



BIRMINGHAM CITY
University

**Student Attendance, Behaviour and Progress: A
Mobile Monitor**

**BIRMINGHAM CITY UNIVERSITY,
BSC (HONS) COMPUTER SCIENCE**
SUPERVISOR: DR. DANIEL DOOLAN

Alexander Treacy
S15106436

Abstract

Within education, there are applications for learning and the process of students receiving work from teachers, but none of which could be found for allowing students to sign in using a QR code generated on their device. SPLAaT allows for teachers to keep in control of signing students in using their mobile device in addition to giving teachers options for updating/keeping track of their own lessons alongside students viewing their own lessons.

Using the Android Studio IDE made it possible to implement an application and user interfaces within different activities. All backend data and authentication were handled using Google Firebase as the resources available allowed the application to utilise authentication, cloud messaging, crash analytics and a Realtime database.

Acknowledgements

I would like to thank my honours project supervisor, Dr. Daniel Doolan for his consistent and helpful communication and feedback throughout the entirety of this project. Your understanding of students along with background knowledge, valuable suggestions and feedback aided the project greatly. I would also like to take the opportunity to thank all participants involved in user testing and completion of the questionnaire.

Contents

Abstract.....	1
Acknowledgements.....	2
1.0 Chapter 1: SPLAaT: Introduction and proposed idea.....	5
1.1 Introduction	5
1.2 Aim	5
1.3 Product.....	6
1.4 Rationale	7
1.5 Platform justification	7
1.5.1 iOS	8
1.5.2 Android.....	8
1.5.3 Other options	8
1.6 Resources	9
1.7 Schedule.....	9
2.0 Chapter 2: Background Research and methodologies.....	11
2.1 Background Research.....	11
2.2 Software methodologies.....	11
2.3 System Requirements	12
2.4 Android guidelines and design elements.....	12
2.4.1 Screen compatibility overview.....	13
2.4.2 Fragmented screen (design and lifecycle)	13
2.5 Similar applications	14
3.0 Chapter 3: Design.....	15
3.1 Preliminary requirements	15
3.2 Design Methodology	15
3.3 User-centred design.....	16
3.4 Design background.....	16
3.5 Application structure/architecture	17
3.5.1 Student account.....	17
3.5.2 Parent account.....	18
3.5.3 Teacher account.....	19
4.0 Chapter 4: Implementation.....	20
4.1 Initial implementation and environment setup.....	20
4.1.1 Android manifest and Gradle.....	21
4.2 Google firebase	21
4.2.1 Authentication	22
4.2.2 Realtime database	22

4.2.3 Cloud messaging	23
4.2.3 Connecting Firebase with SPLAaT	24
4.3 Quick Response (QR) Codes	24
4.3.1 Why QR codes?	24
4.3.2 QR codes within SPLAaT.....	25
4.3.3 Using device camera	26
4.4 UML and implementation analysis	27
5.0 Chapter 5: Post implementation testing and Feedback	30
5.1 User feedback	30
5.1.1 Questionnaire	30
5.1.2 Usability testing.....	30
5.2 Feature testing	31
6.0 Chapter 6: Conclusion	33
6.1 Future of project	33
6.2 Conclusion	33
6.3 Recommendations	34
References	35
Appendix	37
Appendix A: Design and prototyping	37
Appendix B: Application backend	41
Appendix C: User questionnaire and survey results	44
Appendix D: Additional relevant items	45
Copyright waiver	48

1.0 Chapter 1: SPLAaT: Introduction and proposed idea

1.1 Introduction

What lead to the creation of this project was the pursuit for an application which would help teachers to determine which students would require more help than others. Results from this research led to the understanding that such applications existed, but none of which had student information in one place which is accessible to teachers. This led to the idea of creating an app which could have student information available to teachers whom share the same student(s) in classes. Then commenced the search for which assets were available for parents and found that Conner, (2003) states that parents mainly have end of year/term reports to rely on when trying to examine the progress of their children at school, which are based entirely on the teacher's memory.

The inspiration to construct this application originated from an exploration into applications which would allow for enhanced teacher-parent communication. None of which would allow for the teaching staff to use a simple, yet effective, user interface to be able to update students' progress within lessons and allow teachers and parents to view the student's progress, lessons, attendance and timetables. Additionally, Cacho (2016) states that mobile literate student teachers tended to use a wide array of mobile applications for academic and non-academic purposes, meaning that if an application such as this were implemented into schools, it can be assumed that teachers using smartphones would also use such an application.

Regarding changing from a paper-based system to one based on using a new system, Iio (2016) states "The number of attendees without paper form submission gradually increased especially at the first couple of lectures.". If this would apply to a system which uses QR codes, then the outcome of this application would be a success. Therefore, the application would need a way of signing students in to lessons on the application, hence the scanning of a QR code on a student or teacher device to sign the student in as present.

1.2 Aim

This Android application is being created to bridge the gap between parent and teacher communication regarding students. It will allow for teachers to update the students' progress and

reports via a mobile application and for the parents of each student to view this progress. According to Desai et al., (2016) having a system in which teaching staff can refer to when deciding what is correct to teach to a specific class, will allow teachers to use specific clustering techniques of students to identify a subject in which most of the class is not performing well and hence plan extra classes for that subject.

The communication within the application will be one-way so that teachers may give inputs to the application such as amending a student progress, but parents will not be able to give inputs (to be viewed by teachers) into the application as this may disrupt teaching staff whom do not have time to investigate every matter risen by a parent.

1.3 Product

The product being created is a student progress, lessons, attendance and timetable application (SPLAaT). To know whether this application would be used by teachers and parents, a questionnaire will be provided to the appropriate target audience. This questionnaire will ask for feedback on the proposed idea for the application, and questions on whether this would be useful when tracking a student's progress. This will require constructing a questionnaire for parents regarding student tracking and using the results to determine whether the application would be useful in this form as well as whether the app would help record their child's progress, juxtaposed with a second questionnaire which will be filled out by teachers for an indication on whether the application would benefit teaching staff when monitoring students' progress.

Alternative preferences for utilising a server were considered such as using a Raspberry Pi 3 model B as the server but doing so would mean that the hardware would always have to be running and may consume a lot more power than is required for this application. Another consideration was Amazon web services which allow for users to choose how much space is needed, however, this option was not cost effective. For these reasons, Google Firebase was researched and as this system is user friendly, cost effective (free to use for a project of this size) and most importantly would suit the requirements needed for this application.

The student information server will process student data into the application to be viewed graphically. The user interface will allow the user to sign in as either a teacher, parent or student and prompt the user to sign in. Appendix A, figure 12 shows concept sketches of before and after the

login screen for a student account. Once logged in, the student can edit their dashboard to display their chosen information such as attendance or their next lesson. All other information will require the user to open the respective page via the buttons.

1.4 Rationale

With this application operating at its full potential in an educational institution it will bridge the gap between parent and teacher communication without having to spend time communicating i.e. meetings or email. This application will allow teachers to update information on a student which can then be viewed by the parent of this student and all other teachers whom share this student in their classes. Additionally, each student will be able to view their own information. It will benefit teaching staff in terms of working together to create a profile for each student while not having to scrap any paper-based systems. Other applications of a similar nature, such as ClassDojo, do not allow for a parent account to be created unless the child's name is entered into the application and accepted by the teacher selected. In comparison, SPLAaT could use a unique code in the future, which will be given to each parent and will allow them to create an account which will already be linked to the student associated with that individual.

Using a QR code is a cost effective and functional way for students to be able to sign in as opposed to other solutions such as biometric or RFID systems. Regarding RFID, Kuriakose and Vermaak (2015) state: "With this procedure the attendance was registered within a matter of seconds.". But as this method is not cost effective and a QR code can be generated in a short time, RFID would be inconvenient to use.

1.5 Platform justification

The development process and completed application are not the only problems we encounter, (Myers, 2010) states that "you'll need to consider whether or not your app requires any kind of storage server". Although setting up XCode for iOS development is easier than installing the Android SDK (Jansen et al., 2016), the choice was indisputable once personal knowledge was considered (Java programming language) as well as the use of android phones being much greater than iOS devices - "Android dominates the market with a whopping 87%, while iOS comes at a distant second at 12%." (AndroidPub, 2017).

1.5.1 iOS

Being relatively simple to use in the realm of programming, Swift was considered an option to develop this application for iOS devices. Doing so would mean adhering to Apples human interface guidelines (Apple, 2018) which may restrict some graphical content to be implemented into SPLAaT, along with poor interoperability with third party tools (such as those which allow usage of QR codes).

1.5.2 Android

As Android supports Java as the main programming language, this made the choice clear. Choosing to develop for an Android device gives many benefits including utilising Google cloud services. This reinforced the choice of using Google Firebase with an Android application (although Firebase can still be used in iOS development).

Another supported feature of Android (which is not supported by iOS) is the Android Zebra Crossing (Zxing) libraries which can be used to create barcodes locally on a device. This also means that any future development of creating the application for both iOS and Android will have to use either different libraries or a completely different method of generating QR codes (perhaps using a combination of a QR code generator, an image hosting service and an SQL database to reference each QR code).

1.5.3 Other options

Writing the application in Java was a personal, knowledge-oriented choice as it is the most commonly used programming language for writing Android applications. An option which was greatly considered was creating this project as a web application which would utilise HTML and PHP as the programming languages, however, this could cause issues when using push notifications and with offline management of the application once logged in e.g. accessing settings while in offline mode. Another possibility considered was to develop the application using either C or C+ however would require the use of Android studio and the Android Native Development Kit (NDK).

1.6 Resources

The application itself will be created using Android studio as the IDE as it is the optimal software for creating Android applications. To run the application a personal Samsung Galaxy S7 has been used thus far, but within the testing stages many emulators will be used of different devices to analyse the application for potential issues including issues with device compatibility and XML constrained views in the application. Default hardware will be used such as a personal computer to run the required software, however, other services will be outsourced such as using Google Firebase to store a database of students and to run authentication of the application from a server. Other notable software which will/has been used for implementation and design includes: Notepad++, GIMP, draw.io and Microsoft Paint.

1.7 Schedule

Thus far a Gantt chart has been used to ensure the project meets its requirements and deadlines (figure 1). The Gantt chart contains all tasks to be completed, the start date, due date, progress of completion and a colour code for variable importance (will be contingent upon current importance). Additionally, the chart will allow for improved time management and effective allocation of resources based on the importance of each task. Any tasks that have not been included in the Gantt chart will be added later and noted for future reference that they were additionally missing in case of a task being incomplete. "Gantt charts make several behaviours visible, allowing us to see frequently and infrequently visited places, repetitions and outliers, and changes in patterns." (Gupta et al., 2016).

