

|  |
| --- |
|  |
| Bluetooth Security Solution |
| Eye of the Tiger |
| Philip Leung (211908530)  Dan To (206593727)  Simon Ma (212355459)  Jason Kuffour (212172722)  Aloka Kumaranayake (211833647)  4-10-2016 |

Table of Contents

[1.0 Introduction 5](#_Toc448108935)

[1.1 Purpose and Scope 5](#_Toc448108936)

[1.2 Bluetooth Security Solution 5](#_Toc448108937)

[2.0 Marketing 7](#_Toc448108938)

[2.1 Target clients 7](#_Toc448108939)

[2.2 Criticism and Concerns 8](#_Toc448108940)

[2.3 Benefits 8](#_Toc448108941)

[2.4 Product Cost 9](#_Toc448108942)

[3.0 Requirements 10](#_Toc448108943)

[3.1 Administrator Software 10](#_Toc448108944)

[3.2 Android Application 11](#_Toc448108945)

[3.3 Database 11](#_Toc448108946)

[3.4 Bluetooth Scanner Firmware 12](#_Toc448108947)

[3.5 Bluetooth Tag & Scanner Hardware 13](#_Toc448108948)

[4.0 Technical Specifications 14](#_Toc448108949)

[4.1 Administrator Software 14](#_Toc448108950)

[4.1.1 The Model 15](#_Toc448108951)

[4.1.2 The View (User Interface) 16](#_Toc448108952)

[4.1.3 The Controller Component 23](#_Toc448108953)

[4.2 Android Application 24](#_Toc448108954)

[4.3 Database 28](#_Toc448108955)

[4.3.1 Configurations Database 28](#_Toc448108956)

[4.3.2 Dynamic Student Information Database 29](#_Toc448108957)

[4.3.3 Static Student Information Database 29](#_Toc448108958)

[4.3.4 Administrator Information Database 30](#_Toc448108959)

[4.3.5 Class Information Database 31](#_Toc448108960)

[4.4 Bluetooth Scanner Firmware 31](#_Toc448108961)

[4.4.1 First Stage (Python Script) 32](#_Toc448108962)

[4.4.2 Second Stage (Perl Script) 33](#_Toc448108963)

[4.4.3 Third Stage (Java Program) 33](#_Toc448108964)

[4.5 Bluetooth Tag & Scanner Hardware 34](#_Toc448108965)

[5.0 Engineering Design 35](#_Toc448108966)

[5.1 Administrator Software 35](#_Toc448108967)

[5.2 Android Application 35](#_Toc448108968)

[5.3 Database 36](#_Toc448108969)

[5.4 Bluetooth Scanner Firmware 37](#_Toc448108970)

[5.5 Bluetooth over Radio Frequency Identification 37](#_Toc448108971)

[5.5 Bluetooth Tag & Scanner Hardware 38](#_Toc448108972)

[5.7 Bluetooth Scanner Casing 39](#_Toc448108973)

[6.0 Health and Safety 40](#_Toc448108974)

[6.1 Health And Safety Measures Taken By The Team 40](#_Toc448108975)

[6.2 Bluetooth Health Concerns 41](#_Toc448108976)

[6.3 Bluetooth Tag Power Emissions 41](#_Toc448108977)

[7.0 Future Improvements 41](#_Toc448108978)

[7.0.1 Integration with school’s call home system 41](#_Toc448108979)

[7.0.2 Integration with magnetic locks 42](#_Toc448108980)

[7.0.3 Creating an interactive learning experience 42](#_Toc448108981)

[8.0 References 42](#_Toc448108982)

[9.0 Appendix 43](#_Toc448108983)

[9.1 Appendix A (UML Diagrams) 43](#_Toc448108984)

# 1.0 Introduction

## 1.1 Purpose and Scope

In today’s public school systems, the Ontario student attendance is still recorded and filed by pen and paper. This can often be timely and grueling task for teachers and administrative staff. Calculating total absences and lates required for report cards relies on the manual tallying of student attendance records over the course of the term by the teacher. This is a monotonous and time consuming task that can be easily digitized to increase the speed and efficiency of taking, recording and analyzing attendance records.

Dan To

Jason Kuffour

Philip Leung

Simon Ma

Aloka Kumaranayake

Our Team

## 1.2 Bluetooth Security Solution

Our solution involves the development of a Bluetooth receiver proximity scanner and an identification tag, which is similar to the standard ID cards issued at Ontario public schools. The scanners would be placed in all active classrooms and would actively pair with student tags that are expected to be in that class based on their timetables. When a student’s tag pairs with a classroom scanner, the microcontroller on the scanner will update the student’s record with his/her entry time, status and location. The student’s total absences and lates are immediately calculated and recorded. This attendance record is stored and can be retrieved at any time using our software application.

The software will be an Android specific application for cell phones and tablets available to teachers and administrators. This gives them easily accessible data of current student statuses for their current classroom location. The application can also receive input, so that teachers can update records in the cases where a student forgets or loses their tag and cannot interact with the scanner. This solution possesses a security concern in the sense that students could carry the tag of other students and log attendance records for a student who is not physically present. To address this security concern, the scanner is designed to compare student entry times. Most standard classroom doorways permit the entrance of one student at a time. If a student were to follow directly behind another student, there would be a gap of least 1 second between the recorded times. If a gap less than this threshold is detected, a flag will be raised in the software application. The teacher will be notified and advised to verify the presence of the students in question.



# 2.0 Marketing

## 2.1 Target clients

Our current targeted client is the Canadian public and private school systems. The total number of students enrolled in public elementary and secondary schools in Canada in 2011/2012 is about 5,032,183 [1].

According to the studies done by statistics Canada in the years 2004 to 2005 there were about 15, 500 schools in Canada. The distribution of schools are shown below.

* 10,100 elementary
* 3,400 secondary
* 2,000 &amp; mixed elementary and secondary
* The overall average is 350 students per school

This study has also shown that there were 310, 000 educated employed by the elementary and secondary school system within this time range [2].

By looking at these numbers, we can see the target market for our current product design is very reasonable. Our solution is not limited to just the school setting. Any type of building, company or event that requires attendance tracking could benefit from our product, as long as it does not involve high security. It could be used to track things like employee attendance at company events, membership tracking at gyms or community centers such events could benefit from having individual attendance tracked for data analytics.

Although the solution alone carries potential security issues, if it is implemented with existing security measures this system could provide an abundance of data that could help to improve safety and security measures. For example, a typical security check involves verifying an individual’s photograph on an ID tag matches their appearance by some security official. If the ID also contained our Bluetooth solution their time of entry and location of entry would also be recorded. This data could provide a lot of conclusive data if a security issue arises that involves the individual.

The Android Operating System was selected among other operating systems, due to having a large market share. Based on information from the website Net Market Share, the Android Operating System has about 60% of the market share in Feb 2016 [6]. Having a large number of users with the same Operating System will ensure that many administrators and teachers at school will be able to use the Android application for the Security Solution.

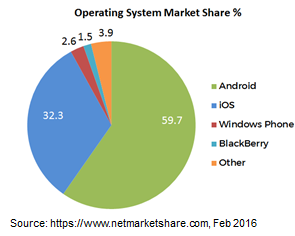


Figure 1. Mobile Operating System Market Share

## 2.2 Criticism and Concerns

The Eye of the Tiger team has identified the school board and the teachers to be the primary users of the system. A survey has been conducted to collect the opinions of the teachers regarding the system. It seemed that the teachers were concerned about enforcing the students to carry their student identity card. However the teachers have expressed positive opinion on how they can use the time saved for helping students as well as collecting resources for improvement of the classroom experience. Also the teachers were keen on the idea of adding interactive capabilities to the student ID to be used as a clicker device. They have agreed that the administration could benefit more than the teachers from this system.

Some of the comments from the survey conducted among elementary and high school teachers:

* *“The real advantage would be having the attendance sent to the office without a student having to take it.”* – ESL Teacher
* *“I would be able to jump straight into my lesson and give a few more minutes to offer my assistance for others who might need instructions.”* – Public School Teacher

The team has contacted Toronto School Board and their response regarding the tracking was not positive due to political and social reasons. They were concerned that the parents will not be happy about tracking their children. However, based on further feedback, the idea of using the system as a security component seemed attractive. Therefore, the team has redirected towards implementing the system to be a security solution. Parents will need to sign a consent form to allow the student to participate in the Security Solution.

## 2.3 Benefits

By implementing the security component of the tracking system, the existing system can be improved. The safety and the security improvements can be beneficial to the school, students, as well as the parents. The system will allow the school to be aware of the student’s location for emergency situations, such as fires or missing students, as well as a system for attendance purposes. These new improvements for schools can be considered highly beneficial in the present society, where the safety of students are a major concern.

In addition, the data collected through the system can be analyzed to identify student behaviour, such as leaving classrooms for long periods of time. Information obtained by the Security Solution will be contained in the school. Student locations for entry and exit times, and location information is only available to school administrators. Student status and profiles will only be accessed by teachers. Current scantron systems display all student information on the sheet. When students take the attendance to the administration office, there is a privacy concern. Since the Security Solution will limit the information to teachers and administrators, there is an overall improved privacy and security.

## 2.4 Product Cost

Based on the target market we have, we are able to reduce the price per units. Following is a cost analysis done excluding the labor costs. The cost is shown as two separate tables as the tag used by the student and the Scanners that will be installed in the classrooms can be considered as two separate

Our standard tags will require PCBs of approximately 1 inch by 2 inches requiring only a battery slot and a Bluetooth chip slot. Research into small scale PCB prototype fabrication services price each chip at approximately $3[3]. An online calculator that provides PCB fabrication comparison prices show that a barebone 2 square inch PCB between $2-33 based on the manufacturer and additional add-ons[4]. The simplicity of our PCB and small number of components means we can avoid the additional costs of PCB add-ons like solder masks and silk screens. Upon further research, instead of fabricating a custom PCB, Bluetooth beacons can be purchased for a cheap price of $1.50 per board. As such, this will allow us to bypass the testing of the tags and save us costs.

For the Bluetooth scanner, we chose to use a Raspberry Pi. Typically, the Raspberry Pi runs for $40-$60 depending on the model. For our prototype, we used a Raspberry Pi Model B, which is in the $60 range. However, any model of the Raspberry Pi will work with our Bluetooth Scanner Firmware.

The product cost scheme will be as follows: 100% markup on the Bluetooth Tags and 25% markup on the Bluetooth Scanners. This decision was made based on the fact that the Bluetooth Scanners will not be needed to replace often, whereas the Bluetooth tags will be replaced when new students are enrolled in the school or when a student loses his or her student card. As such, table 1 shows the product cost based on our prototype.

