reads max amplitude values.
4. The system cannot convert values due to the recording being invalid.
5. The system asks the user to retry the process.

**Termination**

The system has turned the sound input into a frequency value.

**Post condition**

The system goes into a wait state

### Requirement Type of room (ft2 or mt2)

* + - 1. **Description & Priority**

Select the type of room owned by the user

* + - 1. **Use Case**

**Scope**

The scope of this use case is to change the values in the calculations based on the scenario chosen.

**Description**

This use case describes the different scenarios for creating an acoustic treatment and changes the values in the application depending on the scenario.

**Flow Description**

**Precondition**

The system is waiting for the user to select a scenario.

**Activation**

This use case starts when the user chooses a scenario.

**Main flow**

1. The system identifies the scenario chosen by the user.
2. The system selects the corresponding range of values for the chosen scenario.
3. The system continues as normal.

**Termination**

The system has selected the values for the chosen scenario and the system continues.

**Post condition**

The system goes into a wait state

### Requirement Select User Role

* + - 1. **Description & Priority**

The application will ask the user to select a variety of roles based on their specific needs.

These selections will change the values and actions taken throughout the application so choosing a user role is essential

* + - 1. **Use Case**

**Scope**

The scope of this use case is to change the values and room conditions in the calculations based on the role chosen.

**Description**

This use case describes the different scenarios for creating an acoustic treatment and changes the values in the application depending on the role.

**Flow Description**

**Precondition**

The system is waiting for the user to select a role.

**Activation**

This use case starts when the user chooses a role.

**Main flow**

1. The system identifies the user role chosen by the user. (e.g. YouTuber)
2. The system selects the corresponding range of values and conditions for the chosen role.
3. The system continues to user input phase with a pre-set template from the role chosen.

**Termination**

The system has selected the values for the chosen role and the activity is finished.

**Post condition**

The system goes into a wait state

### Requirement Input for calculations

* + - 1. **Description & Priority**

This is how the user will input their specific values that will be entered into the calculation.

This is essential because without the users input the calculation cannot be accurate for the user’s specific use case.

* + - 1. **Use Case**

**Scope**

The scope of this use case is to allow the user to enter their data easily and for it to work correctly with the values from the database to allow the calculation to function.

**Description**

This use case describes the users input into the calculation

**Flow Description**

**Precondition**

The system is waiting for an input

**Activation**

This use case starts when a user inputs their values into the system.

**Main flow**

1. A user enters their values into the system. (length x width for each surface, floor, walls, ceiling)
2. The system identifies the values entered by the user
3. The system places these values into the calculator

**Termination**

The system has accepted all inputs from the user and they have been placed into the calculator.

**Post condition**

The system goes into a wait state

### Requirement Output from calculations.

* + - 1. **Description & Priority**

This is the output from the calculations, this is what the user will receive after the calculation has used all the values from their input and the database.

This is essential as this output is what the users are using our application for, this output will tell the user what they need to do to improve the sound quality of their room.

* + - 1. **Use Case**

**Scope**

The scope of this use case is to give the user a value which they need to reach to have the optimal sound quality of their room, and to tell them what they need to do to reach this value.

**Description**

This use case describes the output of the calculations that the user needs to fix the sound quality of their room.

**Flow Description**

**Precondition**

The system has received the input from the user and the values from the database.

**Activation**

This use case starts when a user presses calculate.

**Main flow**

1. The system identifies the values from the input and from the database.
2. The system inputs these values into the calculations and receives the output
3. The output is displayed for the user, alongside a set of instructions to reach the optimal value.

**Termination**

The system displays the value and the instructions.

**Post condition**

The system goes into a wait state

### Requirement Comparing Results (req 3 and 4)

* + - 1. **Description & Priority**
      2. **Use Case**

**Scope**

The scope of this use case is to compare values to check if results are correct.

**Description**

This use case describes how results are compared in order to check if RT has been reduced.