Although the Gantt chart shows all tasks to be completed, not all tasks will adhere to a specific plan as Gantt charts leave space for error and for additional, unpredicted work to be completed. For this reason, a weekly journal was used to cross reference Gantt chart tasks and any additional tasks would be completed in between tasks when the time is available. After the sign in features task was completed in the Gantt chart, many unpredicted new activities were to be created. Similarly, upon creating fragments for each user account, the time to create these pages substantially increased due to each fragment essentially being a new activity to create (XML and new Java class).



Figure 1: Gantt chart.

2.0 Chapter 2: Background Research and methodologies

2.1 Background Research

Tackling the research for this project meant directly speaking to both students and teachers and querying these groups of individuals concerning their experience with applications they have either used in the past or applications they would use in the future regarding tracking student data. Although some research required field tests, other research was undergone by surveying individuals, searching for online resources, collecting general student/teacher statistics, and gathering data on student usage of android mobile devices.

One key field of research delved into was ensuring that the target audience would use such an application and this research was undergone by creating a questionnaire (mainly for parents) to show the usage of current applications and if they would use an application should one exist. Also included in the questionnaire was questions about what the user would like to see in such an application (appendix C, figure 25). Over 80% of respondents from this questionnaire said that they would use a mobile application to track their child's education and under 10% of respondents said they would not (appendix C, figure 26). Assuming these parents (or potential parents) own an android device then the application would be a suitable fit for these potential users.

2.2 Software methodologies

One key methodology delved into was the Rapid Application Development (RAD) methodology. Such methodology allows for a constant stream of feedback at regular intervals and can be closely examined at each stage in the development process. In simple terms the methodology can ensure that design can be changed at any point and the application can be refined to fit the current needs.

For these reasons, a non-conventional approach was taken to create SPLAaT. This approach utilised the main aspects of the RAD methodology but would repeat any stage when necessary to ensure that the project meets its requirements. The development process is split into 4 parts: design, prototype, testing/debugging and maintenance. Splitting development into these components means that each stage can be assessed before moving onto later stages of development. Not using an existing methodology of software development would usually disrupt the development process, however, using an existing model, such as the waterfall model, may be a detriment to the efficiency

of time as there are parts of this model which will not be used in the development, so the verification/acceptance phase would be irrelevant to the development process. Figure 2 shows the process of the application from the fundamental backbone of the application to the user interface on the operating system.

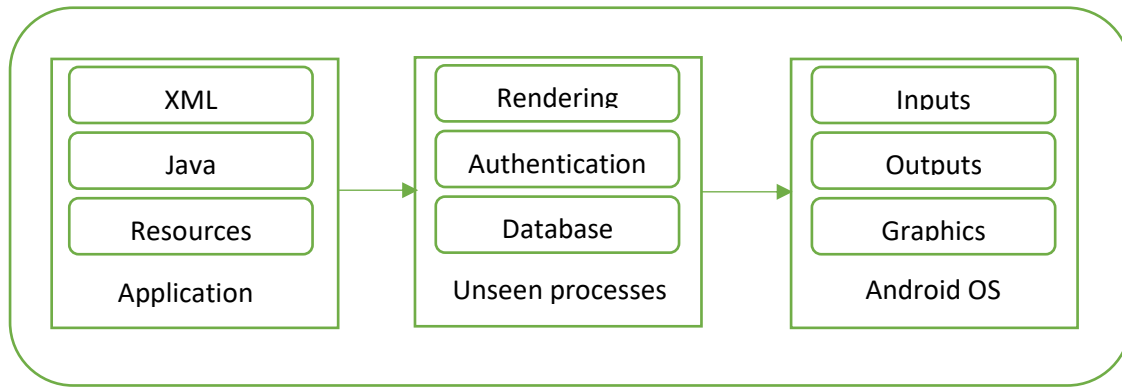


Figure 2: Development process “fundamental backbone”.

2.3 System Requirements

Any user with an Android device above SDK version 16 will be able to run SPLAaT on their device. With the Firebase database containing the bulk of the data to be shown to users, and Firebase authentication being utilised means that only the APK will be required along with an internet connection to be able to use the application. Using an Android based QR attendance system means that the QR codes are created locally on the device from the Zxing dependency.

It is to be created as a native Android application working on Android version 4.1 or higher (SDK version 16) as (according to Android studio 2.3.3) this will work for 99.2% of the devices that are active on the Google Play Store. Although using a mobile application has its limitations, it is the most convenient for all users. “The major constraints of the smartphone computing environment are network bandwidth and traffic costs, interface responsiveness i.e. the time needed to present the user with the data they requested and making efficient use of the battery power available.” Maracic et al., (2013).

2.4 Android guidelines and design elements

When working in Android studio many XML resources and layouts can be applied depending on how you require the resources to be loaded and displayed. When considering different layouts, a linear layout was an obvious option to use as it can be created with ease and only requires layout width and height along with the orientation and any other formatting needed. This decision was changed shortly after designing the first activity as it would not be displayed in the same way on different devices, therefore a constrained layout was used, and constraints were added to the UI elements of all activities (e.g. creating a constraint to have a UI element constrained to a parent UI element).

2.4.1 Screen compatibility overview

As many devices differ in size and pixel density, a design method must be in place to ensure the view is how it is intended to look. Creating a view in XML is only the start of designing the application but all views within the activity must have constraints unless the application is only to run on a single size screen. Constraints were created for each view and each fragment in SPLAaT as this would keep all views both within the viewable screen and would work on any size device (including tablets should the application ever be compatible with this type of device). Developer.android.com explains that using an android system helps achieve density independence by providing density independent pixels as a unit of measurement which can be used instead of pixels. All XML code uses a constrained view with density-independent pixels rather than regular pixels (appendix D, figure 33).

UI elements with XML are declared within the XML activity file but handled within the class associated with the activity using variables to link with the ID of the item in the XML activity (calling the corresponding UI element using findViewById and calling to R class to associate the element with the variable required that the ID is within the scope of the current class).

2.4.2 Fragmented screen (design and lifecycle)

With the minimum API version working on 99% of Android devices (API 16) and fragments being introduced in API level 11 (developer.android.com), means that fragments can be utilised in SPLAaT. Doing so, however, does mean that events within these fragments will have to be handled in separate XML activities and within different Java classes for them to have individual (different) purposes. Fragments are created as subclasses of an existing fragmented screen and uses methods within each fragment (which will override the functionality of an existing method). Therefore when calling the onCreate() or onPause() method, any screens which are not to be changed within the

time a user is cycling through fragments, should stay the same. Rather than using these methods to keep all data in synch, Google Firebase handles all data within these fragments using child/single event listeners.

2.5 Similar applications

Similar applications to SPLAaT do exist, however, such applications contain fundamental differences. One similar application which allows students and teachers to coexist on a single platform is ClassDojo which is an Android and iOS application which employs the use of classroom tools within the application. Some of these tools include sharing classroom notes and moments, allowing students to share their learning experiences and giving teachers the ability to give virtual rewards to students. Although SPLAaT will not employ the same mechanisms, it will contain the same user accounts as ClassDojo and will require authentication from these users.

Another Similar application which uses these same user accounts is Class123. Class123 grants teachers the tools to set goals to students in the classroom and can reward students for good behaviour. This application is more for enhancing the classroom experience. As these 2 successful school applications are dominating in the app store (within the realm of education applications) there is a gap in the market for a student attendance application.

3.0 Chapter 3: Design

3.1 Preliminary requirements

To design the application many mock-ups were created. This included plain paper mock-ups as low fidelity prototypes to high fidelity prototypes created using resources such as using XML to create multiple design prototypes. Android studio was used to create the Hi-Fi prototypes as doing so means that the prototype could be displayed on a mobile device to view the design, giving a more accurate reading of the aesthetics of the design. When using Android Studio and Microsoft Paint to create these design prototypes, many designs were considered (see appendix A, figures 12 and 13). As seen in the prototypes, the colour scheme used was changed when creating the high-fidelity prototypes as the blue colour scheme gave a more professional appearance to the application while also conforming with popular social media sites such as Facebook and Twitter.

One resource used within Android Studio was the Google Firebase tools, which supplied a function for allowing for implementing Google Firebase into the application along with its libraries to utilise when necessary such as using the authentication or the database. Another resource which used imported libraries was the Zxing library which was used to create and scan the QR codes. This library gave the application the tools to encode and decode types of barcode, specifically QR codes. This library was chosen over other barcode scanning resources as Zxing uses a `SCAN_MODE` parameter which can be utilised to determine which barcode is being scanned. “If you know what type of barcode is being decoded, setting `SCAN_FORMATS` to that particular type may result in faster decoding (because the ZXing app won’t try to run through all the barcode decoding algorithms” (Darwin, 2017).

3.2 Design Methodology

When designing any application, whether it be for Android or iOS, a methodology for the design phase is a vital component of the creation of an application. Using the methodology devised in chapter 1 means that any issues encountered will be resolved at the time of finding the issue, hence, removing unnecessary work to be completed at the end of the design implementation.

3.3 User-centred design

Gathering target audience-based data required that themes were tested, and informal interviews were undergone. During one interview an interviewee deemed the theme with the back button within the navigation bar to be simple and understandable as they know that the button has the same function as the common Android device “back” soft key. This was chosen to be implemented into the design as it gives users the option to go to the previous screen rather than just reversing the order that the screens have been shown in (as the soft key back button would).

Ensuring a simple experience for secondary education aged students is an essential part of the user-centred design for SPLAaT. This required the use of simple buttons and fragmented screens to ensure the user can navigate easily through the application with the buttons or fragments. Clearly marked text informing the user where they need to move to within the application would assist the user throughout their experience with the application and would allow for simple navigation for the user.

SPLAaT will use a user interface suitable for all ages and abilities with a simplistic yet functional design. Apps with poor UI responsiveness lead to many user complaints (Gao, et al., 2015), so creating an application for school age students will require positive responses based on the user’s interactions, such as that used by the application WhatsApp, in which users are given immediate feedback of a tick when sending a message indicating that the message has been uploaded. Gao, et al., (2016) states that some users would become angry when interacting with a slow responding system and notes that users would use phrases such as “too lag to use” and “slow as f**k”. It can be assumed that users will want immediate feedback when using any application, so all load screens are kept to a minimum and any loading messages will use rotating load bars as feedback to assure the user of successful loading. Additionally any errors which occur during the use of the application will create an error message on the screen to display to the user (using a Toast to create the message within a try-catch block nested within an if statement).

3.4 Design background

While most design features are not yet included in the application prototype, some aspects of the application were created with aesthetics in mind. Using the blue theme for example along with the

name “SPLAaT” gives a professional yet “quirky” feel to the application so may be aesthetically pleasing to all types of users.

One piece of artwork was created for the application and this is the icon which will be used to launch the application from an Android device. This icon is compatible on all devices as it was imported to Android Studio’s image asset configuration. The icon was created using a combination of both Microsoft Paint and GNU Image Manipulation Program (GIMP) and adheres to the consistent blue theme (Appendix D, figure 30).