Table 1: Product cost based on current prototype.

|  |  |  |  |
| --- | --- | --- | --- |
| **Bluetooth Scanner** | | **Bluetooth Tag** | |
| Cost to produce | Selling Price | Cost to Produce | Selling Price |
| $60 | $75 | $1.50 | $3.00 |

# 3.0 Requirements

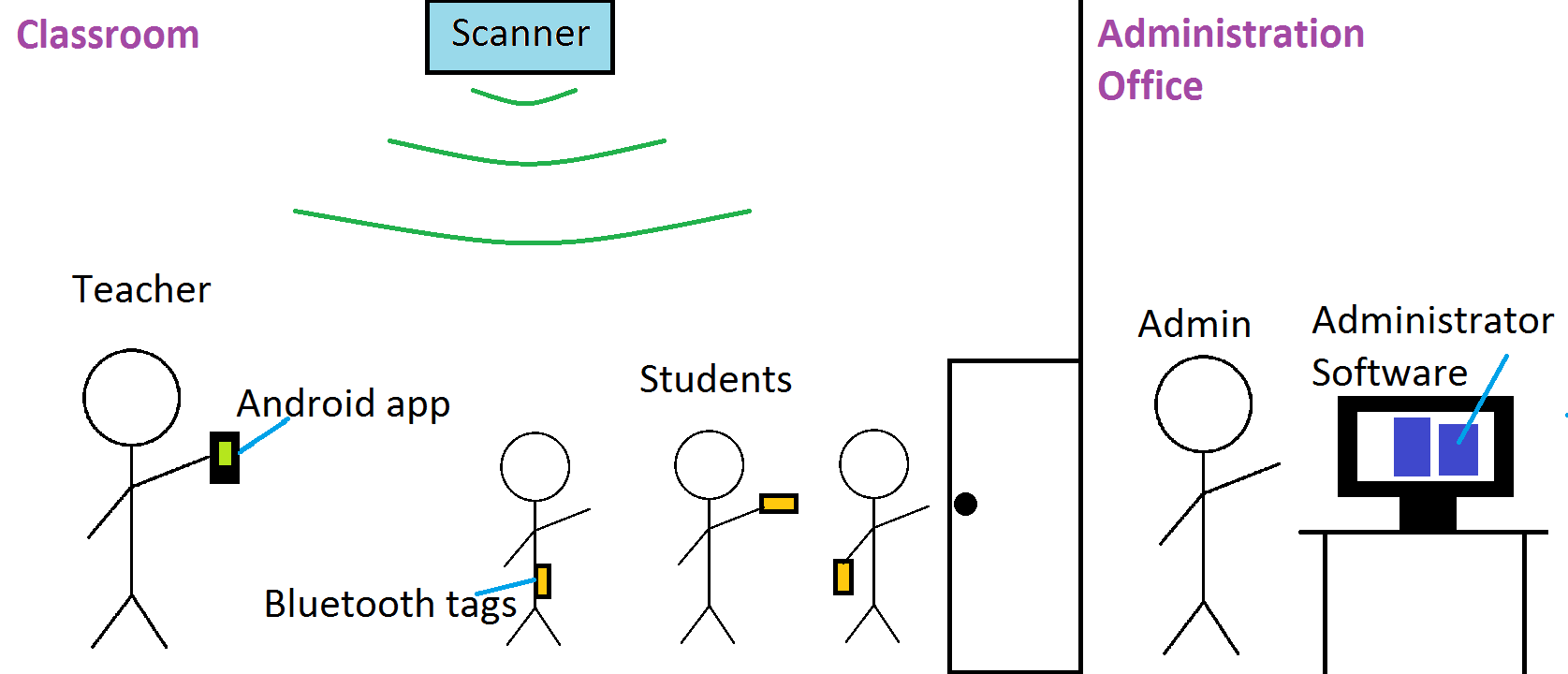


Figure. 2 Security System overview

## 

## 3.1 Administrator Software

The administrator will mainly be used by the school’s administrative staff to view or delete student/administrator records, add new class/event information, specify school start and end times, as well as specify information related to period start times, end times and duration. The administrator Software is the school’s main access point for configuring the attendance tracking abilities of the Scanners, and as such it has the following main requirements:

1. It must be accessible to only qualified administrative staff with granted permissions.
2. It must be able to update the school’s Scanner system which consists of the school’s start time, end time, periods’ start times, and lunch start and end times.
3. It must be able to allow administrative staff to add, view or delete student/administrative information tracked by the Scanners
4. It must allow administrative staff to view all course information as well as their start and end times.
5. It must allow administrative staff to search student, administrative or class information either by using their tag ID (student or administrator only) or unique ID
6. It must be fast to view attendance updates in near real time. Hence it must provide updates in no more than 10 seconds
7. It must be robust to be able detect invalid inputs during school time configuration and respond accordingly.
8. It must allow users to manually check for any updates in attendance information.
9. It must be user friendly and easy to use, as well as provide users with easy to understand attendance information

## 3.2 Android Application

The android software application will allow administrators and teachers to view student record data, as well as retrieve calculated and logged data of all students. The application will also be able to receive input from the user and update the database in real time, in case the student is unable to communicate with the system or the user detects some other anomaly. The application will consist of three screens accessible from swiping on the screen, clicking links or from the navigation menu. When the application is opened, the user will be prompted to log in. This is to ensure that only someone with appropriate credentials is accessing the information. The primary screen will contain live student data specific to the current user’s current location, for instance, the total students expected in classroom, current absences. The secondary screen will present more student specific details. On this screen, all of the student’s names and status expected in the current classroom, will be displayed. Selecting a listed student’s name will bring the user to a tertiary screen with the student’s complete information data, such as total number of lates, timetable, current status, last location etc. The software will be written using Android’s Java IDE and will target android devices. If time permits, an iOS version may also be implemented.

## 3.3 Database

IBM’s cloudant.com cloud no-SQL service was chosen as the storage database for our solution. It also possesses a Java and android API making it incredibly suited to our requirements. It provided all the necessary tools and resource to access our data stored on their service through http requests or the java programming language

The choice of using IBM’s cloudant database was also motivated by the fact that it satisfied all of the requirements that were identified for the project. Below are the main requirements that were required for the database, which were all satisfied by Cloudant.

1. Availability: The attendance information we need to store must always be made available to the school and our Bluetooth security system so as to provide optimal performance and accurate data. By using cloudant, we have ensured that the data in the database will always be made available to the school and the system since multiple redundancies have been put in place by IBM to ensure their services are always made available to their clients.
2. Cross-platform usage: The data stored on the database must be easily accessible through any program, app or application independent of the implementation used in developing the program. IBM’s cloudant easily met this requirement due to the fact that it provided libraries for a number of programming languages to access their services as well as making it available through the RESTful api which means any application can access the data through http requests.
3. Data organization: The database must provide a flexible data organization scheme that does not limit how the data is stored or organized within it. Since Cloudant is a No-SQL database, the data stored within it is not confined to specific schemas or values. This made it possible for us to easily define how we wanted to store the attendance information as well as define what fields to search by (explained in section 4)

## 3.4 Bluetooth Scanner Firmware

The Scanner Firmware is the software that is running on the Bluetooth Scanners and is responsible for detecting student/administrator tags and updating their attendance information accordingly by considering their timetable and the school’s time configuration settings. It uses most of the information entered into the administrator software and as such must be in sync with it (accomplished through the Cloudant database). The Bluetooth Scanner Firmware has the following requirements which it must meet for maximum functionality:

1. It must be able to detect all school registered student tag IDs.
2. It must be able to determine whether a detected tag is leaving or entering its scanning area to update the enter/exit times accurately.
3. It must ensure that the student’s attendance information is updated only once when they enter and once again when they exit whenever their tag is detected.
4. It must be able to detect tag IDs every 1 second to ensure that the attendance tracking information is as accurate as possible.
5. It must be able accurately update a student’s attendance information using the time the tag was scanned, student timetable and the school’s time configuration.
6. It must be able scan, detect and update multiple (more than 30) students’ information concurrently without any delays, errors or missing data in the information.
7. It must be able to update the student’s attendance information in the Cloudant in no more than 10 seconds i.e. the new student attendance information should be made available for the administrator software and app within 10 seconds of the tag being scanned by the firmware.

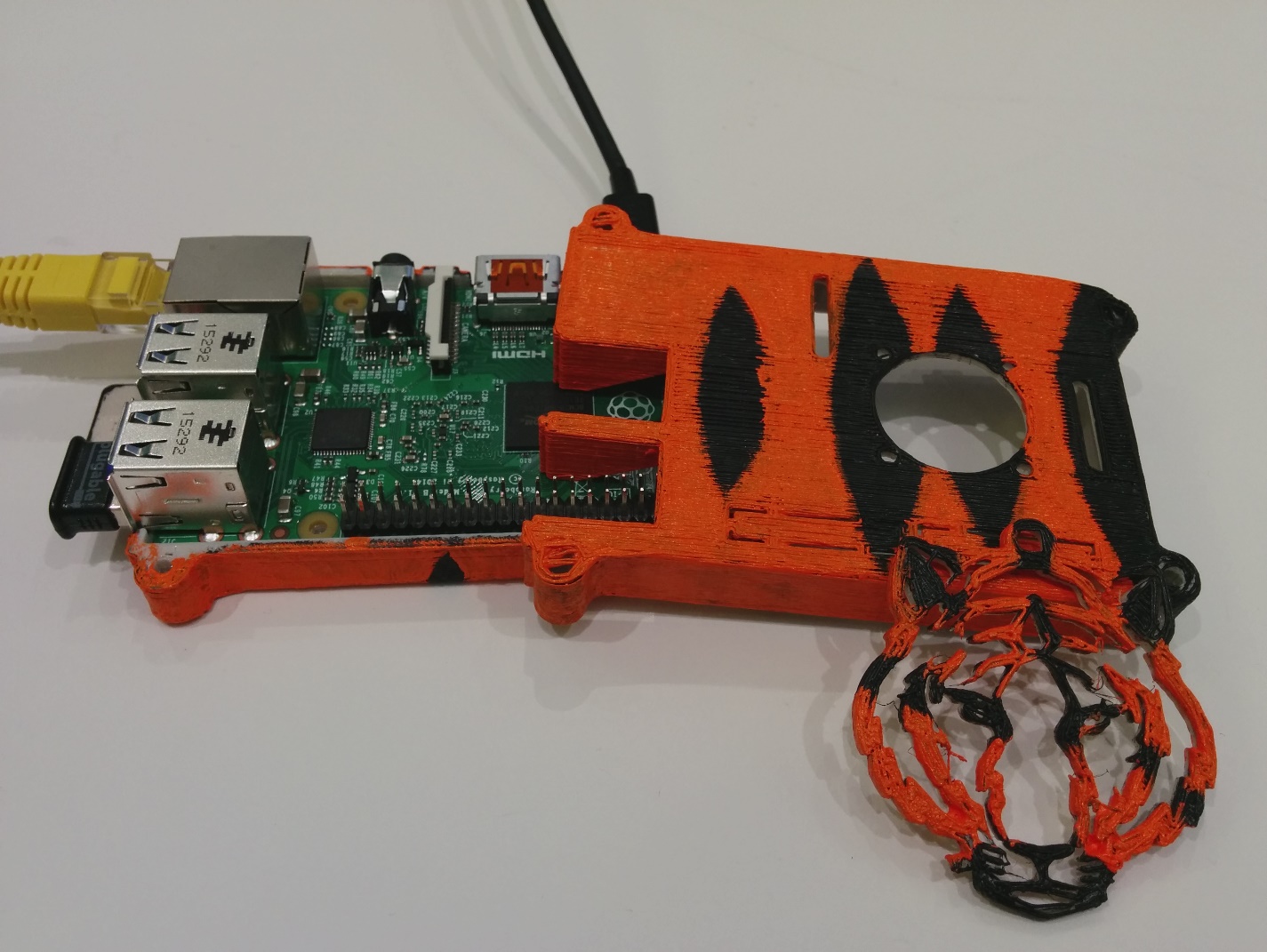


Figure 3: Bluetooth Scanner

## 3.5 Bluetooth Tag & Scanner Hardware

The Bluetooth tag will be carried by students. Once the student enters a classroom, the scanner will detect it and update the database accordingly. As such, it has the following requirements:

1. Small form factor
2. Low energy consumption
3. Low cost

The Bluetooth scanner will be placed within each room. It will be updating the database according to the tags it detects. It has the following requirements:

1. Able to pick up multiple tags concurrently
2. Able to update the database

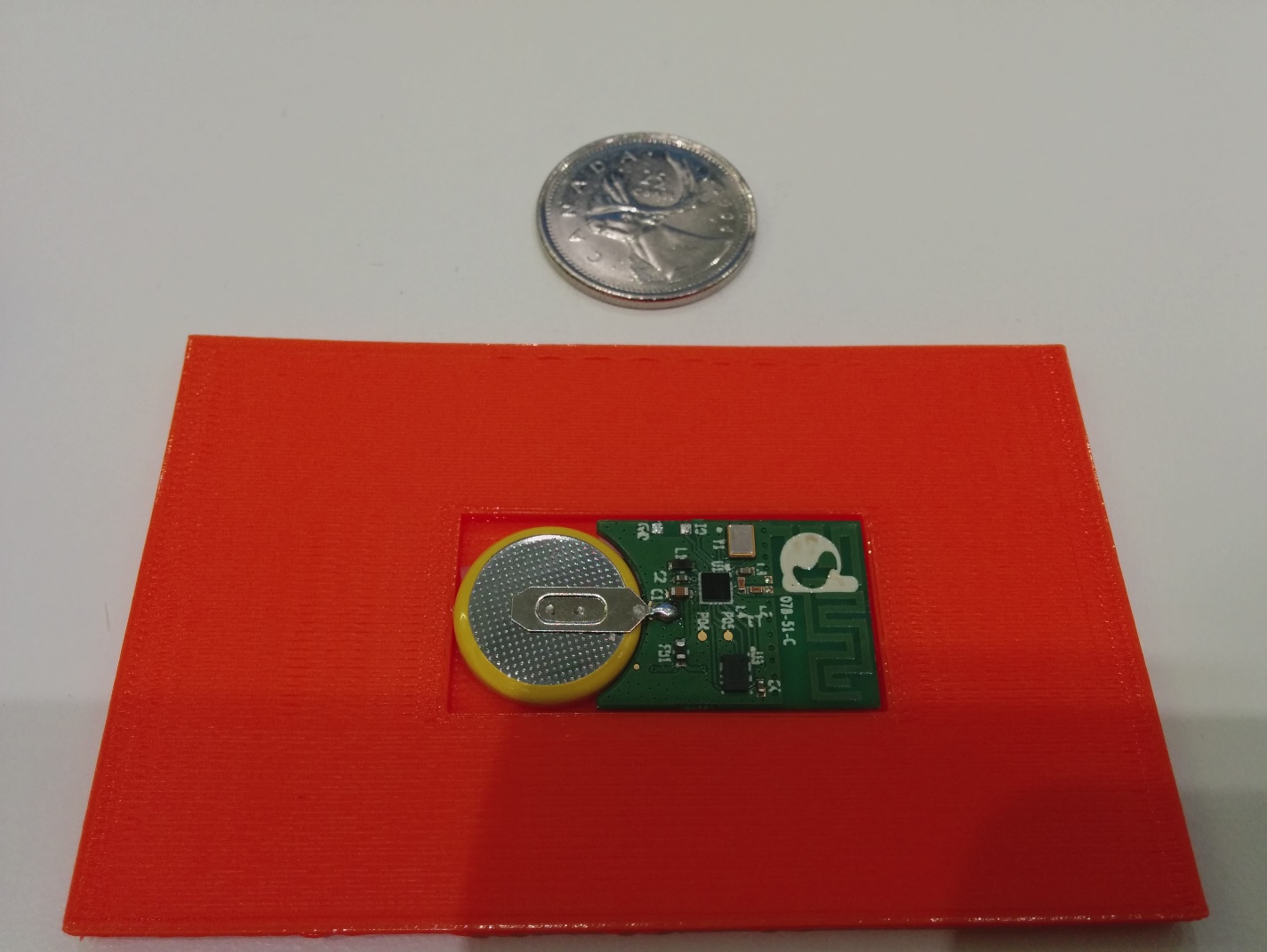


Figure 4: Tag ID

# 4.0 Technical Specifications

## 4.1 Administrator Software

The administrator Software, which is a Desktop Application used by administrative staff, was all designed and implemented using the Java programming. The decision to use Java was based solely on the fact that IBM’s Cloudant provides ready to use libraries and resources to directly access their databases using Java. This made using Java the most viable choice of implementation language as opposed to other languages such as C++.

The administrator software was designed using the Model-View-Controller software architectural pattern. In that sense, the administrator software has three main components which handle different aspects and functionalities of it: the model is responsible for obtaining and updating data from the Cloudant database, the view presents/accepts all the necessary data to the user and the controller is responsible for sending this data to and from both the view and controller.

Each of the three components listed above address a few of the requirements listed in section 3.1, but never the less all three as a whole satisfy all 9 of the requirements. Further discussion of the three components and how they were implemented follows.

### 4.1.1 The Model

The model component’s main functionality is obtaining all the information from the Cloudant database and wrapping it in an easy to use interface (Java class) for easy use by the controller. It is also responsible for performing updates such as adding a new student, to the database.

The model for the administrator software consists of five Java classes and one main Java class that acts as the main access point to using the other five classes for the Controller component. The main class (Model) combines the functionality of all five classes into one easy to use interface and makes all the functionality, required by the controller, easy to access with a few method calls. The UML diagram for all six Java classes in the model component of the administrator software is displayed in the Appendix A. A simplified UML diagram of the classes and their interaction is provided in figure 5.

Four of the five java classes handle a different aspect of the cloudant database, i.e the students (Students class), administrators (Admins class), courses (Courses class) and the time configuration of the school (Config class). The functions of these four classes also include communicating with the cloudant database directly using the client libraries they provided. These four classes provide methods that allow for easy creation, deletion, edit or obtaining of important database information related to the students, administrators and school. In addition to this, they also provide methods for validating the creation or deletion of new students, courses or admins. One very important method provided by all four methods is the getData method which obtains all the relevant information related to that class from the database. The fifth class (JSONhandler) acts as sort of like of translator to convert the data, being sent to or from the database. This class was specifically designed for this purpose since it’s a functionality needed by all the other four classes and is also used in the Scanner firmware (discussed in a later section).

The Model class also performs a bit of multithreading to concurrently get all the database information through the four classes. Whenever the Model class is initialized, it creates four threads which simultaneously pull data from the database by invoking the getData functions in the four classes. This greatly improves the response time of the software during initial start-up and manual refresh of the displayed data since four database transactions are being made at the cost of one. All the methods in the Model class consist mostly of the methods present in the Admins, Config, Students and Courses classes. The methods in these four classes are built upon in the model class to provide further abstraction and to allow easier access to the data obtained from the database. For example, the method retrieveStudents in model combines the method getDynamicInfo and getStaticInfo from the Students class to combine all the student information that will be displayed by the view in one data structure as opposed to two separate structures.

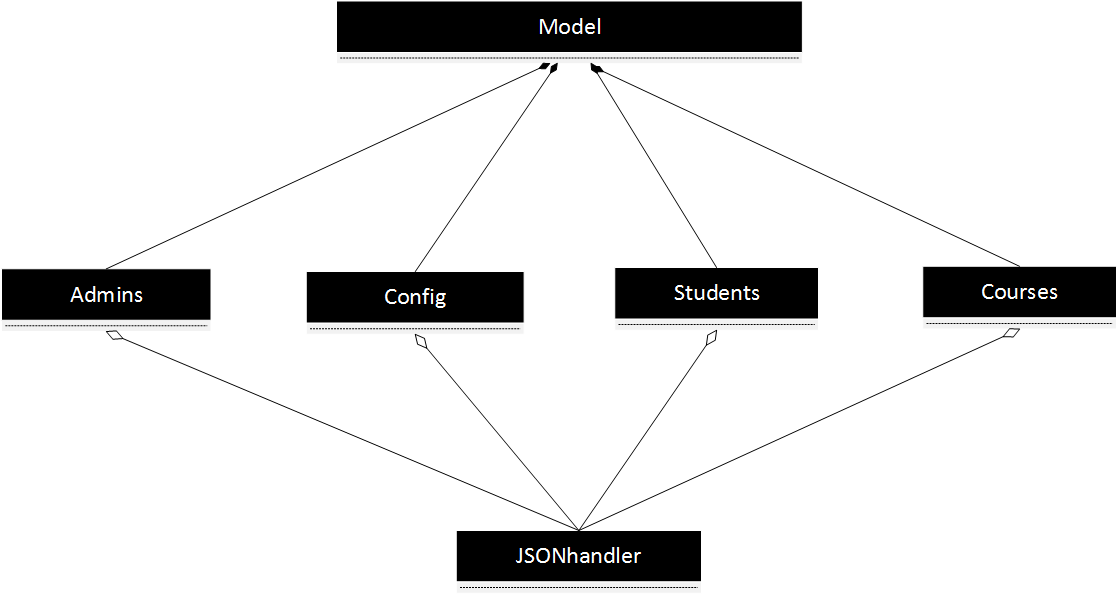


Figure 5: Diagram overview for the Model Component and its classes. Refer to the Appendix for a detailed UML description of each class

### 4.1.2 The View (User Interface)

The main purpose of the View component is to visually present all the attendance tracking information obtained by the Scanner Firmware to the user. The information displayed by the View component is obtained from the Model component through the Controller (discussed in the next section). There were two iterations of the View designed and implemented with the last iteration being the final version. Both views were designed using two different Java frameworks and completely different visual designs.

#### 4.1.2.1 The Prototype View (User Interface)

The first View designed for the administrator only met a few of the requirements listed in section 3.1. It was implemented using the Java Swing UI framework and designed using the Eclipse Window GUI Builder. This allowed rapid development of the UI.

The View only displayed a student’s first name, last name and timetable. It did not show their student number or their attendance information. Student, administrators and class records were displayed in tables in a tabbed pane view, one for each category. Each tab contained two sub panes called view and add. The view tabs display records while the add tab allows the addition of new records.

Adding a new student/administrator only involved entering their first and last names, and entering their timetables separated by the ‘/’ character. While adding a new course involved the class name, start time, end time and the location. It did not allow the user to specify which period it is in nor did it allow the times to be changed or dynamically configured. It did not allow the users to search the records by their unique IDs nor did it allow them to manually refresh the displayed information. The only validation either of them performed was checking if the text fields were non-empty.

The only requirement that this UI fully satisfied is “It must allow administrative staff to view all course information as well as their start and end times”. The prototype view was a good stepping stone for designing and implementing the final UI which meets all the requirements listed in section 3.1. Figures 6 – 7 display screenshots of the prototype’s student and course views. The administrator view is very similar to the student’s view. The prototype view only has one Java class that instantiated and created all the UI elements for it.

Figure 6: Prototype UI For viewing (left) and adding (right) students in the database

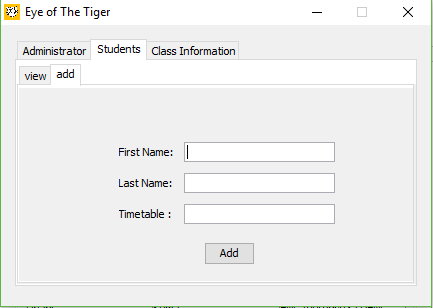
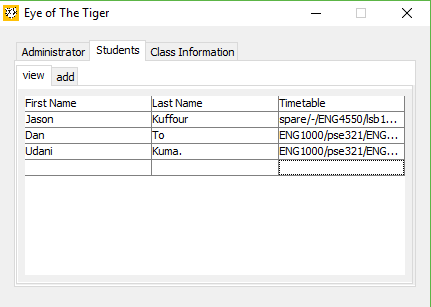
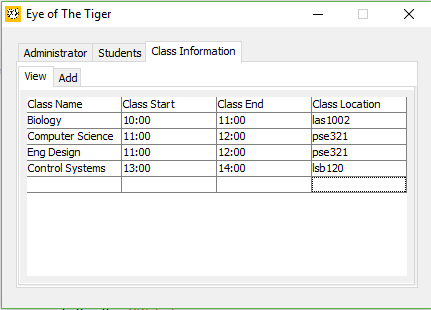
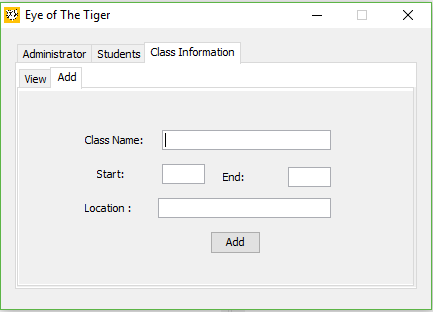


Figure 7: Prototype UI for viewing (left) and adding (right) courses to the database



#### 4.1.2.1 The Final View (User Interface)

The final UI for the administrator software was designed and implemented using a completely different UI framework from the prototype view. It was designed using the QtJambi framework which is based on the C++ Qt GUI framework. The framework allowed the use of a UI designer to design the UI and the use of Qt stylesheets (css) to apply different styles, shapes, transforms etc to the each UI component. These features allowed the final view to be designed and implemented much better than the first view. The final view meets all the requirements listed above and provides a more detailed view of student attendance information.

The final UI includes a login screen which limits the access of the administrator software to select individuals with the right login credentials. The login page includes an input validation and displays a message whenever the user enters an invalid login credential. This ensures that the first requirement is satisfied. The login page is displayed in figure 8 below.