**Flow Description**

**Precondition**

The system holds the RT values of the room before and after calculations (with and without absorbing materials).

**Activation**

This use case starts when the user compares results.

**Main flow**

1. The system holds RT value of the room (microphone input).
2. The system holds RT value of the room with absorbing materials.
3. The system compares both values to check if they are the same.
4. If they are the same message displayed “Ideal Reverb Time Reached”
5. Else repeat process to add more materials.

**Termination**

The system displays congratulation message.

**Post condition**

The system goes into a wait state

List further functional requirements here, using the same structure as for Requirements 1 & 2. Most systems would have at least five main requirements.

## Non-Functional Requirements

### Database Requirement

All databases will hold all the preset values (absorption factor, frequency constants and actions to be performed under specific conditions, all data that will be placed into every calculation and as part of the result.

The database creation is essential to the application as the application will not function without a database.

### Navigation Requirements

The navigation requirements include a mix of action bars and selection menus to improve compatibility. Screen relationship design for descendant and lateral navigation as well as ancestral and temporal navigation (Wireframe).

### Performance/Response time requirement

The performance requirements of the application are that the user can navigate the application seamlessly through the different sections. The application shouldn't crash when navigating through the different calculations, and each calculation should operate quickly and produce a result with minimal loading time.

### Availability requirement

The user will need to have an Android device and Google Play Store installed.

The application will have a set of basic calculations that are available to everyone through a free app on the Google Play Store.

A full version of the application will give the user access to a full range of advanced calculations however this full version will cost a one-time payment through the Google Play Store.

### Recover requirement

In the event of the application crashing, the user will be given a prompt to shut down the application and to send the details of the crash to us via email.

### Robustness requirement

The robustness of the application will be strong as it will run on multiple versions of Android from 4.4 - 7.

### Maintainability requirement

The application shouldn't need to be maintained or updated as the calculations and functions of the application are quite simple.

### Portability requirement

The application will be available on a variety of devices. This includes a variety of different smartphones and tablets all with different screen sizes

### Extendibility requirement

The application should be easily extended, Once the Database of values has been created adding new calculations or values will be easy.

### Reusability requirement

The code of the application could easily be reused in other applications, as the database and GUI code could be reused to create a similar app that uses calculations outside of audio acoustics.

## Design and Architecture

The focus on why we chose this **architecture** is because they are essential to our apps **creation, development** and **success**. Every successful application needs a simple and effective **home screen** with a **logo** and a **login system** to collect data.

We developed a **UI**/**UX** that satisfies the user in a friendly (simple) way and encourages them to use our application.

There will be **two versions** of the application, a **freemium** version with ads, and a **premium** version without ads, both will be separate uploads on the google play store. It will not have all options and is mainly for testing our product, it has mandatory advertisements that will help keep our application monetized (ads and banners included code embedded).

The **premium** version has all the content and is kept up to date with software improvements and changes as they happen under a fixed price.

On the **categories page**, the user can choose their own specific user role, from **YouTuber**, **Home Artist** or any **Event** regarding audio and acoustic enhancements. On the **type of room page**, the user will have two options an **empty room** or a **bedroom**. Depending on user’s category and room type values will be stored holding **dB drop capabilities**.

User is forwarded to the **recording/meassurement page**. Here we use the phones **mic** **input**, and androids “**AudioRecord**”, “**MediaRecorder**” classes, to record sound samples into an external temporary file, and from this file obtain the max amplitude (binary value) recorded using androids **getMaxAmplitude()** method. This value is **stored** in a **stringBuffer[]** and converted into a dB value by applying the following formula:

**dB = 10log(n1/n2) ------- original formula for decibels.**

By modifying the formula to work with our values and convert the amplitude value we get:

**dB = 20 \* Math.log10(getAmplitude () / reference (0.1)**

Values inside the array are displayed in a **TextView()** every **1000 ms** to create a **real-time response** feeling to the user (measuring).