3.5 Application structure/architecture

All user accounts within the application follow a basic structure of selecting a login screen to login as either a student, parent or a teacher. From whichever account is chosen the user is required to login or if they do not currently have an account they can register for a new account. Firebase will then check the user’s credentials against the authentication on Firebase or if registering for a new account the user will be added to both the authentication and to the database as one of the three types of user. A newly registering user will have the opportunity to enter their personal details on the registration screen which will contain textboxes for their name, email and relevant information for the type of user.

3.5.1 Student account

As a student, after signed in using their credentials and this has been authorised through Firebase, will be able to use a fragmented screen to cycle through fragments. The first fragment will display any important information for the current day such as an upcoming lesson which will be gathered from firebase using a child event listener and the application itself getting the current day to sort which lesson is next on the student’s timetable (appendix A, figure 14).

Another fragment for the student to view recent data is the attendance fragment in which they can view their current attendance for the day and for the week (appendix A, figure 15). These attendances will be based on the lessons that they have been present for and had their QR code scanned by a teacher in the lesson. All student attendance data will be gathered from firebase and will search the user ID of the student to check if they were present for a specific lesson (appendix B, figure 19).

One fragment is simply the means of getting the child nodes from the student profile in the database (appendix B, figure 20) and will echo this information on the profile page and will display the information entered by the user upon registration. This information is the student first and last names, the email address, and they year that the student is in (this will be used for determining what lessons the student has that day). Once this information is gathered from the database the user will be able to view it within the profile fragment on the student home page (appendix A, figure 16).

Another fragment is used to display the upcoming lessons that the student is to attend. This fragment will also utilise Firebase database as a data source for all lessons which were already written to the database as this information will be fixed between the years in a secondary education establishment. All upcoming lessons for the student signed in will be displayed on this screen along with the day and the time of the lesson(s) (appendix A, figure 17).

The main feature of the student account is the generation of QR codes which are created using the Zxing library for android devices. The QR code contents will be a concatenation of strings of user information. The use of QR codes within the student account is explained in chapter 4.

3.5.2 Parent account

SPLAaT contains an account for parents which was created at the later stages of development. This type of account can view information on their own profile but will additionally be able to view current lessons that their own child is to attend along with their child's attendance (appendix A, figure 18). This was confirmed to be implemented as a feature upon surveying willing participants whom 100% of which answered that this would be a feature they would like to see in such an application (Appendix C, figure 26).

Additional parent information can be displayed on their account such as the lessons that their child currently has along with the timetable of the student. All other information is simply echoed from the Firebase database onto the parent account (parents personal information and child's current student year). Any other parental information on this profile can be displayed on other fragmented screens from the parent homepage.

3.5.3 Teacher account

As Firebase was used for authentication for the application, it was also used to control what happens with the student scanned QR code. This code can be scanned by a teacher and will be inputted into the database via the teacher account, hence allowing students to then view their own attendance as a percentage.

Another teacher fragment includes allowing the user to update their current lessons for the respective day. For example, if a teacher has English at 09:00 on a Monday then this can be inputted into the application as a lesson and will be updated on the Firebase database and echoed back to the user on a specific day when they are required to teach this lesson (appendix B, figure 25). This could also be used in the future for students to see what teacher they have for a current lesson on that specific day and time.

4.0 Chapter 4: Implementation

4.1 Initial implementation and environment setup

Using the Android Studio IDE for coding in Java gives many tools to work with. One of the tools integrated into the IDE is the Android emulator which will run a virtual machine and allow developers to run their application and debug using the Logcat within the IDE. A variety of emulators can be used but the chosen emulator for SPLaAT was the Nexus 4 emulator as this has an average screen size and pixel density across Android devices. Other devices used (not emulators) was a Samsung Galaxy S7 and a Samsung Galaxy S6 Edge and SPLaAT was tested on all these devices with no compatibility issues. Figure 3 shows the Android Studio IDE open along with the Nexus 4 emulator running on the virtual Android machine.

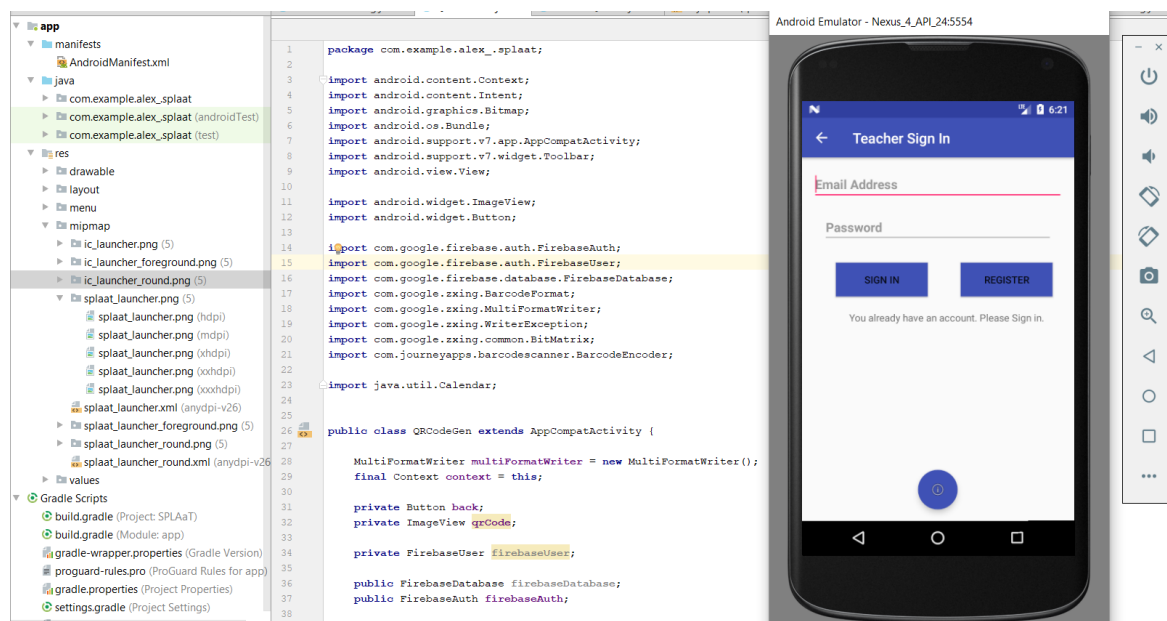


Figure 3: Android Studio IDE with Nexus 4 emulator open.

Upon implementing the Java code for the application, there were many considerations which had to be reviewed. These mainly being the decision to utilise an iOS-based system to implement and create the application or using a version of Android. It was concluded that due to the resources, libraries and tools available when using Android Studio, that to create the application in Java for an Android device was the clear option.

4.1.1 Android manifest and Gradle

The AndroidManifest.xml file contains the bare bones of the application including but not limited to: device permissions, screen labels, the application theme, the application start-up screen, parent activities, and any meta-data for the application/classes. The Gradle for SPLAaT contains all the external data, API's, version information and additionally contains all external resources to be compiled on launch. When the SPLAaT APK is installed onto a device (or locally onto the emulator), the Gradle will compile and implement the appropriate resources, however, when accessing Google Firebase dependencies there were some compatibility issues which were solved by ensuring that all dependencies were of the same version as incompatible types may lead to runtime crashes (see figure 4).

```
dependencies {
    implementation 'com.android.support:support-v4:21.2.0'
    implementation 'com.android.support:recyclerview-v7:27.0.2'
    implementation 'com.google.firebase:firebase-messaging:11.4.2'
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.google.firebase:firebase-core:11.4.2'
    compile 'com.google.firebase:firebase-database:11.4.2'
    compile 'com.google.zxing:core:3.3.0'
    compile 'com.google.zxing:javase:3.3.0'
    compile 'com.journeyapps:zxing-android-embedded:3.3.0'
    androidTestCompile('com.android.support.test.espresso:espresso-core:
        | exclude group: 'com.android.support', module: 'support-annotati
    })
    compile 'com.android.support:appcompat-v7:27.0.2'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.android.support:design:27.0.2'
    compile 'com.google.firebase:firebase-auth:11.4.2'
    testCompile 'junit:junit:4.12'
    compile 'com.google.android.gms:play-services:11.4.2'
}

apply plugin: 'com.google.gms.google-services'
```

Figure 4: incompatible version of one dependency implementation may cause issues.

4.2 Google firebase

Within the planning stages of SPLAaT, many options were considered as a backend and database for the application. After considering Amazon Web Services (AWS) the next to consider was Google Firebase. Google Firebase is a platform used by developers to use as a backend to their application and has been used in popular applications such as Shazam (GoogleFirebase, 2018). Google Firebase was utilised to perform many actions within SPLAaT, these actions include using the Firebase database, authentication, and the cloud messaging service all from the Google Firebase system. After consideration Firebase was chosen as the platform to use for SPLAaT as it was both cost effective and functional. As Google Firebase also has a database system, this was utilised by SPLAaT

as it would make for a secure server for the application. All options had to be reviewed at this stage as data protection may be an issue when working with sensitive data of students but would not be an issue when creating a prototype application.

4.2.1 Authentication

The use of Firebase Authentication gives options for signing in for users and managing different types of users. SPLAaT has 3 types of users which can sign in or register for a new account (students, parents, and teachers) and these users will be added to the authentication pane of Firebase and can be signed in any time after registration. When registering the user will be required to enter a valid email address and password before being added to the Firebase authentication system. When using this type of authentication programmatically, a valid email address was to be used for registering, therefore the original idea of allowing users to sign in using an email address had been kept but additionally meant that they could later have a display name of what to be known as to other users (e.g. a teacher account known as Mrs. Smith).

4.2.2 Realtime database

The Firebase Realtime database allows for developers to add entries into a database either manually or using methods within the application code. All fields and values in the database which SPLAaT uses are created within the application and sent to the database using a combination of database references, Firebase authentication permissions, event listeners and data snapshots. Once data is received by the database, it can then be sent back to the application whenever necessary.

All database data is permitted to be edited by specific users only and some data cannot be called from the database by any users. Handling these permissions involves creating Firebase database rules which deal with authentication, authorisation, data validation and determining permissions for users. Some of the permissions given in the database include read and write rules, validation rules and index rules for ordering querying in the database.

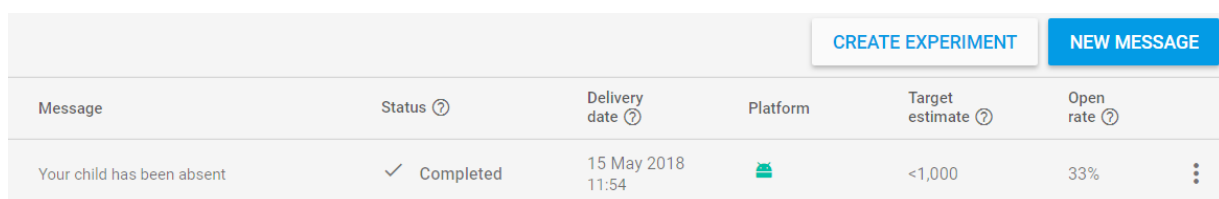
Some data within the database could not be easily accessed as it would be nested within specific database fields. This would mean that when retrieving data, each field key would have to be specifically called for each item to show in the application. The process of denormalization was used on some fields of the database to simplify the data and ensure that the data could be called to the

application using child event listeners and getting all children of a certain node (appendix B, figure 21).