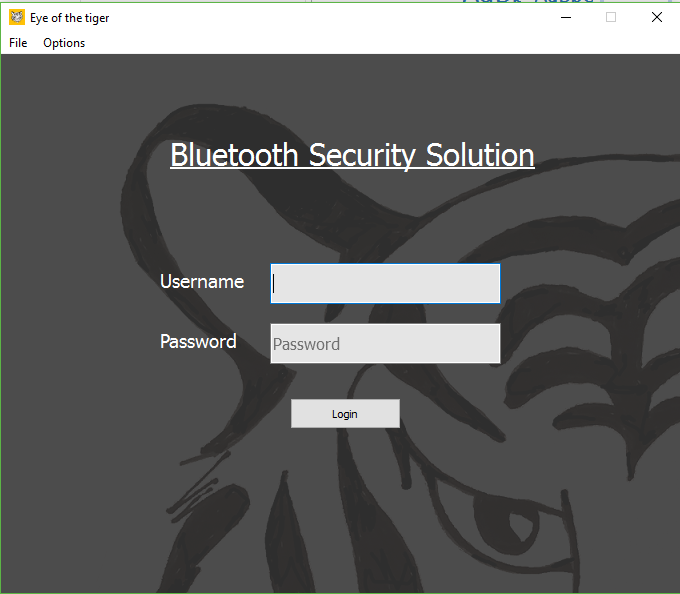


Figure 8: The login screen for the final view

The main page of the final UI consists of a tab view with three tabs: one each for the administrator, courses and students view. Instead of using a table to display all the information, like the prototype, each record is displayed on a custom UI widget, which is then dynamically added to each view on the tabs. This allows all of the attendance tracking information and class information to be displayed to the user in a unique and distinguishable format. It allows two different student/administrators/courses to be easily distinguishable from each at first glance. Each UI widget also includes a delete button to allow users to delete students, courses or administrators from the database. The student and administrator UI widgets also include an expand button which allows additional information for each student/administrator to be displayed to the user.

The student/administrator view now displays all the necessary information needed to convey their attendance record. It includes their current status, current class, entry/exit times, timetable, and total number of lates and absents as well as their first and last names. Initially it displays only the students first and last name, their student ID, current status and entry/exit times, and on expansions displays the rest. In addition to this, important attendance information is highlighted by making the student’s/administrator’s status and current location different location colors depending on whether they are absent, late or present.

The main page also includes, at the bottom, five buttons that each performs additional functions. From left to right, the buttons are the search button, refresh button, configuration setting (to change school time settings), the reset button and the logout button. With the exception of the refresh, reset and logout button, all buttons open a dialog page, which perform another action and will be discussed later on. The refresh buttons allows the user to manually update all the information currently being displayed by the main page (which satisfies one of the requirements) and the reset button resets all the tracked attendance information to their default values. This serves the purpose of allowing the school dynamically change the school time settings during school hours in the event of an emergency which may delay class schedules. The logout button logs the user out. Figures 9, 10 and 11 display the main page, the student/admin UI widget and course UI widget of the View component.

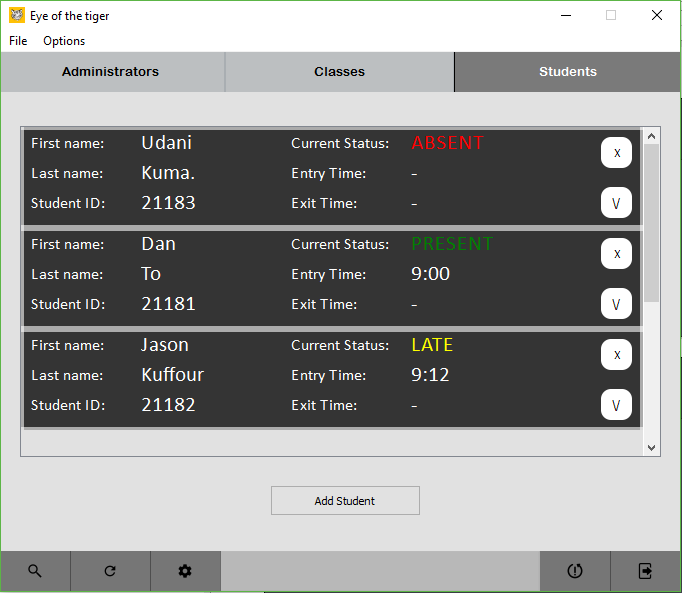
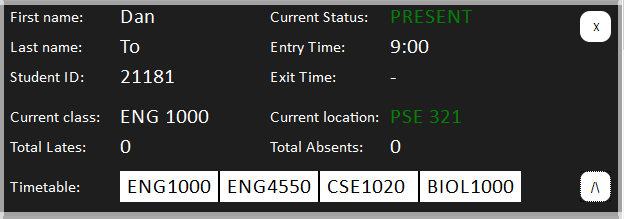
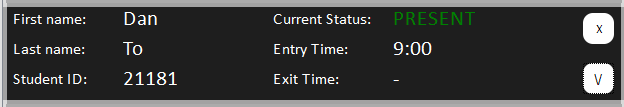


Figure 9: Main view of the administrator software. Currently displayed is the student tab which displays allow student records, an add student button and from left to right at the bottom, the search, refresh, settings, reset and logout buttons

Figure 10: The normal student UI widget (top) displays a student’s first name, last name, student ID, current status and entry and exit times. The expanded student UI widget (bottom) includes the additional information of the student's class and timetable



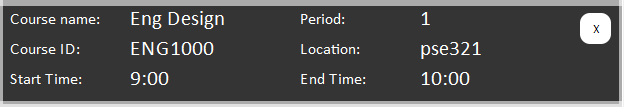
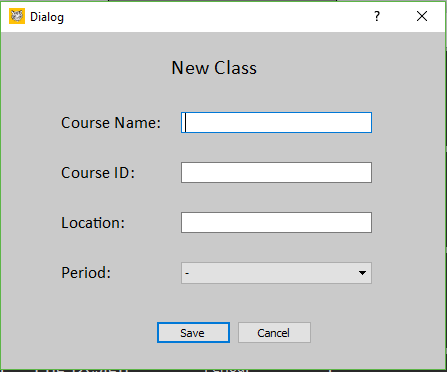
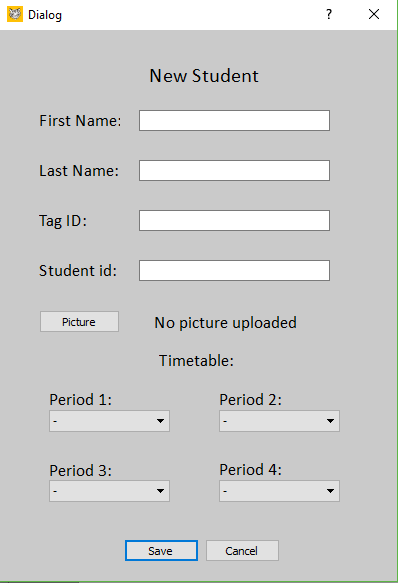


Figure 11: Course information UI widget which displays the course name, course id, start time, end time, location and period

The new course and new student/administrator views are displayed in figure 12. Unlike the first prototype UI, these two pages now include input validation as well as referential integrity checking. Both pages check that the new course and student does not have the IDs that are already present in the database. The new course page now only asks for the period of the course instead of start and end times; this ensures that all courses can have their times dynamically configured by the school through the configurations page. This is also used in the new student/administrator page to create their timetable by ensuring that courses are only added by their periods as opposed to the user manually checking and entering it themselves. This maintains referential integrity in the sense that non-existent courses are not added to timetables. The new student page also includes an image upload field, which can be used to add the student’s picture to the database. Whenever a new student or course is added, a message is displayed to indicate whether they were successfully added to the database or not. This action is performed by the model component.

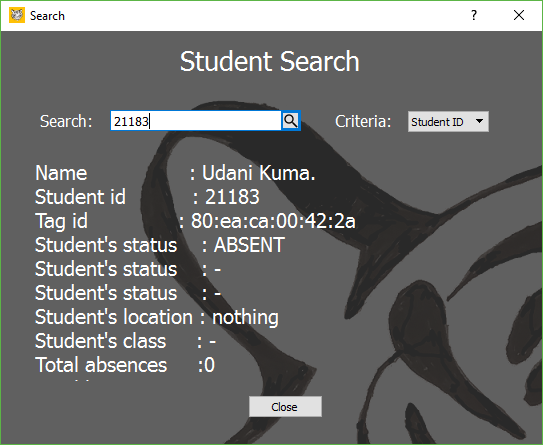
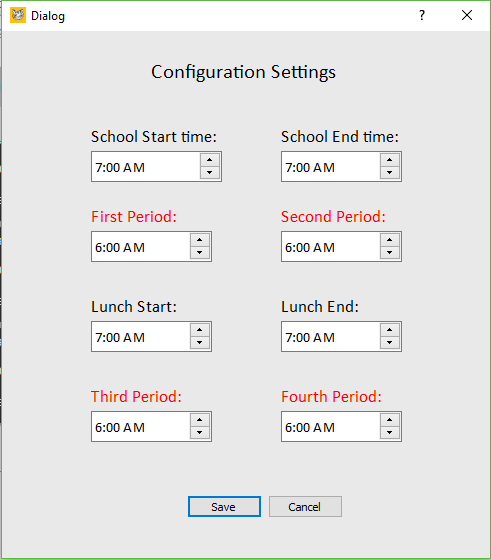
Figure 12: The new course dialog page (left) and the new student/administrator dialog page (right)



The search and configuration buttons at the bottom of the main page open a new dialog page to perform the searching functions and configuration settings. The search dialog page allows the user to search student, course or administrator records to find specific information about each category. The student and administrator search allows the user to search by the tag ID or their school ID by specifying the criteria in the dropdown box beside the search field box. While the course search only allows you to search by course ID. This ensures that only one result is obtained and displayed to the user. The results of the search are displayed on the same page with all the data pertaining to that result being shown.

The configuration setting page displays allow the school’s time settings which includes the start time, end time, first, second, third and fourth periods start time and the lunch start and end times. These fields are easily configurable by the user and on clicking the save button, input validation is performed to ensure that the times entered are consistent and valid. The end times for the first, second, third and fourth periods are implicitly specified to be the start times of the second, lunch, fourth and school end times. The search page and settings page is displayed in figure 13 below.

Figure 13: The student search page (left) with displayed search results and the configurations page (right) with default time settings



The final UI consists of a total of 9 Java classes with each handling a different aspect of the UI. They are the following: AdminTabs, CourseTab, StudentsTabs, which are responsible for the UI widgets used to display the information, the Settings, SearchDialog class that display the configuration settings and search dialog boxes respectively, New\_admin, New\_course, and the New\_student class which display the pages responsible for adding new information to the database. The last class, which is called the MainView, is the main page that displays all the UI components. It is the class that is used by the controller component to create and display the View component.

### 4.1.3 The Controller Component

The main function of the controller component is to facilitate the interaction of the model and view components. The controller class (component) provides an easy to use interface for the View component to obtain and pass information to the model component which can then update the database. The controller has an instance of the Model class (main representative class of the Model component) and an instance of the MainView class (main representative of the View component).

Most methods of the controller class are used by the MainView class but only the Settings and SearchDialog classes use the methods directly. The controller provides methods for obtaining all the information needed by the View component to display to the user and also provides method for sending information to the database that is obtained through the view component. It acts as layer between the Model and View component and provides methods to facilitate communication between the two.

The controller component is the main class for starting up the administrator software. This is done by creating another Java class called Main which instantiates an instance of the controller class and then calls the method activate which runs the whole software. The following figure is the simplified UML diagram for all three components of the administrator software.