Once the **recording stops** the system takes this **initial dB** value **to** proceed and calculate the **amount of acoustic foam** (absorbing material) needed by applying the following algorithm:

**acousticKit = dB - (items)**

**20**

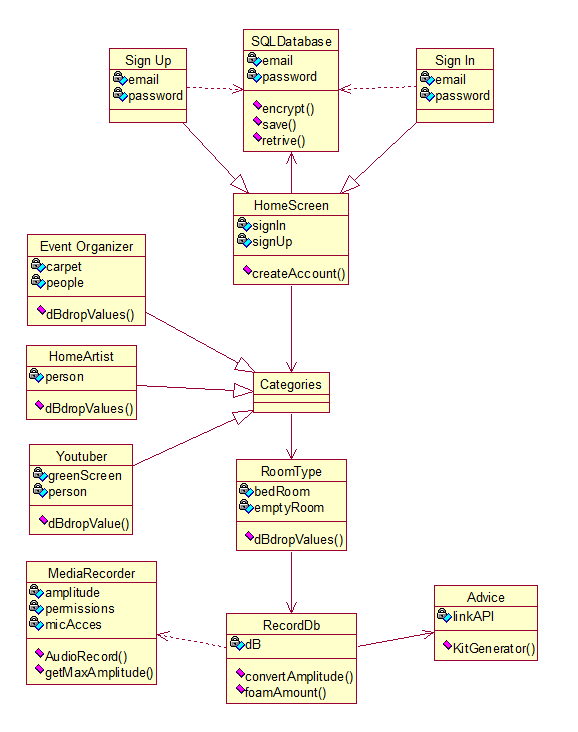
**-dB= value obtained from room record. -20= acoustic foam dB drops per mt2**

**-items= absorbing items like green screen and carpets.**

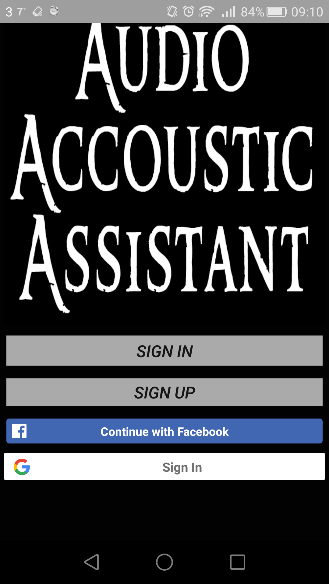
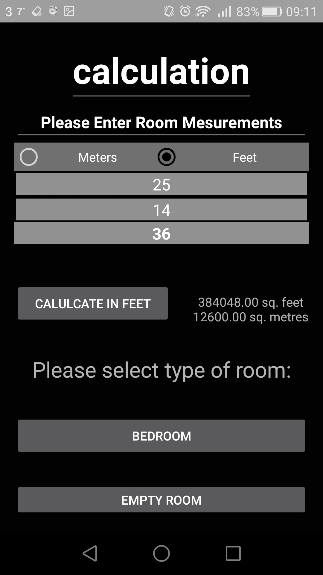
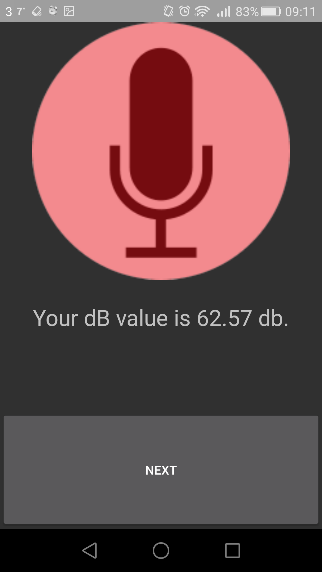
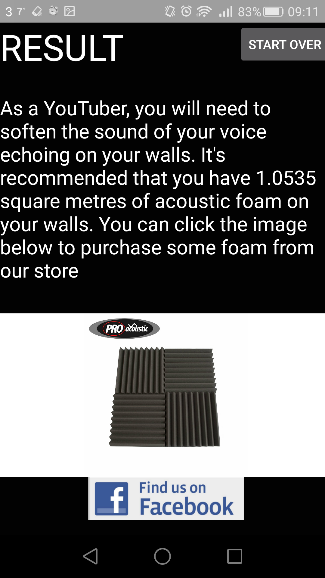
The resulting value is displayed in the advice page as a string variable **“Kit”** with the amount of foam/material needed in mt2, along with items like green screen and carpets (items mentioned also have absorbing capabilities).