One notable problem faced when creating database reference in the application was when working with fragments the methods would behave in an unpredictable manner. For example, creating the database reference and adding child nodes to write information to a specific field would set this information without any issues, however, when reading from the database within a fragmented screen, the child event listener would only recognise the last child in the node which was being received. After researching this issue, it was evident that the issue was with Firebase being asynchronous, luckily each of the values that were needed contained a unique character (such as a colon in a time) meaning the string received from the database could be checked to see if this character was contained within the string (appendix D, figure 32).

4.2.3 Cloud messaging

Within SPLAaT, push notifications were created using Google Firebase's own cloud messaging service. This service allows to create a push notification to send to users of the application and rules can be set up within the application itself to display these notifications upon a specific event taking place. The event handling for the push notifications were not implemented into SPLAaT, however, push notifications were working with the application and the notification succeeded in being sent to the user. Figure 5 shows the cloud messaging service from the backend of Google firebase and figure 6 is the push notification being received on a user device.



The screenshot shows the Google Firebase Cloud Messaging console. At the top right, there are two buttons: 'CREATE EXPERIMENT' and 'NEW MESSAGE'. Below these is a table with columns: Message, Status, Delivery date, Platform, Target estimate, and Open rate. A single message is listed with the text 'Your child has been absent', a status of 'Completed' with a checkmark, a delivery date of '15 May 2018 11:54', an Android platform icon, a target estimate of '<1,000', and an open rate of '33%'. A vertical ellipsis menu icon is visible at the end of the row.


Message	Status ?	Delivery date ?	Platform	Target estimate ?	Open rate ?
Your child has been absent	✓ Completed	15 May 2018 11:54		<1,000	33%

Figure 5: Cloud messaging on Firebase.

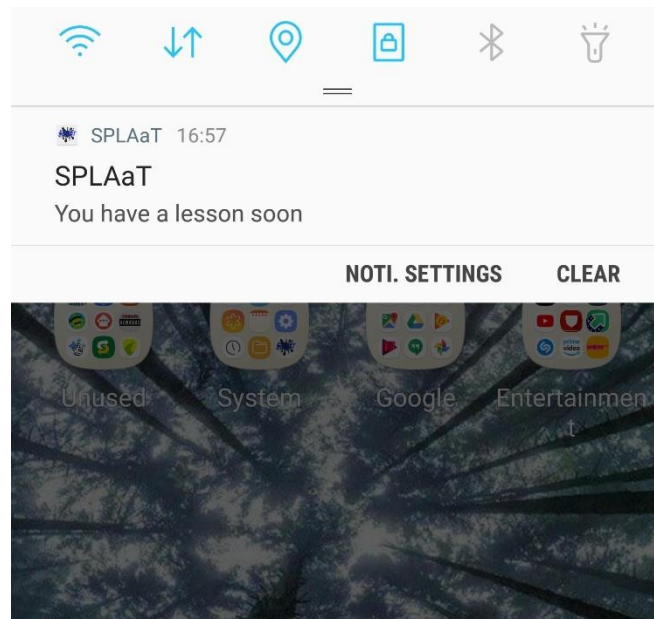


Figure 6: Push notification being displayed on device.

4.2.3 Connecting Firebase with SPLAaT

Android Studio has a built-in tool within the Android Studio assistant which allows Google Firebase to be added to the application with ease (appendix B, figure 19). This tool was utilised in SPLAaT and gives options to add the Realtime database references as well as authentication settings which are also added into the Android Studio Gradle file. Once the permissions are given within the application the Firebase imports are added and can then be accessed by the application.

4.3 Quick Response (QR) Codes

A QR code is a two-dimensional type of barcode with the means of being scanned both vertically and horizontally, this enables a QR code to contain much more information than a barcode (Winter, 2011). The use of a QR code also allows for the code to be generated locally, meaning internet connection will not be required when generating the code.

4.3.1 Why QR codes?

QR codes are a universally used medium with the capabilities of storing large amounts of data which can be used to later store the information embedded in the QR code, in a database. QR codes can be

easily generated on a device due to all modern smartphones having the capability to display this code at a moment's notice. Other possible signing in systems which were considered include: use of RFID (Radio Frequency Identification), traditional barcodes, simple checkboxes within the teacher account, biometrics, or even a location-based system in which the students are automatically signed in.

All the possibilities of student sign in have their advantages, however, not all smartphones have RFID capabilities and this system would not require the student to be physically present, only the device. Similarly, this issue could also occur when using a location-based sign in system as the student does not have to be present and a mobile device would only have to be nearby as location services (such as that used by Google) do not have 100% accuracy. Using a traditional barcode would have been a simple solution, however, does not have the capabilities of being able to store the same amount of data without changing the size of the barcode itself. For these reasons, QR codes appeared to be the best option as a student must be present to be signed in a present, and the code can store up to 4296 alphanumeric characters (Kato, Tan and Chai, 2010) which is more than enough for the system as it currently stands and allows for future development of the application and adding additional features within the QR codes.

4.3.2 QR codes within SPLAaT

The usage of QR codes in SPLAaT is the means of a student signing into their own account and generating a new QR code which is encoded with a concatenation of strings. These strings will consist of the student's user ID, the lesson they are currently in, and the day and time of this lesson. Once this QR code is generated, a signed in teacher can then open a QR code scanner on their device to scan the student in as "present", if the user is not signed in then they will automatically be deemed as "absent". Any Lesson signed in for will be updated on the Firebase database and will be echoed into the users account using event listeners and handling the inputs by converting them to a percentage over a specific amount of time, hence displaying the attendance to the student (or parent).

Upon a teacher scanning a student's QR code the teacher will be informed that they have scanned the student in with a message alongside haptic feedback which uses the vibrate permission on the Android device. Once a QR code is scanned, the teacher account will add the student information to the Firebase database and this is now ready to be displayed on the student and parents' attendance

fragment of the respective accounts. Using some of the built-in hardware features of the device, such as the camera and vibration, requires that these permissions are granted either by the device or for more intrusive permissions, granted by the user. Figure 7 shows the Hardware Abstraction Layer of the Android software stack which is where permissions are granted for using some hardware tools and consists of multiple library modules which implement an interface (developer.android.com, 2018).

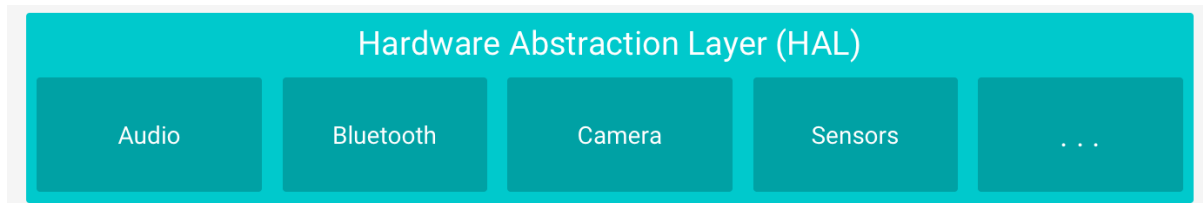


Figure 7: Hardware Abstraction Layer of the Android software stack.

When creating the QR code it was evident that codes could not be manually inserted into the Firebase database without using outsourced hosting service to store or generate the bitmap images. Another option considered would be to store the QR code locally on the internal storage (as this could then only be accessed by the application) of the user's device, however doing so would take up resources and memory on the device. Although it is possible with a service such as Amazon's S3 bucket service to store images to be called on request, a self-generating technique was used for the QR codes instead which would ensure that each user would have a code available even when new users are created. This solves the issue of storing QR codes as the same QR code can be created many times if necessary and will just be discarded when it is no longer needed.

4.3.3 Using device camera

Within the realm of modern smartphones, a digital camera is an accompanying mechanism. This grants users access to be able to take still images and videos with their device and this mechanism can be utilised with the scanning of QR codes. Upon giving the application permission to use the camera, the user is also accepting to allow the application to read any images caught with the camera, therefore, allowing QR codes to be captured on a teacher account whenever the camera is activated (and when it is focused on a QR code).

4.4 UML and implementation analysis

Figure 8 shows a UML diagram of the order in which the application was created including any testing along the way. Some stages in this diagram were repeated where necessary but for the general flow was followed in the creation of SPLAaT. Other stages were spread out over the entirety of the creation process such as populating the Firebase database as this was a progressive task which could only be completed once implantation was complete (or after certain stages were complete). Once final testing of the application was complete, user testing and further investigation to any feature testing was underway. The diagram does not show tasks completed in great detail so some phases of implementation were not included in the UML diagram.

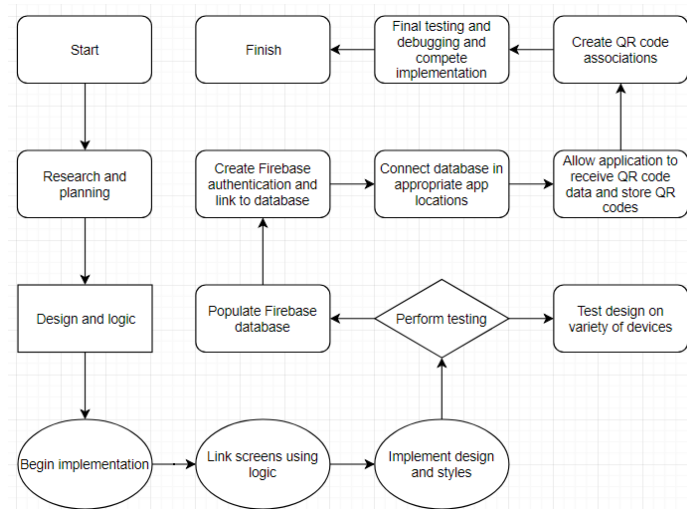


Figure 8: UML diagram of implementation flow.

Figure 9 contains an architectural diagram of the basic blocks of SPLAaT. The teachers, parents and students are all linked to the authentication on firebase and will have to be authenticated either through signing in or registering for an account. Once authenticating, only then will the user have access to the database. Within this architecture, the database can only be written to by the teacher users and the database permissions will only be to read from the database by the rest of the users. This is because only teacher users will have information to update (such as student attendance) and other accounts will only need to receive data from the database (e.g. percentage of a student's attendance).