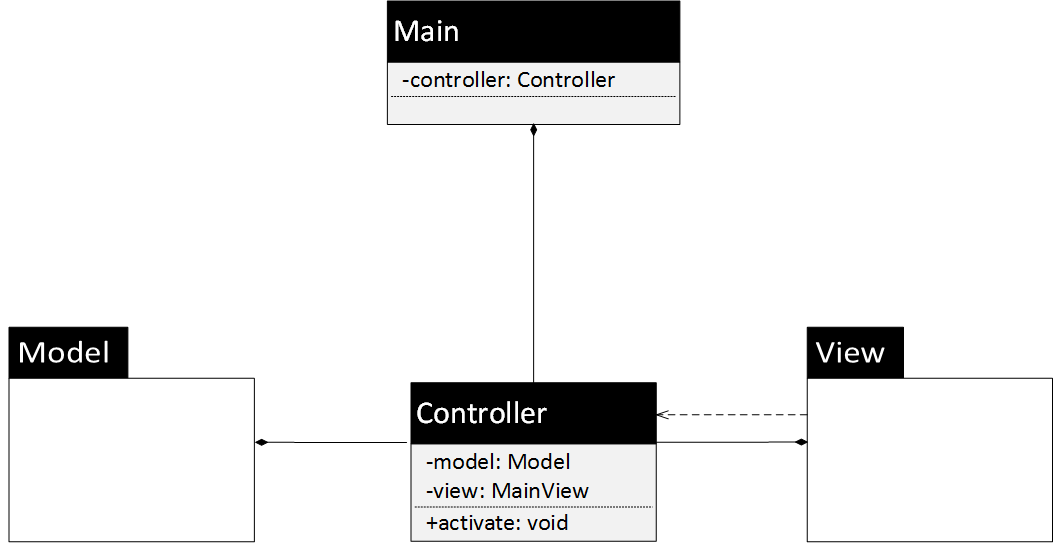
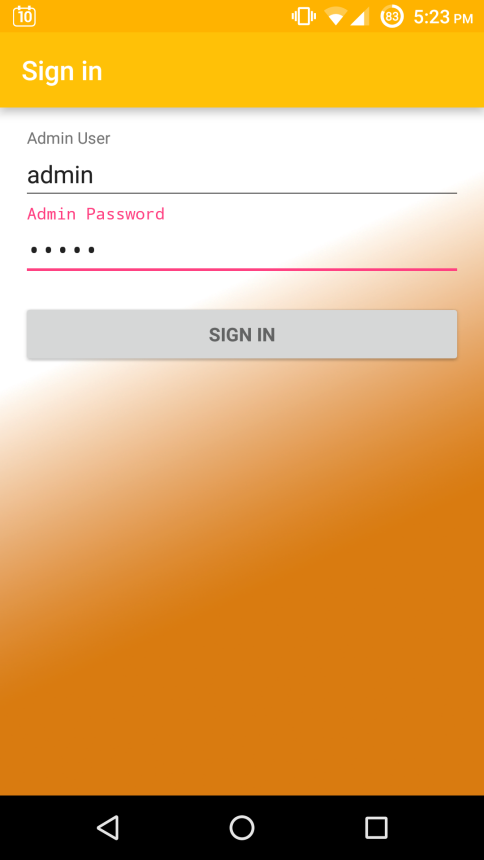
****

Figure 14: UML diagram for all three components of the administrator software.

## 4.2 Android Application

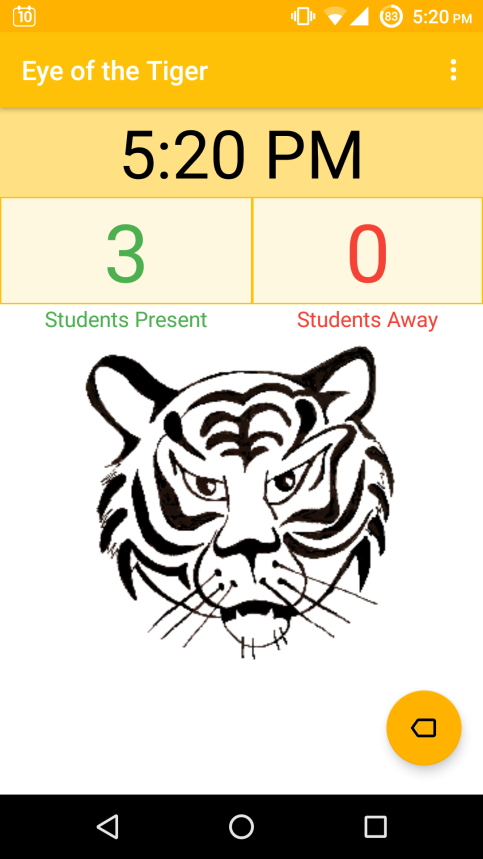
The application will use the Android Operating System. Communication with our cloud database using Cloudant works syncs information from the database with the Android application. The main page displays the number of students in the classroom for Present and Absent. The Expected Attendees page displays the list of students in the class. The data is presented in a table format with a single student per row, which displays student ID, last name, first name and status. All student data is retrieved by the app, but clicking on the student’s names will bring up details student profiles, of timetables, profile pictures and classroom locations. Currently, the application will run on any android device running operating system version from Jellybean 4.1 to the latest Marshmallow 6.0.0. The bright orange colour represents the company’s theme of black, white and orange. It is also mean to be bright and attractive.

Android was chosen as the deployment platform because it permits access to wider range of devices. The cost and variety of android devices is also much cheaper and more widespread than Apple devices. Java was chosen as the coding language because it is natively supported by android and possesses high level syntax, making it is easy to program complex graphical user interfaces. The Java community is also incredibly large possessing easily obtainable implementations for a variety of database, network and graphical tasks.

After loading the splash screen for the app, a login page is shown. Each teacher and administrator is registered in the database that needs authorization to use the app.

After verifying correct login information, the app leads to the main page.

Figure 15: App login page

The home page shows the current time and number of students present and away.

Swiping the screen to the left leads to the Expected Attendees page. A button on the right corner shows the direction for the swipe gesture.

Clicking on the button will give help for the swipe gesture and leads to the Expected Attendees page. Clicking the back button will prompt users for logging out.

The page is meant to be simplistic with intuitive gestures and buttons.

Figure 16: App main page

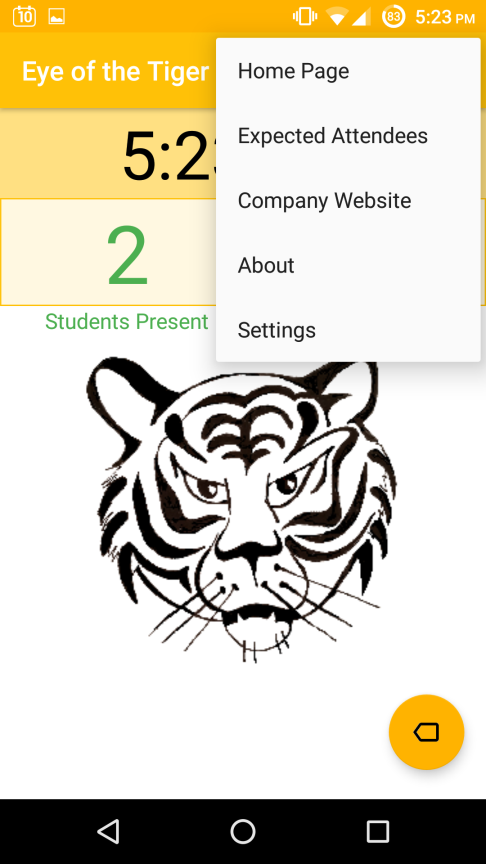
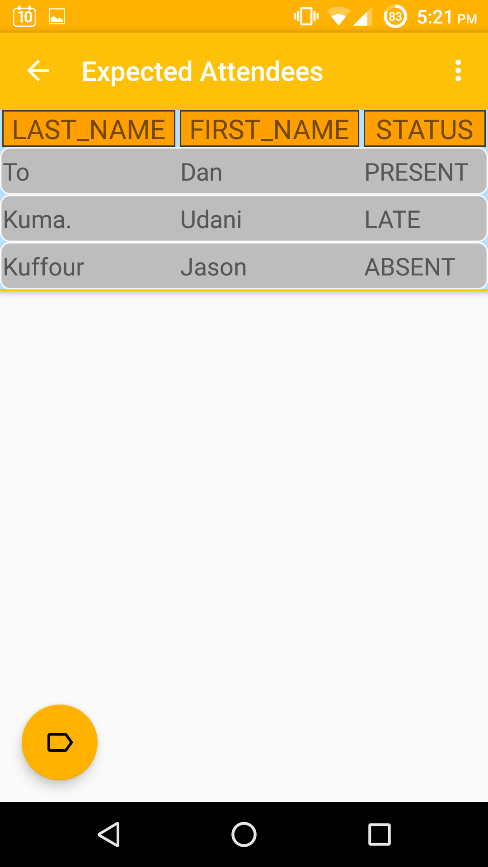
Most pages will have an options menu at the top right corner. Loading the options menu will have options to navigate to different pages easily, company website links, About page and logging out of the account.

Figure 17: App options menu

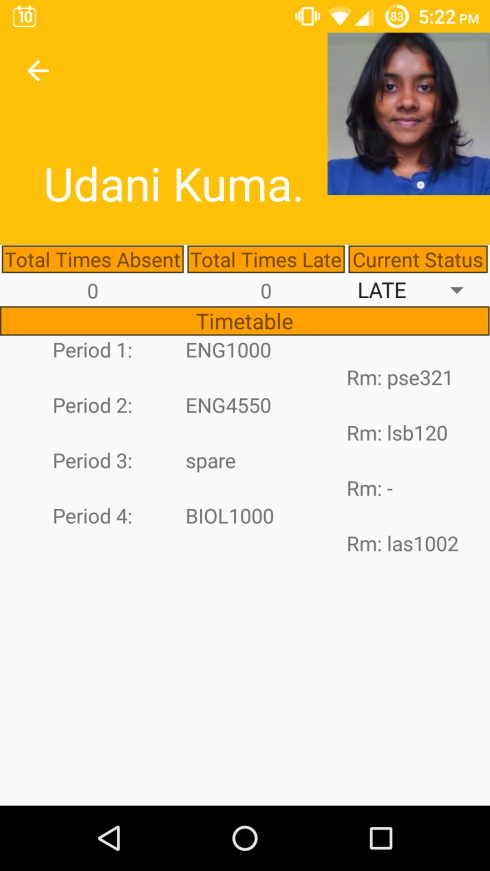
Swiping the page will lead to the Expected Attendees page. This page will show the list of students in the class.

Information is listed by last name, first name and status. The three statuses are Present, Late and Absent.

A button on the left corner shows the direction for the swipe gesture. Clicking on the button will give help for the swipe gesture and leads to the home page.

Clicking on the student will load more detailed student information.

Figure 18: App Expected Attendees page

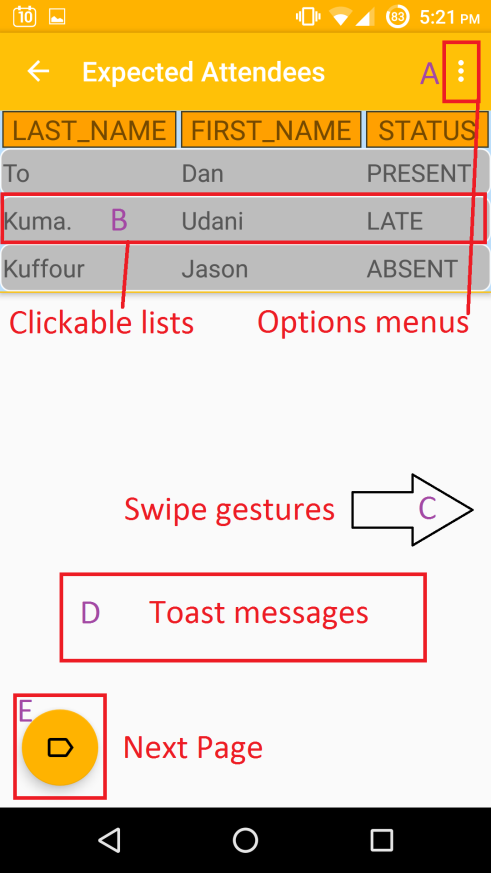
This page shows detailed student information. The right side has a student picture used assisting in verifying students.

The total times absent and total times late keeps a counter of the total number of these statuses. This information will allow teachers to record it in student report cards without having the need to manually count the statuses themselves.

The current status dropdown allows teachers to change the status of the student. For example, a student that was previously away could arrive late for class.

The timetable list shows the class periods, name and room locations.

Figure 19: App detailed student info page

A – Options Menu will bring a dropdown of different pages to navigate to or for logging out of the app

B – Student list is clickable and will load the detailed student profile

C- Swiping Left or Right will load Expected Attendees or Home page

D – Toast Messages appear to notify users of loading pages or status changes

E- Button that provides gesture help and leads to Expected Attendees or Home page

Figure 20 App interactive buttons and gestures

## 4.3 Database

The database used in storing the attendance information obtained by our project is IBM’s Cloudant. It is a no-SQL DbaaS which allowed us to define our own schema for storing data as well as making the data available to our app and administrator software.

The Bluetooth security system uses a total of 5 databases on Cloudant that each serve a different purpose. One database is used to store the configuration settings of a school, two databases are used to store the dynamic and static information for each student, the fourth database is used to store information about school administrators and teachers and the fifth database is used to store information about each course. Each database has a different schema used to organize the data they store and also provides a fast and easy method for retrieving and updating the data they store.