All items outputted inside the Kit will have links for affiliate marketing providing the user with an option to buy them. (APIs provided by Affiliate programs).

## Implementation



## Graphical User Interface (GUI) Layout

(Allen, C; Feeney, K; Lawlor, P; McMorrow, F; Vechionacce, C; National College of Ireland, 2017)

# Discussions and reflections

We found that communication could be difficult. The team was split between two classes with shifting timetables and had few in person hours to communicate on not only this coding project, but also a business plan for another module. This ate into a lot of time in the beginning, there was a lot of documentation that everyone had to have some part in.

This lead to three individual teams working separately, Paddy & Fearghal, Colin, Keith & Cedric. This would cost time later as these various versions of the application had to be built into one working version.

Therefore, the bulk of the coding was shifted towards the end of the allotted time span.

We used technologies that we had including how to use Android Studio. We implemented what we’re learnt over the past two years and put it into the Project. We used various coding languages such as Java, Php, Json, Sql etc.

The team member’s coding ability has dramatically increased. Team members also improved their commenting and referencing both in coding and with Harvard referencing within the various documentation.

In Week 1 – 4, The Project development process began with the Proposal and Requirements Specification. From Week 4 - 7, we were learning about Android Studio, watching tutorials, finding out how it all worked.

Week 8 & 9 involved us creating a prototype for the Mid-Point Presentation. Week 10 - 12 was used by getting the behind-the-scenes coding working, along with improving the layout and design. Testing also occurred.

We all felt the Client Presentations was a very positive experience. We felt that it was a way to get an outsider’s point of view on the application and visually how it could be improved. We really appreciated the opinions of people at the presentation and their input.

The Team Project module was a little daunting. We would like to be more informed about what ideas that other people have had to get our creative juices flowing. Also, we felt it wasn’t fully clear about what we needed to do, in terms of how complicated our app needed to be. A suggestion made by a team member is to have a day reserved for the Team Project and independent work. 6 hours of class time is just not enough.

**Git hub issues.**

Git hub was supposed to facilitate group work. It failed to do so.

The problem with git hub is that it doesn’t allow users to work side-by-side, when this is exactly what we were trying to achieve through git hub.

Essentially, the problem is that multiple developers cannot work on the same project at the same time. If all five of us start from a base version 1.0, we will all end up with separate versions. 1.0a, 1.0b, 1.0c, 1.0d, 1.0e. If someone uploads version 1.0a, and then afterwards the next person uploads version 1.0b, all the work in version 1.0a will be wiped.

This means to move a project forward, development must be done in stages.

Starting development at 1.0, one developer builds to version 1.1, then another developer takes version 1.1 and adds his part and passes on version 1.2 and so on.

This slows down development potential and wastes manpower. We separated into two main teams to alleviate this issue, but that resulted in one team waiting for the other team to be finished their work on the project. Or this led to everyone building 5 separate versions and rushing to integrate them together towards the end, which is where the major faults of android studio arose.

**Android Studio**

Android studio was the biggest challenge we had. Android studio is nothing like any of the previous IDE’s we have used before.

Android studio does a lot of things right, but the worst thing about it, is that android studio allows you to develop on multiple Gradle versions.

Gradle is the compiler for android studio, and Android studio supports multiple versions of gradle.