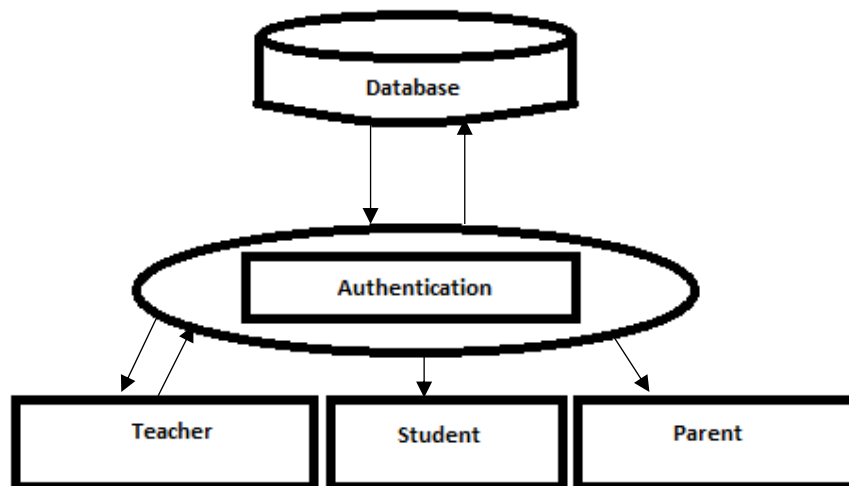


Figure 9: Architectural diagram of basic blocks of implemented product.

The basic application flow is for the user to select the login type (student, teacher or parent), followed by logging in as the selected user and then moving onto the fragmented home page of the selected user (which will display different information). Teachers can scan in students with a QR code scanner and students can generate a QR code to be signed in by a teacher whereas a parent account will not require either of these screens. Users can log out of the application using Firebase authentication whenever deemed necessary. Figure 10 shows the general flow of the application upon launch.

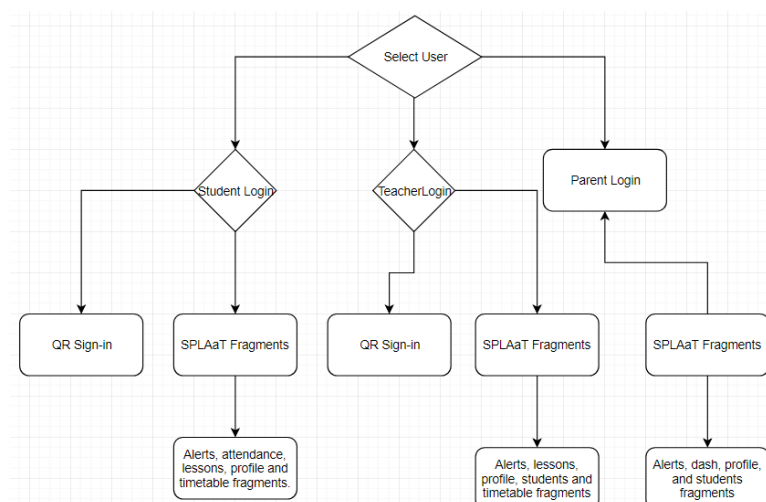


Figure 10: Application flow from launch.

Once a teacher is signed in and they access the QR code scanner they will be prompted to aim their camera towards a valid QR code. Once the code is in frame it will immediately be scanned, and the

Firebase database will be updated with the relevant student data and the timestamp of the lesson the student is present for. Figure 11 shows the flow of the QR code scanner and how the scanning activity is handled.

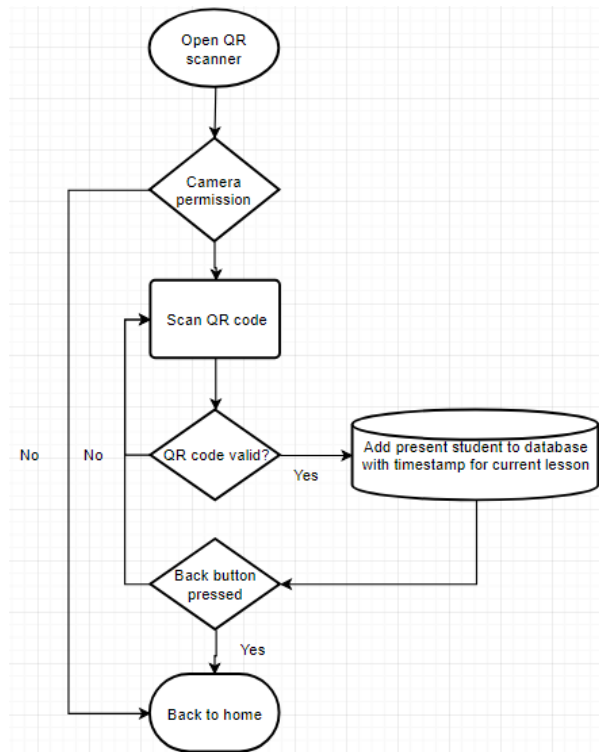


Figure 11: Flowchart of the teacher QR code scan activity.

5.0 Chapter 5: Post implementation testing and Feedback

5.1 User feedback

The user feedback from one informal interview conducted gave an insight to how users would respond when using the application. As no eye tracking software was at disposal, all eye movements and verbal activity was examined closely. Using the think-aloud testing approach an insight was gained as to how the application was to be used. All fragmented screens were immediately noted to be fragments and the user could immediately determine general aspects of the application (text fields, buttons, when to swipe to next fragment etc.).

5.1.1 Questionnaire

Analysing the results of the questionnaire juxtaposed with short verbal interviews shown a great difference in respondent answers. When asked the question “what would you like to see in a student tracking application?” one respondent answered “to be able to see the grades”, and this was the extent of the answer hence the use of a multiple choice questionnaire in which respondents could select any number of features they would want to see and this opened up an entirely different perspective on user needs.

5.1.2 Usability testing

As mentioned, informal interviews were undergone and the think-aloud testing method was used. During this time 3 of the 5 users whom used the application (3 on student accounts and 1 on each other account) seemed intrigued by the QR code icon in the corner and clicked on it. Upon the QR code appearing, users questioned what the QR code was for, it was explained that it was to sign a student in as present, but this was not evident prior to clicking on the floating action button to bring up this screen. Taking this into account, student users may require a clearer indication of what the QR code’s function is as if these users are unsure then a similar issue may occur with teacher users until the button is clicked, and the screen is created. Additionally, student users immediately pointed out that their attendance was currently at 0%, however this was to be edited over time each time the student is signed in as present by a teacher.

5.2 Feature testing

Feature	Expected outcome	Actual outcome	Actions taken
Login as one of the chosen users.	After a user has already registered they should be brought to the home page of the account type being used.	The user is brought to the home page of whichever account they logged in with.	None.
Student generating a QR code.	A QR code should be generated and should be unique for each user and each lesson.	Barcodes were created but were not unique.	The user ID was concatenated with the string in the QR code therefore changing the QR code to be unique (appendix D, figure 6).
Teacher scanning QR code.	Once scanned the teacher should receive haptic feedback and the student attendance will change accordingly.	Once scanned the student was successfully signed in as present (appendix B, figure 4).	None.
Teacher adding new lesson.	Once the add button is pressed the lesson should update in the database and should be displayed on the screen.	As expected the database updated the lesson data for the current day and shows current day lessons on the screen (appendix B, figure 5 & figure 7).	None.
Incorrect password entered.	A “Toast” message should appear informing the user that login was unsuccessful.	Upon entering an incorrect password, the “Toast” message does appear however,	Create an if statement to check whether the credentials are correct before showing if the

		so does the “User registered” message.	user is registered (appendix B, figure 6).
--	--	---	---

6.0 Chapter 6: Conclusion

6.1 Future of project

In the future, the application would be created on iOS to improve versatility. Other features of the application such as payment methods (school trips or lunches) could be implemented to save parents time with sending money to the school. Additional future development would include the application being directly linked to the school database to be updated through their servers, to be able to link students accounts directly to a teacher at the start of an academic year by prompting the students to input a code to enrol them with that teacher, and possibly a bespoke version of this application could be created for schools if required and would contain the school insignia as well as the application icon, also this would mean that the school would have full control over what can be viewed on the application (which student information would be viewable and which data can be edited by teaching staff).

6.2 Conclusion

Overcoming one of the most problematic situations upon implementing this project only strengthened how the attendance system would work. As mentioned in chapter 1, originally students would be able to sign in using a QR code, however, upon creating a fragmented screen for this, a problem occurred with using the Zxing library. However, once the fragmented screen was switched to a separate screen accessed by the floating action button, all problems could be easily solved with debugging the code and creating breakpoints where the code needed to be change accordingly for a new screen (as opposed to a fragment).

If this project was to be restarted from the beginning, the use of fragmented screens would have been researched further as this was a problematic situation when dealing with Firebase database nodes/children. Additionally, more in depth user testing would have been performed prior to the implementation of the application and using a platform such as Adobe Muse would allow to fully design the application to be tested by users and this would give deeper understanding of human-computer interaction.

Overall the project research conducted could have been improved with additional types of testing. One type of testing which was tested on 5 individuals was black box testing, however, this did not give a great insight into how the application would be used. Perhaps white box testing could have been conducted on other individuals with a background knowledge of programming to ensure that all internal methods were fully tested, including performance and stress testing of the application.

6.3 Recommendations

One further development for this project would be to provide student users with the means to customise their own profile and include the use of avatars for each user which can be chosen by the student. A similar concept was used by the application ClassDojo which used graphical avatars to represent each student (appendix D, figure 4). Contextualising text with images is a technique used by many applications and would be implemented into SPLAaT if resources, or time, were not an issue.

Further research could have been conducted regarding sensitive data and the data protection act for students. A system could be put in place to not allow accounts to be created at will but instead would need a unique code from a teacher account on registration to ensure that a parent and student (child of the parent) are who they claim to be.

Following a strict flow while using a Gantt chart was not a realistic goal as most aspects of implementation would take time than assumed so it would not give an accurate, or time realistic estimates. For example, the estimated time to complete the XML design of the application was one month but in fact, stretched out throughout the entirety of creation up to the finished product.