All data in stored in a cloudant database must have unique ids which are stored in a field called ’\_id’. This field is used by cloudant retrieve specific records from their database and cannot be changed. Additional fields can also be defined to be used as search indexes by cloudant. The fact that cloudant enforces the the ids of each record must be unique is exploited in the definition of our data schemas for some of our databases. The schemas defined for each database are not enforced by cloudant but rather by the user of the database, hence the schemas we have defined is more of a rule of thumb we have used in storing and retrieving data from the database. The schemas used and the purpose of each database is discussed in the following sections.

### 4.3.1 Configurations Database

The configurations database is used to store information about the school such as period start and end times, lunch start and end times and when the school starts and ends. It contains only one page/record and is accessed by the Scanner firmware and also updated by the administrator software. Through the configuration database, a school can dynamically change the start and end times of each period and also change the tracking functions of the Scanner system since the configurations database is accessed every time the attendance information of a student is being updated ( in stage 3 ). There are no special search indexes defined on the configurations database since all the information is required from it. The configurations database has the following schema:

|  |  |
| --- | --- |
| Field name | Value |
|  |  |
| \_id | “configurations” |
| first\_period\_start | number : number |
| second\_period\_start | number : number |
| third\_period\_start | number : number |
| fourth\_period\_start | number : number |
| Lunch\_start | number : number |
| school\_start | number : number |
| school\_end | number : number |

The value “number : number“ means that the data stored in those fields must be of the form number followed by a colon then followed by another number. So the value ,9:00 is valid but 3:45:40 is considered invalid. The end times of each period can be determined from the end times of other periods.

### 4.3.2 Dynamic Student Information Database

The dynamic student information database is used to store the attendance information of students in the school. All data obtained by the Bluetooth Scanner for each student is stored in this database. This database is both accessed and updated by the scanner firmware and the Android app and is only accessed by the administrator software. The administrator software also has the ability to and or delete records from this database and whatever student record is added/deleted in this database, the corresponding record is also added/deleted from the static student information database. This database has the following data schema:

|  |  |
| --- | --- |
| Field name | Value |
|  |  |
| \_id | Student tag id ( string ) |
| user\_current\_class | String |
| user\_location | String |
| user\_status | Either ‘PRESENT’, ‘LATE’ or ‘ABSENT’ |
| entry | Number : number |
| exit | Number : number |

The schema for dynamic student information database uses the unique id field to store the student’s tag id. The tag id is a unique string that will differ for each student and hence can be used to uniquely identify them in the database. It is also the data detected by the Scanner firmware and used to update the corresponding student’s attendance information.

### 4.3.3 Static Student Information Database

The static student information database is used to store the information of a student that is not related to their attendance tracking information. It stores information about their name, student number, timetable, their profile picture and their total number of absences and lates. This database is accessed by the Android app and Scanner firmware while the administrator is the only one that can create/delete new records to be stored in this database. Whenever the administrator software is used to add new records, it simultaneously adds it to both the static and dynamic databases and when deleting it deletes from both of them. This ensures that every student has an attendance tracking record. This database has the following schema:

|  |  |
| --- | --- |
| Field name | Value |
|  |  |
| \_id | Student tag id (string) |
| user\_first\_name | String |
| user\_last\_name | String |
| user\_id | String |
| user\_number\_of\_absences | Number |
| user\_number\_of\_lates | Number |
| user\_timetable | String values separated by “/” |
| attachments | Png picture file |

Similar to the dynamic student information database, this database also uses the tag id as the id value. The field user\_timetable is used to store the student’s timetable as well as their class location. For example if they are taking two courses called c1 and c2, and both courses are located in the class room l1 and l2 respectively, then their user\_timetable field would be the value “c1/l1/c2/l2”. This field format is used by the Scanner firmware to easily check if a student belongs in a class based on their timetable alone. There is a special search index defined on the user\_id field which enables records in the database to be searched by those fields. This index is by the administrator software to ensure that new students do not have the same user\_id or tag id as another student.

### 4.3.4 Administrator Information Database

This database is used to store all the information related to school administrators and teachers. This database combines the schema of both the dynamic and static student databases but excludes the entry and exit times as well as the number of lates and absences. It also uses the tag id as the value for the id field and has a search index on the administrator id. It has the following schema:

|  |  |
| --- | --- |
| Field name | Value |
|  |  |
| \_id | Administrator tag id (string) |
| admin\_first\_name | String |
| admin\_last\_name | String |
| admin\_id | String |
| admin\_timetable | String values separated by “/” |
| admin\_location | String |
| admin\_status | String |
| admin\_current\_class | String |
| \_attachments | Png picture file |

### 4.3.5 Class Information Database

This database stores the class name, class id, period info and the class location. It is exclusively used by the administrator to maintain referential integrity between a student’s timetable and the courses offered in the school. It has a search index on the field class\_period which is used by the administrator software to get all classes that occur in a certain period. The id field is used to store the course id of the class. This database has the following schema:

|  |  |
| --- | --- |
| Field name | Value |
|  |  |
| \_id | Course id (string ) |
| class\_name | String |
| class\_period | Number |
| class\_location | string |

## 4.4 Bluetooth Scanner Firmware

  The Bluetooth scanner firmware is the software that runs on the Bluetooth scanner. It is responsible for detecting Bluetooth tags and updating the corresponding student’s attendance information. The scanner firmware can be divided into three stages/programs that are all implemented in different programming languages. This kind of software architecture is called pipe and filters. By using this software architecture, the scanner firmware can support concurrency which will be need to update multiple student attendance information.

The first stage of the scanner firmware is responsible for scanning and detecting Bluetooth tags. The second stage is responsible for formatting the input data, from the first stage, to a format more suitable for the stage 3 and the third stage is responsible for updating the cloudant database with their current attendance information. The first, second and third stage are implemented in Python, Perl and Java respectively.

An overview diagram of the scanner firmware is displayed in the figure below. Detailed descriptions of each of the three stages are in the sections that follow.

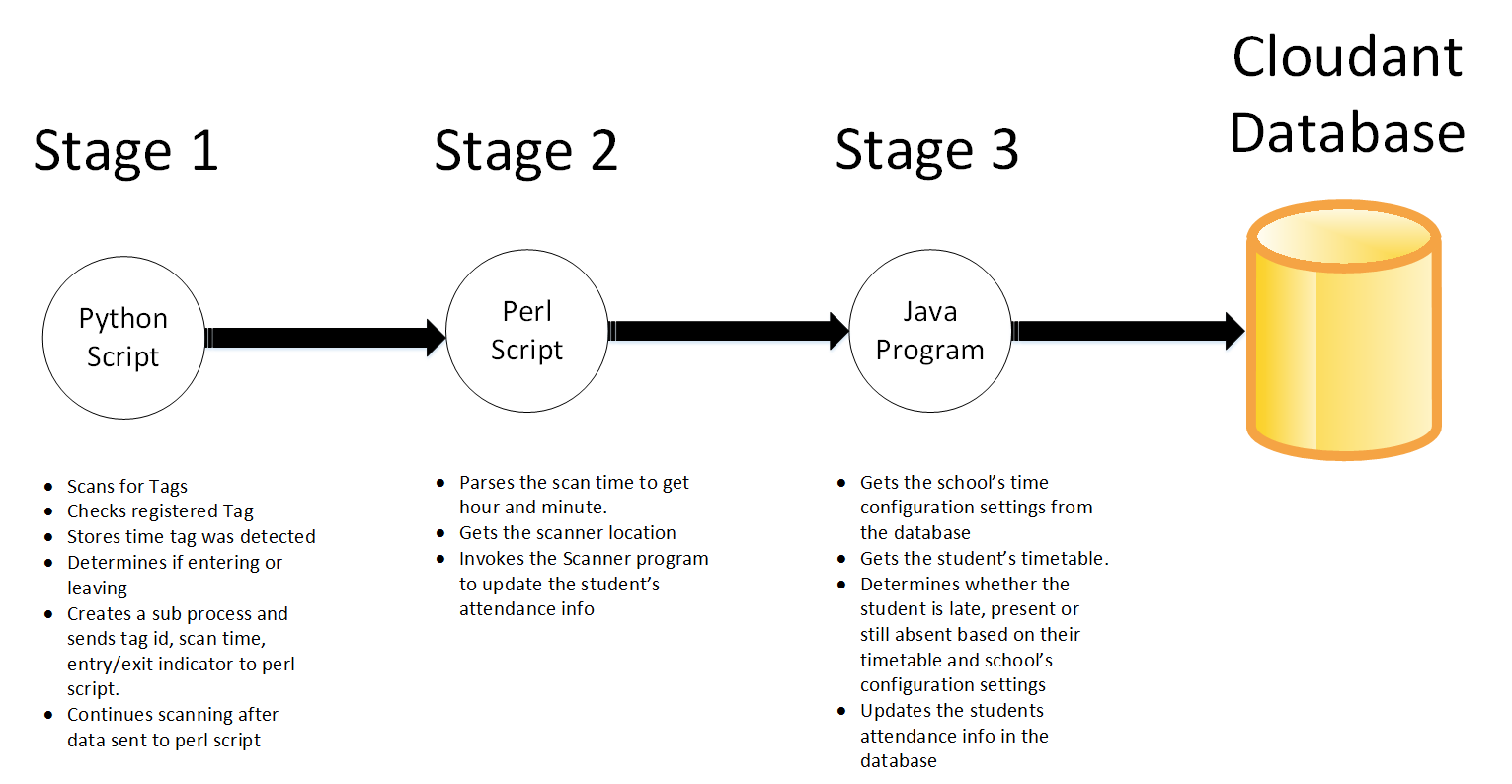


Figure 21: Diagram displaying the flow of inputs through the scanner firmware to the Cloudant database

### 4.4.1 First Stage (Python Script)

The first stage of the scanner software, which is implemented in Python, is responsible for scanning and detecting school registered tags. The decision to use Python was based on the fact that Python has a library called ‘pyBLE’ which provides an interface to using a Bluetooth adapter to scan and detect Bluetooth low energy devices. This library provided the means of keeping the Bluetooth adapter in constant scanning mode.

Using Python a script was created to constantly scan for Bluetooth tags every 2 seconds. Whenever the script detects a Bluetooth device, it checks whether the detected device is a school registered tag or another foreign device. If it is a foreign device, the script ignores it and continues scanning; otherwise it performs a few other actions before passing the data to stage 2. Once a student tag is detected by the script, it stores the time it was scanned to the last second and determines whether the student is leaving or entering the classroom by comparing the detected tags RSSI value with the entering/exiting range of the classes. If the tag is considered to be entering for the first time, the script stores the tag’s mac address in a dictionary and a Boolean value to indicate that the student’s attendance information has been updated already. If the same tag is detected again and still within entering range, then the tag looks at the Boolean indicator and does not update the attendance information for it if it is true. If the tag is in the exit range or leaving, the Boolean indicator is set to false and the script passes it to stage two. The Boolean value false once again serves as an indicator to the script to update that student’s attendance information if they are within entering range.

Once the script determines that the attendance information for a tag needs to be updated, it passes the time the tag was detected, its Mac Address (tag ID) and a string to indicate whether the person is entering (enter) or leaving(exit). The transition to stage 2 is made by spawning a separate thread in Python that executes the Perl script. After creating another thread, the script continues scanning. This is to ensure that there is no delay in detecting and updating new student attendance information.

### 4.4.2 Second Stage (Perl Script)

The second stage of the scanner firmware is responsible for formatting the input from the first stage into something more suitable for the last stage.

Specifically the Perl script parses the time value from the Python script and separates it into hours, minutes and seconds. The Perl Script also contains the location of where the Bluetooth scanner is located in. Once it obtains the hour, minute and second the tag was detected, it discards the seconds and passes the tags mac Address (tag ID), the hour the tag was scanned, the minute the tag was scanned, entry/exit indicator and the class room location to the last stage.

The transition to the third stage is made by executing the Perl function ‘exec’ and executing the jar file, with command line arguments, that contains the final stage.

### 4.4.3 Third Stage (Java Program)

The last stage of the scanner firmware is responsible for actually updating the student’s attendance information in the database. It starts by first redundantly checking that the inputted tag’s mac Address (tag ID) belongs to a registered student in the database. If it doesn’t, it ignores it but otherwise updates the student’s attendance information as described below.

The Java program first begins by pulling all the student’s information and the school’s configuration settings from the database. The tag ID is used to pull that specific student’s information. The information is obtained from the database in parallel to speed up the response time of the Java program; this action is performed by the DBpull Java class. The configuration settings is used to obtain all the periods’ start and end times and also allows the scanners to be configurable through the administrator software.