So, when one person is building a login system on gradle version 2.3.X for example, and then someone else wants to build another system on gradle version 2.3.Y.

Then, when putting these two versions together, one of them isn’t going to work because one of them is going to have to function on a different gradle version that the one it was built on. This requires an in-depth understanding of android studio that is simply not possible for beginners.

Another issue that is consistent throughout Android studio development is that it requires a lot of restarts. Sometimes certain imports will throw errors and require a countless number of fixes to even get it working.

In conclusion, Android studio is okay for android development for one developer, however when you try to combine code and work side by side the significant issues arise.

# Conclusions

The advantages of the Project were working in a group and having different ideas and visions about how the app should look. It was also really refreshing that everyone in the group were committed to working, unlike previous projects we’ve all been in.

One major disadvantage was Android Studio and trying to run them on the college computers. By the end of the Project, we completely gave up on the college’s computers and just used our own personal computers to run Android Studio.

Our limit of the Project is that we weren’t technically savvy with Android Studio and the vast amount of coding and rules (such as having the relevant SDK files, the tags for XMLs, etc.) Ideally, an entire year course, solely for Android Studio would be required for some of the ideas we’ve had. A lot of things were too complex or just wouldn’t work. It’s as if Android Studio is its own world of coding.

In the end, we did what we could with the time and resources we had.

# Further Development

There are many options that the app could progress with. Such examples include:

* Room Measurement by using GPS or photograph
* Instead of using text field inputs to calculate room volume and Ideal RT, we’ll use GPS component on phone to calculate volume of the room.
* Virtual Reality / Use of camera
  + Instead of a user depending entirely on the phone’s microphone. With the help of a virtual reality headset or when the user pans the camera, the user could see sound waves and how they bounce off specific surfaces. The user could also see if a surface was changed, how that could impact the sound wave on that surface.
* Radio equipment
  + If a user has a radio (e.g. Bluetooth) microphone or camera, the above could be implemented, but using external devices instead of native ones.
* Professional equipment
  + If a user is a DJ, for example, they could link their Android device with their equipment. The equipment would automatically adjust, based on the results on the app, to produce the best sound quality.
* Statistics
  + Collection of information from users could see which is the most used room type and surfaces. Defaults could then be added, making the app faster, based on these statistics.
* Request a professional
  + If the user is completely perplexed and would simply just want someone who would do it for them, the user could request a call to a sound professional who could assist them.
* Instrument Tuner
  + By customer demographics and feedback, we’d be able to see if a tuner for musical instruments is what users would like in the app. There are many apps on the Play Store that are instrument tuners, but depending on the user’s experience, they may not like to go into another app. This is not required on the current version as the app itself is not a tuner.

# References

Balsamiq Studios, LLC; McMorrow, F., 2017. *AAA - Grid.* [Online]   
Available at: https://epiccool.mybalsamiq.com/projects/aaa/grid  
[Accessed 17 February 2017].

Google, n.d.. *Dashboards.* [Online]   
Available at: https://developer.android.com/about/dashboards/index.html  
[Accessed 8 February 2017].

Hagelskjaer, C., n.d.. *Reverberation time (RT 60) – what is it and why is it important?.* [Online]   
Available at: http://www.hzandbits.com/articles/recording-studio-project-index/recording-studio-design-theory/reverberation-time-rt-60/#.WJnJ8TuLSUl  
[Accessed 7 February 2017].

Hzandbits Sound Effects, n.d. *Reverberation time (RT 60) – what is it and why is it important?.* [Online]   
Available at: http://www.hzandbits.com/articles/recording-studio-project-index/recording-studio-design-theory/reverberation-time-rt-60/  
[Accessed 17 April 2017].

Allen, C; Feeney, K; Lawlor, P; McMorrow, F; Vechionacce, C; National College of Ireland, (2017), *Screenshots*  [Accessed 19 April 2017].