References

- Activedashboards.dynistics.com. (2017). Attendance Tracking. [online] Available at: <http://www.activedashboards.dynistics.com/Colleges/ActiveDashboards/DashboardPage.aspx?dashboardid=c0373680-e723-4393-9851-f9bd744b2aef&dashcontextid=635666787341639485&resetFilt=true> [Accessed 4 Nov. 2017].
- Android Developers. (2018). *Fragments* | *Android Developers*. [online] Available at: <https://developer.android.com/guide/components/fragments> [Accessed 17 May 2018].
- Android Developers. (2018). *Platform Architecture* | *Android Developers*. [online] Available at: <https://developer.android.com/guide/platform/> [Accessed 17 May 2018].
- Android Developers. (2018). *Screen compatibility overview* | *Android Developers*. [online] Available at: https://developer.android.com/guide/practices/screens_support [Accessed 17 May 2018].
- AndroidPub. (2017). Apple Vs Android — A comparative study 2017 – AndroidPub. [online] Available at: <https://android.jlelse.eu/apple-vs-android-a-comparative-study-2017-c5799a0a1683> [Accessed 16 Dec. 2017].
- C. Gao, B. Wang, P. He, J. Zhu, Y. Zhou, M. R. Lyu, "Paid: Prioritizing app issues for developers by tracking user reviews over versions", Proc. of the International Symposium on Software Reliability Engineering (ISSRE '15), pp. 35-45, 2015.
- Cacho, R. (2016). "Student teachers' smartphone academic uses and preferences: Perspectives for mobile-aided pedagogy. Fourteenth International Conference on ICT and Knowledge Engineering". 23-25 November 2016. Bangkok, Thailand. IEEE.
- Conner, C. (2004). "Assessment And Testing In The Primary School". UK, Hampshire. The Falmer Press.
- Desai, A. Shah, N. Dhodi, M. (2016). "Student Profiling to improve teaching and learning: A Data Mining Approach". 23-25 August 2016. Cochin, India. IEEE

Developer.apple.com. (2018). *Themes - Overview - iOS Human Interface Guidelines*. [online] Available at: <https://developer.apple.com/ios/human-interface-guidelines/overview/themes/> [Accessed 15 May 2018].

Firebase. (2017). Firebase. [online] Available at: <https://firebase.google.com/pricing/> [Accessed 17 May. 2018].

Iio, J. (2016). "Attendance Management System Using a Mobile Device and a Web Application". 7-9 September 2016. Ostrava, Czech Republic. IEEE.

Jansen, R.H., Vane V., Wolff, I.G. (2016). Typescript: Modern Javascript Development. (pp. 101). Birmingham, United Kingdom. Packt Publishing Ltd.

Kato, H., Tan, K., Chai, D. (2010). "Barcodes for Mobile Devices". 8 April, 2010. England, Cambridge. Cambridge University Press.

Kuriakose, R.B. Vermaak, H.J. (2015). "Developing a Java based RFID application to automate student attendance monitoring". 26-27 November 2015. Port Elizabeth, South Africa.

Maracic, H. Bojic, I. Kusek, M. (2013). "Accessing student information systems using mobile connected devices". 1-4 July 2013. Zagreb, Croatia. IEEE

Myers, M. (2010) Creating iPad Apps. (pp. 53). Genius Idea Studio, LLC.

Appendix

Appendix A: Design and prototyping

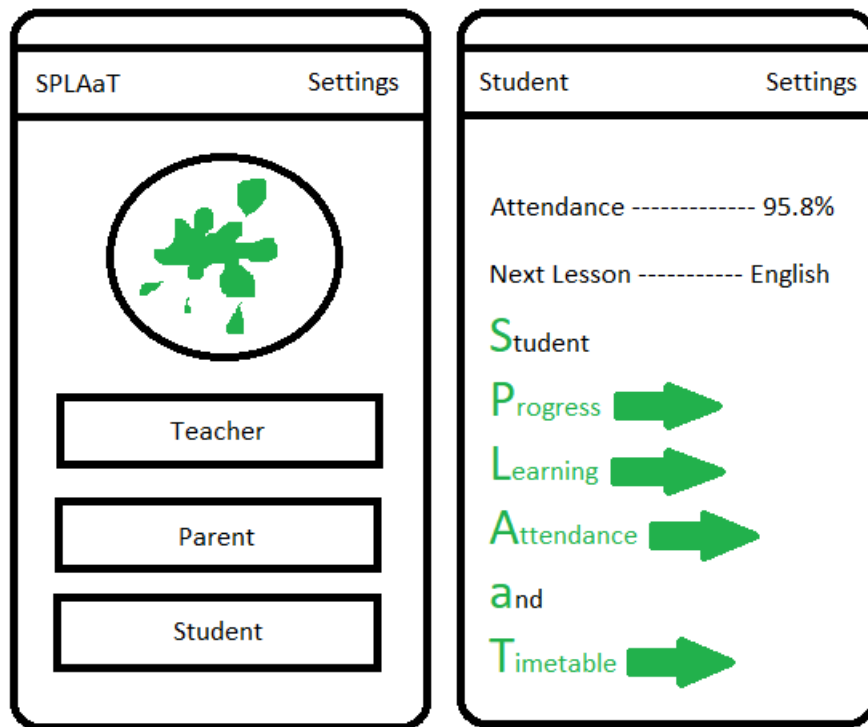


Figure 12: Hi-Fi home screen and after sign in screen prototypes.

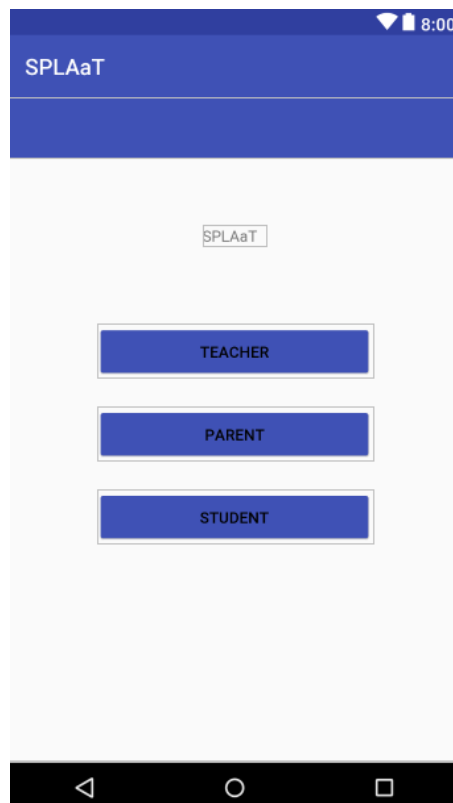


Figure 13: Hi-Fi home screen Android Studio prototype.

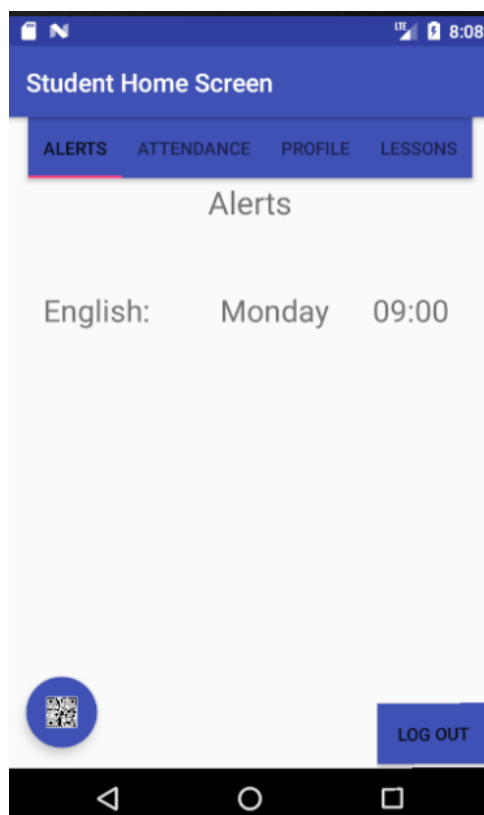


Figure 14: Student fragmented home page, alerts fragment.

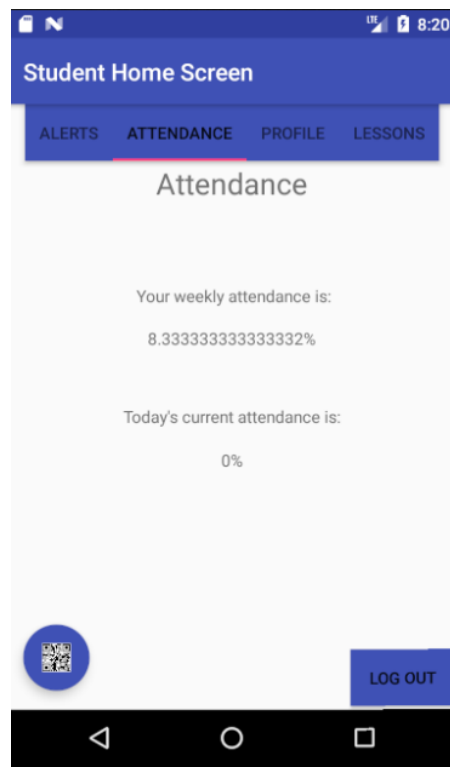


Figure 15: Student fragmented home page, attendance fragment.

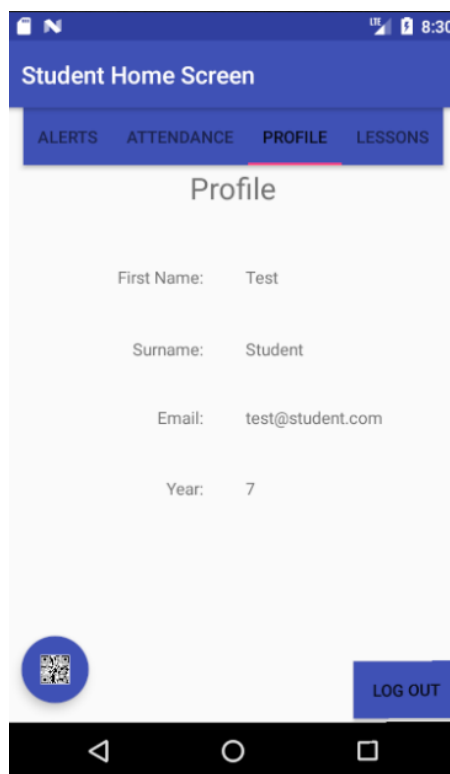


Figure 16: Student fragmented home page, profile fragment.

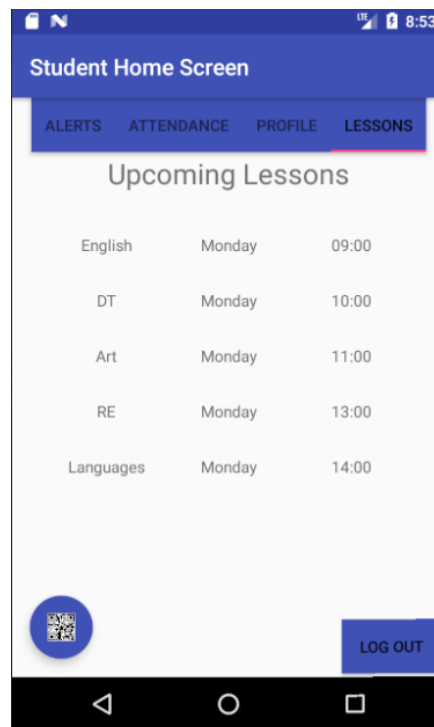


Figure 17: Student fragmented home page, lessons fragment.