Once the data is obtained, the Java program then uses the hour, minute and location inputs from the second stage and the student’s timetable to determine whether the student is present, absent or late. In either three cases, the student’s current location and current class are updated in the database to the current values. The entry and exit times of the student are also updated according to the whether they are entering or leaving. A student is considered still absent if they don’t belong in the current location in which their tag was detected.

When all the student’s current information is updated, the Java program updates their record in the database and stops execution. The DBpush Java class is responsible for updating the database. The third stage consists of 4 Java classes. There’s the JSONhandler class that handles all data sent to and received from the database, the DBpush and DBpull which function as described above, and finally the main class that abstracts the use of DBpush and DBpull to two simple methods for the Main java class that executes it. The figure below shows the simplified UML diagram for the Java program.

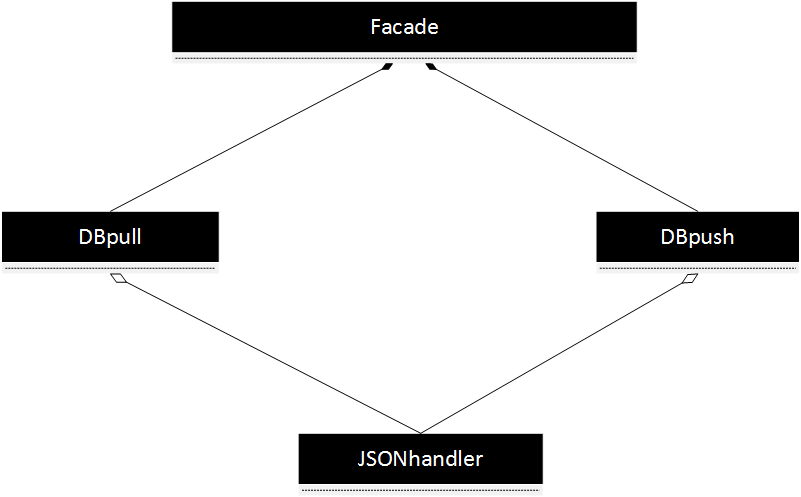


Figure 22: Diagram for the Java program responsible for updating a student's attendance information in the database. Refer to the Appendix for a detailed UML description of each class

## 4.5 Bluetooth Tag & Scanner Hardware

The Bluetooth Tag was designed on the Dialog Semiconductor DA14580 Development Board (Dev. Board). The Dev. Board contains an ARM Cortex M processor with Flash memory and Electronic Erasable Programmable Read Only Memory (EEPROM) as well as One Time Programming memory (OTP). The DA14580 Bluetooth chip is on the QNF40 Daughterboard attachment.

In the current design, the Bluetooth chip is programmed to broadcast a Bluetooth frame in the proximity profile. This frame is capable of holding different fields; one of the fields contains the Universal Unique ID (UUID). Another field will hold the reference Received Signal Strength Indicator (RSSI) value. Using the UUID and RSSI along with the MAC address of the Bluetooth chip, we are able to identify the unique tag and determine if the tag is in the proximity of the scanner.

As for the Bluetooth scanner, it is based on the Raspberry Pi. The Raspberry Pi provides powerful hardware with great flexibility. With a Bluetooth add-on, the Raspberry Pi will be able to pick up the broadcasted Bluetooth frames. Once the broadcasted frame is detected, the scanner will need to determine whether the RSSI value is within a threshold and update the status of the tag in the database accordingly. With the flexibility the Raspberry Pi provides, this can be easily done.

# 5.0 Engineering Design

## 5.1 Administrator Software

The administrator software was designed using the SCRUM development process. The SCRUM software development process is the technique of developing software in incremental parts. For each part of the software currently being developed, you analyse a part of the requirements, design code to address it, implement, test it and maintain it so that it is in accordance with the requirements and specifications of the main software as the project progresses.

Each component of the software was developed in incremental stages which facilitated room for the growing and ever evolving requirements of the project. As evident by the fact that there was a prototype UI, the requirements for the administrator software kept growing from initially being only that the administrative staff should be only able to add/view students, courses or administrators. As the project progressed, the needs of the school, which included being able to determine the location of any student in the school, required the administrator to evolve into something more that would satisfy these new requirements. Other requirements also arose such as being able to search the records being able to display entry and exit times, etc that were able to be analysed, designed, implemented and integrated into the current version of the administrator software.

The use of the SCRUM process and the Model-view-controller software architecture allowed both our team and the administrator software to gracefully address these ever changing requirements.

## 5.2 Android Application

The Android application allows teaching faculty members to be able to access, and modify student attendance information for their classes with mobile devices. Information that can be displayed and modified consists of a list of students present and absent from the classroom and the students’ timetable data. To assist with student attendance validation, each student will have a unique profile image. The user interface was designed to be user friendly and easy to navigate. Features include interactive swiping gestures, clickable tables and helpful action prompts. The Android platform was chosen, due to covering around 60% of the market share in the mobile industry, according to the data from Net Market Share. For testing, all buttons and clickable items were pressed with different combination patterns to simulate user behavior. Loading of data for the app was optimized to reduce delays when retrieving information from the database.

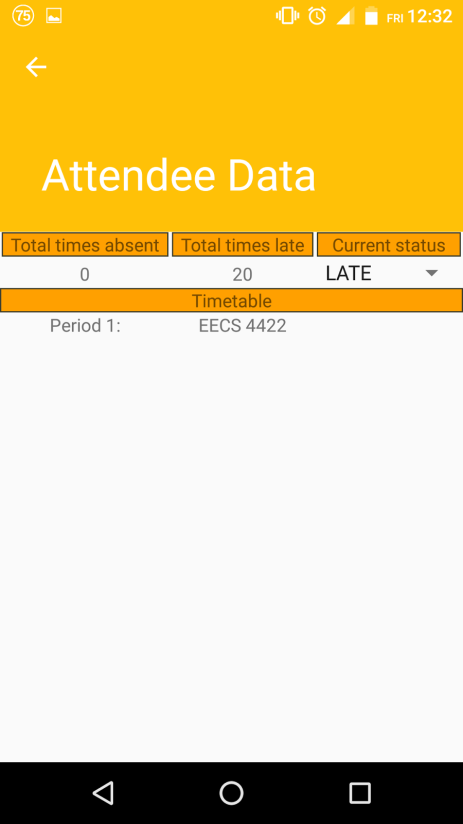
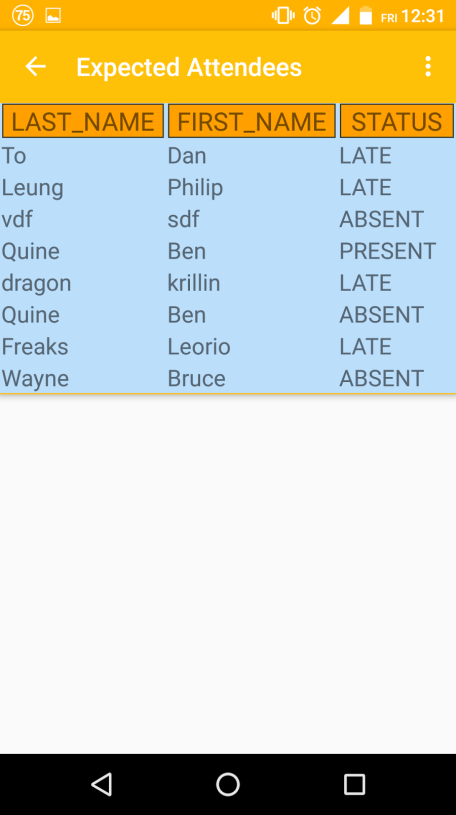
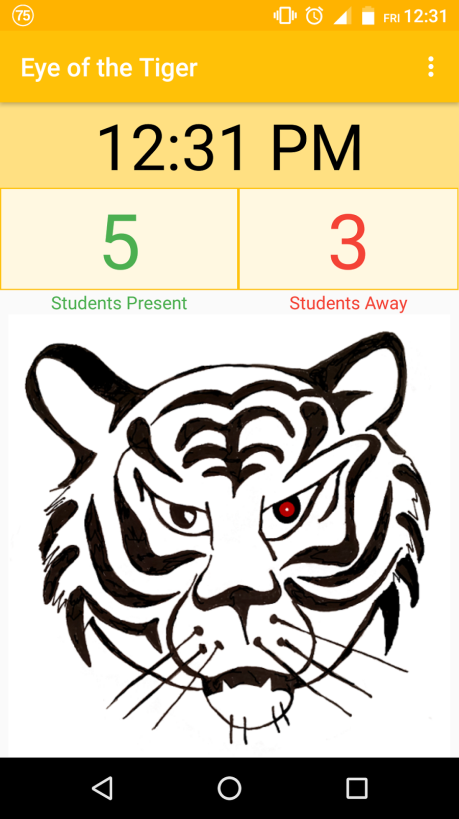


Figure 23: Previous revisions of the app. There were improvements for UI, interactive gestures and buttons

## 5.3 Database

To handle the data associated with our product, an inexpensive and easily scalable solution was needed. A cloud database was chosen to support our product because it eliminates hardware costs associated with a physical database. The solution for the database was free, provided the data usage limit is under $50 a month. Furthermore, cloud solutions are easily scalable and able to accommodate any size of data being received. Using a cloud database as a service (DbaaS), it introduces data protection and security issues.

The IBM Cloudant service, which is the DbaaS, is hosted on IBM’s softlayer, Window’s Azure and several other established services. As a result, the data can be accessed through a variety of secure services, which will reduce the possibility of down time if one of the hosts were to go offline. All of these services also run frequent encrypted backups of their data, which further protects potential data loss.

If a user decides to use a physical local database, our software can be easily adapted to read from any type of database, and can be transitioned easily.

The current data schema for our databases was iteratively refined from various others due to the evolving requirements of the project. Initially all information related to courses including their start and end was stored in the class information database and there was no configurations database. The student’s timetable was just stored as a string that only contained the course IDs and whenever the Scanner firmware would check whether a student was late it or absent, it would search for the course that occurred during the current period at the current location. The requirement of improved response time and easily configurable school times enabled us to reevaluate our current schema, refine it and implement it into the current version it is now. Testing of the database include checking for proper format of data.