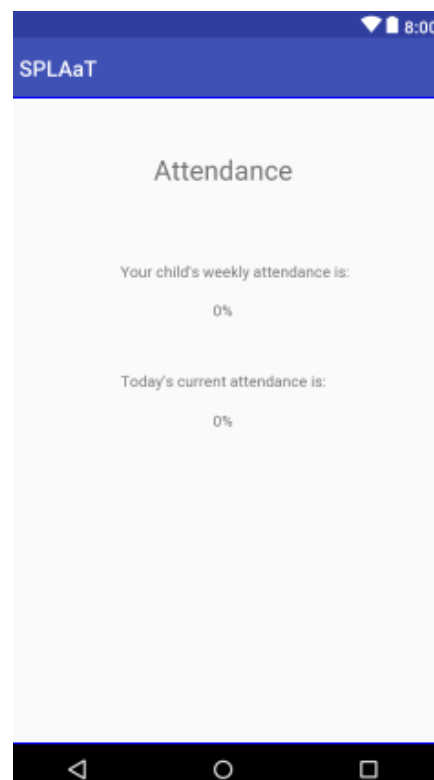


Figure 18: Parent home page, child attendance fragment.

Appendix B: Application backend

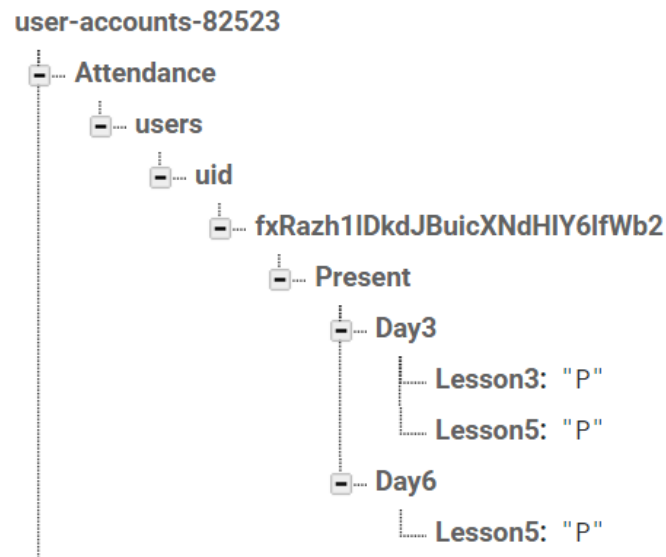


Figure 19: Google Firebase student attendance database.

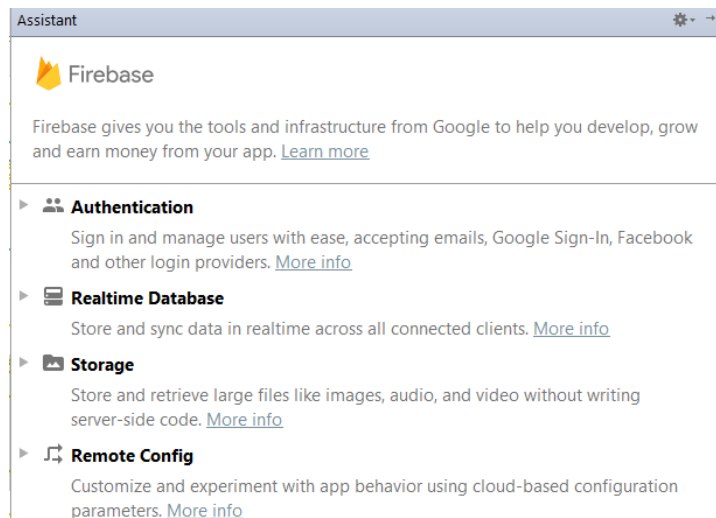


Figure 20: Android Studio, Firebase implementation tool.

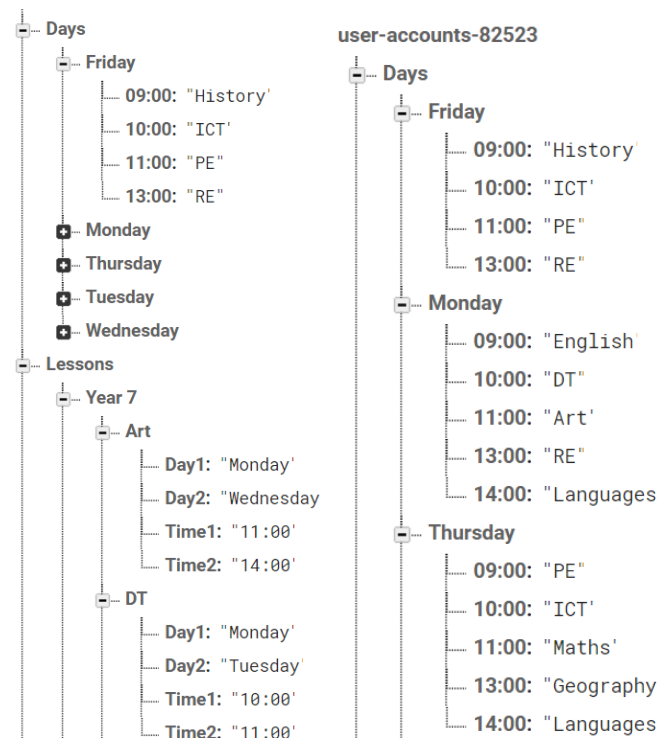


Figure 21: Denormalizing database data for easy retrieval.

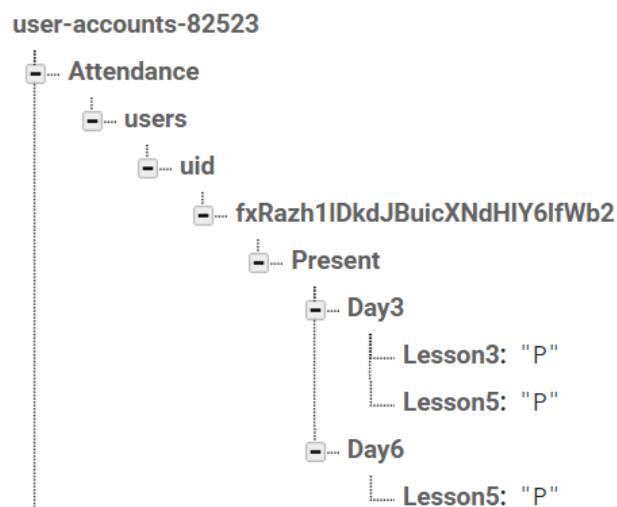


Figure 22: Updated attendance data for a student for days 3 and 6.

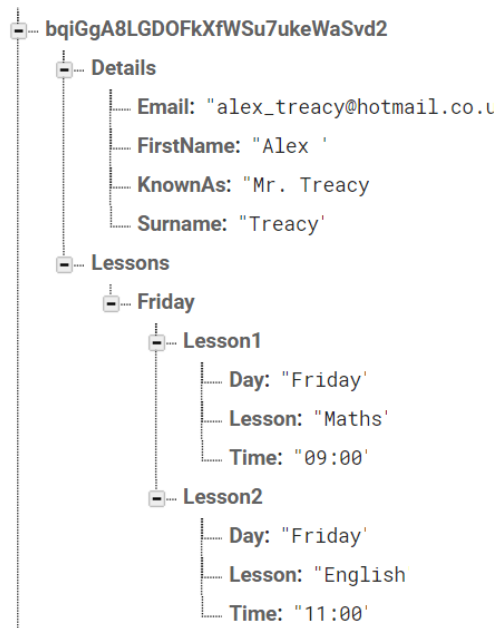


Figure 23: Updated database of lessons added upon entering details.

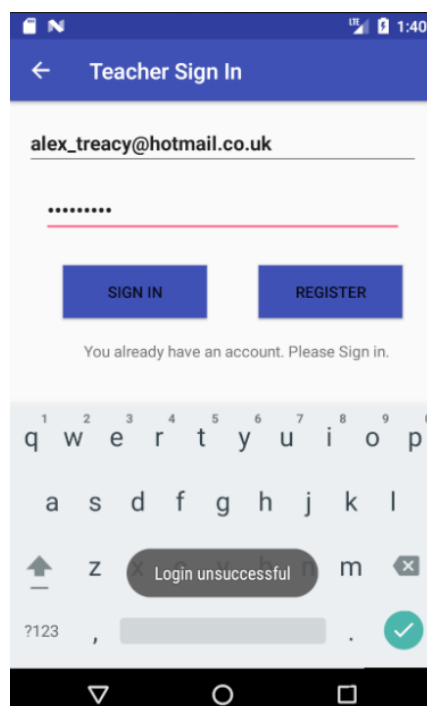


Figure 24: Login unsuccessful message upon entering incorrect password.

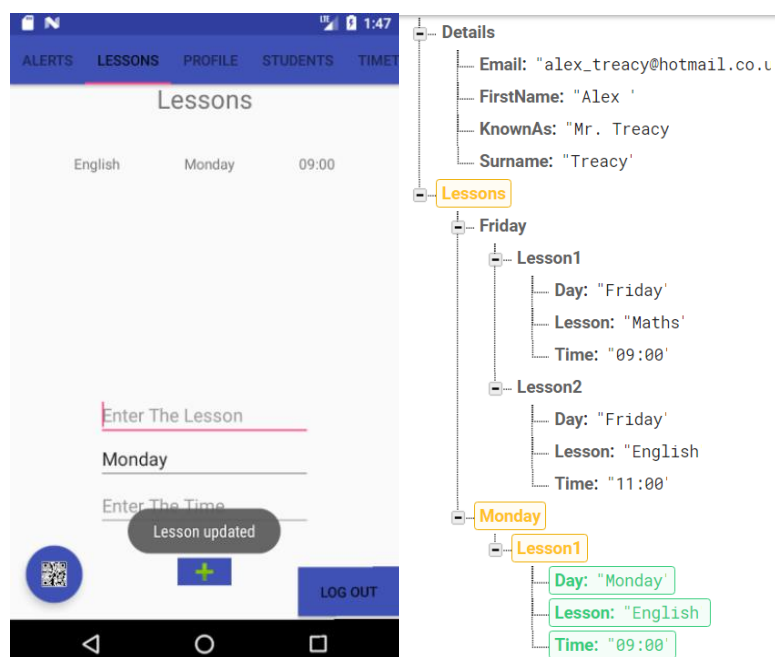


Figure 25: Updating lessons for the current day.

Appendix C: User questionnaire and survey results

Which of these tracking choices would you use if incorporated into a mobile application?

Answered: 11 Skipped: 0

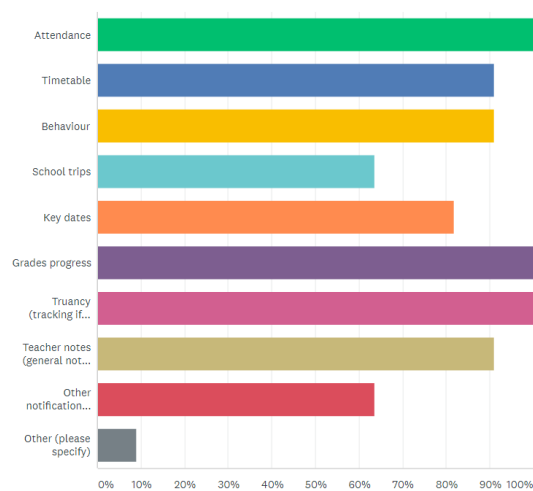


Figure 26: Bar chart of application features to be implemented.

Have you, or will you ever use a mobile application to track your child's education?

Answered: 11 Skipped: 0

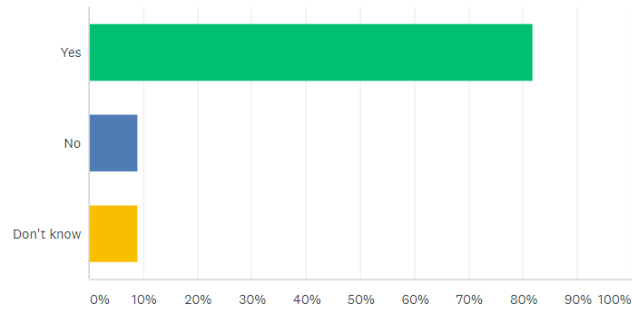


Figure 27: Bar chart of application potential users.

Appendix D: Additional relevant items

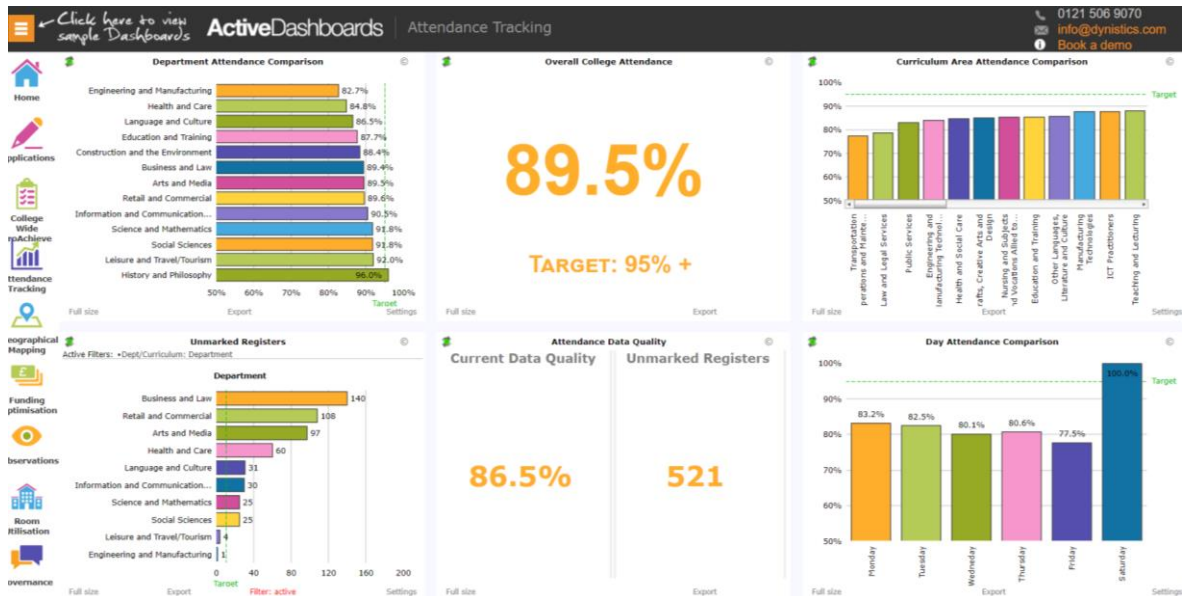


Figure 28: Example dashboard.

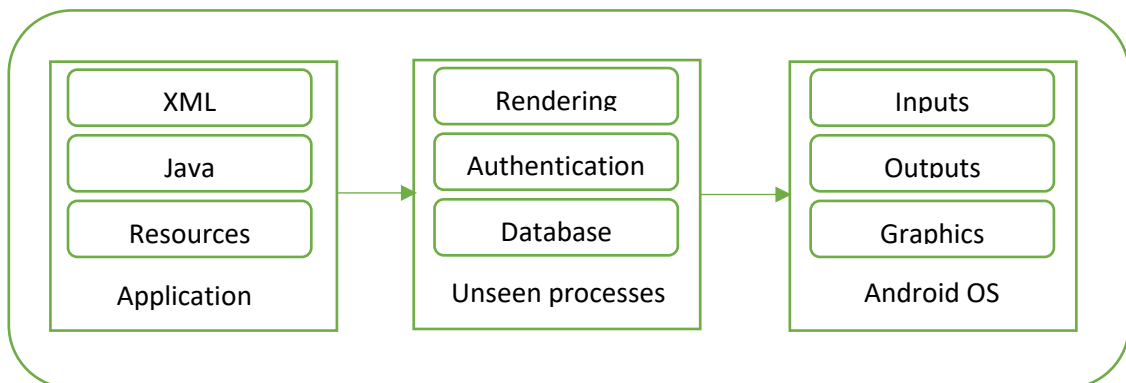


Figure 29: fundamental backbone of the application to the user interface on the operating system.



Figure 30: SPLAaT Android image icon for launching application.

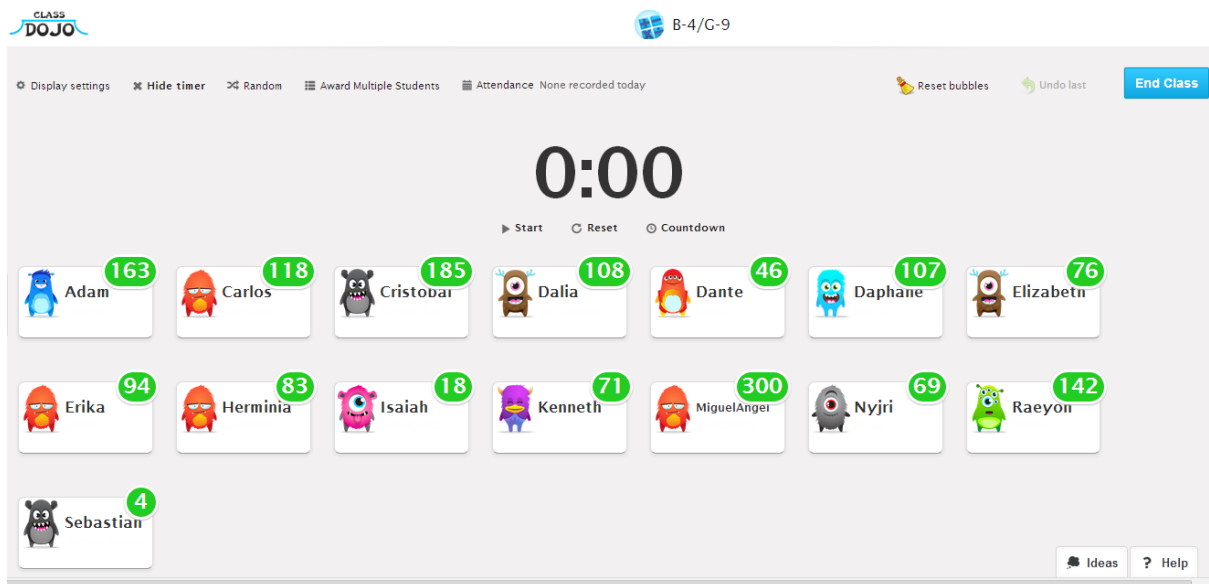


Figure 31: ClassDojo, teachers view of student in current lesson.

```
final DatabaseReference lessonRef = mFirebaseDatabase.getReference().child("Teacher");
lessonRef.keepSynced(true);
lessonRef.addChildEventListener(new ChildEventListener() {
    @Override

    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        String value = dataSnapshot.getValue(String.class);

        tvL1 = (TextView) view.findViewById(R.id.tvL1);
        tvD1 = (TextView) view.findViewById(R.id.tvD1);
        tvT1 = (TextView) view.findViewById(R.id.tvT1);

        if (value.contains(":")) {
            tvT1.setText(value);
        }
    }
});
```

Figure 32: Code snippet of catching the time value from the “value” variable.

```

String userID = user.getUserId();
String studPresent = "P ";
System.out.println(currentHour);
String fullQRPresent = studPresent.concat(userID).concat("/").concat(Integer.toString(lesson).concat(": " + day));

back = (Button) findViewById(R.id.back);
qrCode = (ImageView) findViewById(R.id.qrCode);
Bitmap bitmap1 = getIntent().getParcelableExtra( name: "QR code");
qrCode.setImageBitmap(bitmap1);

try {
    BitMatrix bitMatrix = multiFormatWriter.encode(fullQRPresent, BarcodeFormat.QR_CODE, width: 2500, height: 2500);
    BarcodeEncoder barcodeEncoder = new BarcodeEncoder();
    Bitmap bitmap = barcodeEncoder.createBitmap(bitMatrix);
    qrCode.setImageBitmap(bitmap);
} catch (WriterException e) {
    e.printStackTrace();
}

```

Figure 33: Code snippet of concatenating user ID with the lesson and day and generating QR code.



Figure 34: XML code with constrained view, using dp rather than px.

Copyright waiver



UCEEL Copyright Waiver

Student Name: Alexander Treacy

Project/Thesis Title Student Attendance, Behaviour and Progress: A Mobile
Monitor

Course: BSc (Hons) Computer Science

Student Agreement

1. I confirm that Birmingham City University can electronically archive and make accessible the project / thesis described above via the UCEEL Electronic Library system. I retain all other ownership rights to the copyright of the document / project work described above.
2. I confirm the above project / thesis is a true and unaltered representation of the project / thesis as submitted to Birmingham City University course tutors and examiners.
3. I confirm that the above project / thesis **includes** material copied from a source (e.g. a book) where ownership of the copyright does not belong to myself.

If the **project / thesis includes** such material please supply the following details:

a) **Page reference / item reference:**

b) **I have** obtained and attached a written permission statement from the owner(s) of each third party copyrighted matter included in my project / thesis

☒ **Yes** ☐ **No** (please circle)

*(If **No**, I understand the electronic copy of my project / thesis available on UCEEL will omit these sections from view)*

Signature: *A. Treacy*

Print Name: ALEXANDER TREACY

Date: 15/05/2018

N.B. If you are at anytime in consultation with a publisher regarding this work you will need to declare the copy held on UCEEL. Some publishers may regard the UCEEL copy as constituting prior publication. The copy can be removed from UCEEL if it becomes an obstacle to future commercial publication.

Official Use only

ORION unique number: _____

Date added to the system: _____

IS228a/Oct07