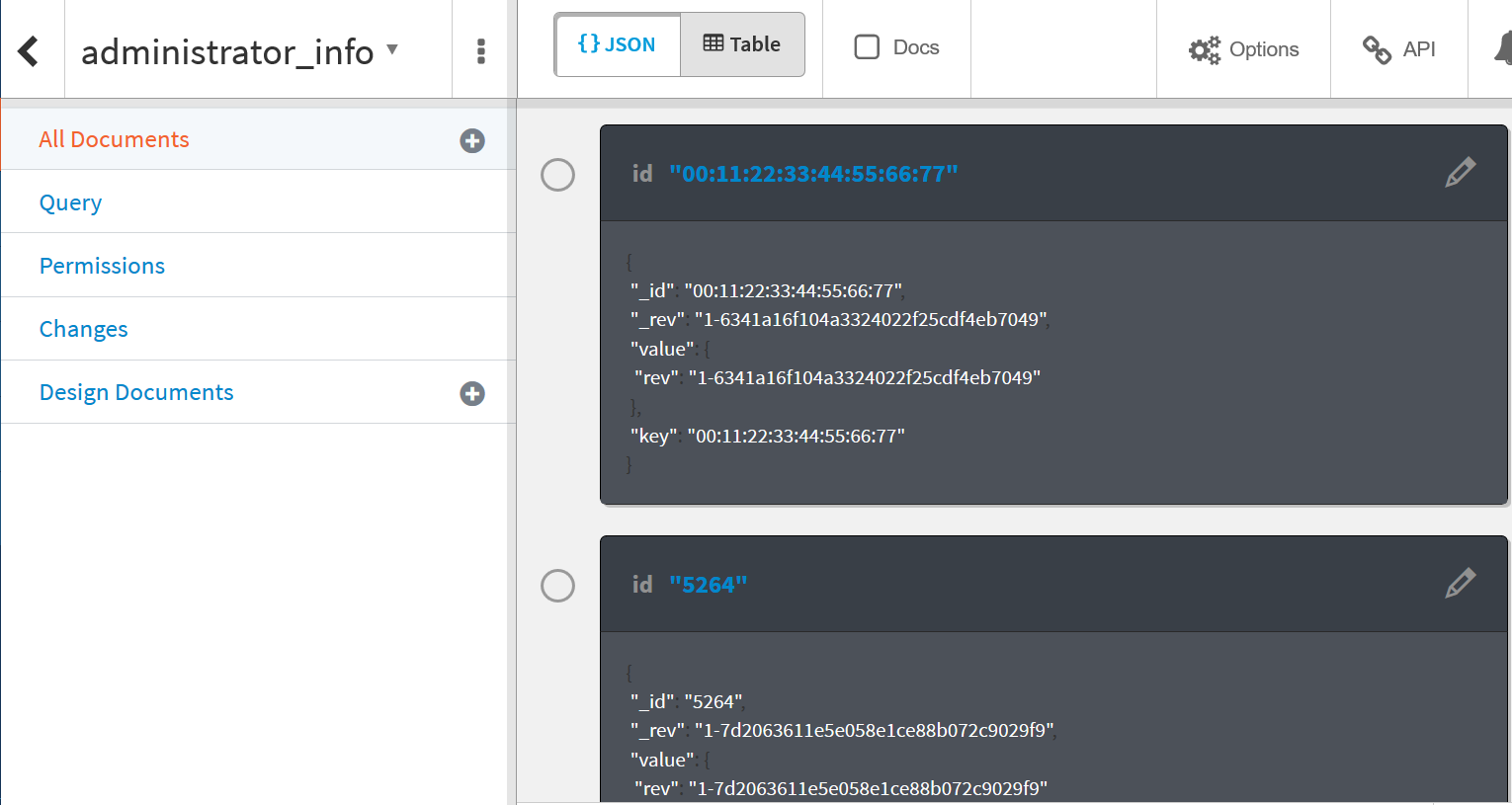


Figure 24: Cloudant Database

## 5.4 Bluetooth Scanner Firmware

The scanner firmware was designed and implemented in the same way as the administrator software. Initially the only requirement that the scanner firmware had was to update the student’s attendance information as either present, absent or late based on whether they were in the right class at the right time. But now, the requirement also include being able to update their entry/exit times as well as their current location and their current class.

The addition of those requirements necessitated the use of the three stage pipe and filter architecture to facilitate these requirements. The use of the architecture also enabled the scanner firmware to the multiple tags requirements as well as the increased response time. By spawning sub processes to handle stages two and three of the scanner firmware, stage one is free to handle other detected tags. The final iteration of the scanner firmware arose due to all these new requirements which were iterative analyzed, designed and integrated into iteration.

The use of the pipe and filter software architecture allowed each stage to be designed, implemented, tested and maintain independent of the other stages.

## 5.5 Bluetooth over Radio Frequency Identification

Originally, RFID was to be used for the automated attendance tracking solution. The original concept was to attach RFID sensors on the doors to each classroom and as the students pass through, the RFID sensors will detect the RFID tag and update the student’s status. However, in order to achieve this, we would have needed to employ active RFID over passive RFID. Active RFIDs have an advantage over passive RFID in range as the RFID tag will be constantly powered by a coin cell battery whereas the passive RFID will need to be powered by the signal from the scanner. As a result, the passive RFID tag will need to be in extremely close proximity to the scanner to be able to get powered. Furthermore, to achieve the required range, the active RFIDs will need emit a more powerful signal over passive RFIDs, resulting in health concerns. Finally, the costs of active RFID systems are extremely costly compared with passive RFIDs.

## 5.5 Bluetooth Tag & Scanner Hardware

Due to the complications of using RFID solutions, we decided to use Bluetooth Low Energy instead because it addresses most of the problems with RFID. With Bluetooth Low Energy, the Bluetooth scanner can retrieve packets over a wide area while keeping the signal power to reasonable levels. Furthermore, Bluetooth solutions are low-cost as it now commonly used in most technologies, such as mobile devices. The Bluetooth solution we decided to employ was to use a Bluetooth scanner to scan the air for Bluetooth Low Energy packets. The Bluetooth Tag will be running the Proximity Profile configuration and will be constantly broadcasting a Bluetooth packet. This packet will include important data, such as the MAC address of the tag, the Universal Unique Identifier (UUID), as well as the Reference Received Signal Strength Indication (RSSI) value. Using the MAC address, UUID, and the RSSI value, the scanner will be able to determine whether the Bluetooth tag is within the classroom and the Bluetooth scanner will update the database accordingly.

The Bluetooth scanner we decided to use was the Raspberry Pi. The options we had were either the Arduino or the Raspberry Pi. Comparing the two, the Raspberry Pi was more flexible as it was able to be configured to fit our specifications, while the Arduino was more limited in that respect. The Raspberry Pi had built-in USB ports and multimedia ports, which reduces the cost of having to buying separate components. Furthermore, the programming of the Raspberry Pi was simpler as it was capable of loading a Linux distribution, whereas assembly code was needed to program the Arduino. As a result, due to the simplicity of the Raspberry Pi over the Arduino, we decided the Bluetooth Scanner will be developed on the Raspberry Pi.

During development of the Bluetooth tag, we utilized the DA14580 Development Kit made by Dialog Semiconductor. The firmware was loaded onto the QFN40 Daughterboard via the Development board and was tested in conjunction with the Raspberry Pi. The development kit provided a flexible platform in developing the tag as it allowed us to test and debug the tag before having the hardware designed.



Figure 25: Development Board



Figure 26: Raspberry Pi

## 5.7 Bluetooth Scanner Casing

To encapsulate and protect the scanner circuitry of the Raspberry PI, a casing was 3D printed. The new casing model allows the scanner to be enclosed to secure the small chip inside a casing without any space for mobility, and to prevent the chip from being damaged. Furthermore, air hole locations on the casing allow the scanner to operate without overheating. The casing was painted to match the theme of the team, with orange and black colours. Earlier designs of the scanner casing was created with the software Maya, and the old design was not used due to not being able to properly secure the circuits, and had a large size.

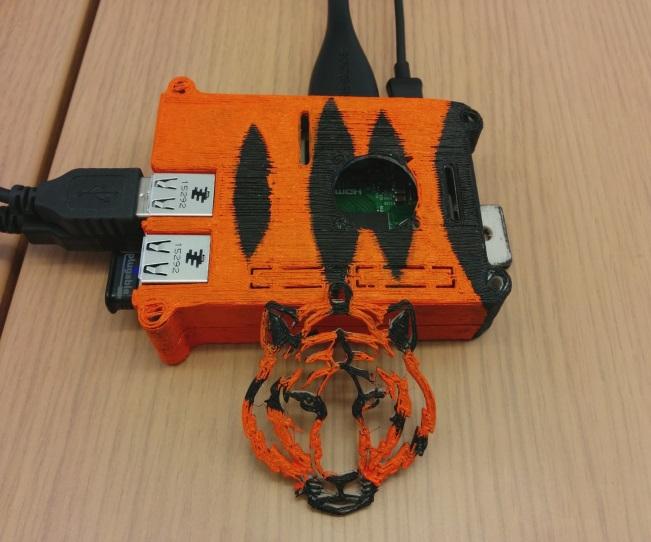
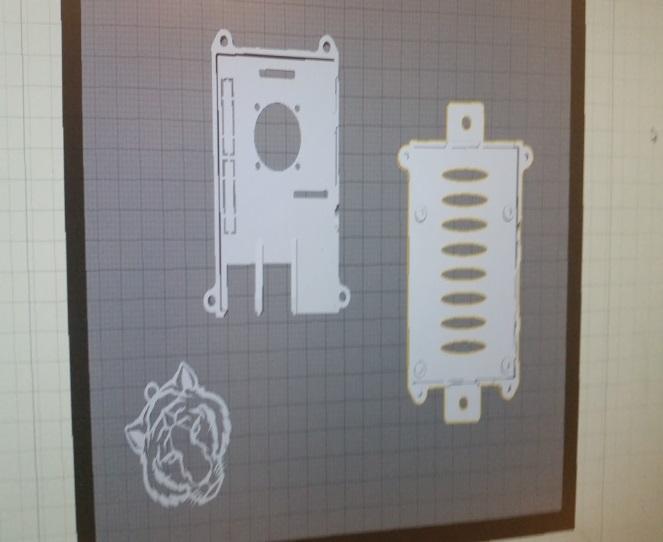


Figure 27: 3D CAD model and printed casing

# 6.0 Health and Safety

## 6.1 Health and Safety Measures Taken By The Team

The Eye of the Tiger team have been educated about the health and safety through the Lassonde School of Engineering Health and Safety information session. The team members are WHMIS (Workplace Hazardous Material Information System) certified. Due to the nature of the project, it was not required to have any other health and safety related training.

We have adopted the York University policies which are stated below.

* York University is committed to the prevention of illness and injury through the provision and maintenance of healthy and safe conditions on its premises. The university endeavors to provide a hazard free environment and minimize risks by adherence to all relevant legislation, and where appropriate, through development and implementation of additional internal standards, programmes and procedures.
* York University requires that health and safety be a primary objective in every area of operation and that all persons utilizing university premises comply with procedures, regulations and standards relating to health and safety [9].

The Eye of the Tiger team is committed to providing a work environment in which all individuals are treated with respect and dignity. Workplace harassment will not be tolerated from any person in the workplace. Everyone in the Workplace must be dedicated to preventing workplace harassment.

## 6.2 Bluetooth Health Concerns

The Bluetooth tags that will be used are Bluetooth Low Energy (BLE) chips that operate at very low transmitting power at 1mW. These chips are in the Class 3 category, which is the lowest tier for transmitting power with short ranges of 15 meters [7]. Bluetooth uses the same RF range as Wi-Fi at 2.4 GHz. Health Canada claims that Wi-F is safe with a low value of RF energy absorption; therefore, Bluetooth is safe as well, since it operates like Wi-Fi at short distances. Wi-Fi operates at higher powers and longer ranges, while Bluetooth uses low rangers and lower power. In comparison, cellphones operate at microwave frequencies and use high power, which is less safe than Bluetooth [8].

## 6.3 Bluetooth Tag Power Emissions

The Dialog Semiconductor DA14580 Bluetooth chip utilizes the latest iteration of Bluetooth, Bluetooth Low Energy (BLE). The benefits of BLE is that a new profile, the proximity profile, is supported. With the proximity profile, the Bluetooth chip will be in advertisement mode, constantly broadcasting a Bluetooth frame. It operates at a RF range of 2.4 GHz, similar to Wi-Fi. From the specifications of the DA14580 Bluetooth chip, the average current is 967µA while the average charge is 9.53µC. Compared with other devices, the power consumption of the Bluetooth tag is kept at minimal levels of 1 mW. In this proximity configuration, the average battery life of the Bluetooth tag will be from 3.7 months to 30.6 months with a 100 ms broadcasting interval [5].

# 7.0 Future Improvements

Due to the nature of our project there are a number of possible improvements or extensions that can be made it, some of which are school related and security related. Three of the main future improvements we have identified that this project can be applied to, are integration with the school’s “call home system”, school door locking mechanisms and the use of it in interactive classroom learning. All three improvements or extensions are discussed in the following sections.

## Integration with school’s call home system

Every high school in Canada has an automated system that calls the homes’ of students whenever they have been reported absent or late (some schools) for any of their classes every day. One major improvement for this project is to integrate it with the high school’s call home system. This would provide the benefit of further automating the entire school’s attendance system from manually checking a student’s attendance to manually entering this into their system. This would be the most logical extension to our project as it also provides the security benefits of informing parents whether their child made it school safely.

## 7.0.2 Integration with magnetic locks

The Bluetooth security system can also be used in conjunction with a magnetic door lock which can be applied as the school’s locking mechanism. By integrating our project with magnetic door locks, we can provide a means of only allowing students the ability to unlock them. This can be achieved by connecting the magnetic lock to a scanner and locking/unlocking based on whether a registered student tag is detected and verified by the scanner. A locking mechanism can then be placed on all school entrances except the main one which must be used by all visitors or strangers. This improvement further highlights the security abilities of our project as all unauthorized personnel will be force to enter the school through the main entrance while students will have access to the school through, some if not all, entrances.

## 7.0.3 Creating an interactive learning experience

Another possible improvement that can be made to the project, is the integration of clicker system which can be used for classroom participation. By making some additional hardware changes to each student tags and adding a few software modules to the Scanner, our project can easily be made to functions as a clicker system also. This provides the additional benefit of making each classroom an interactive learning environment where quizzes or discussions can easily be performed without time being wasted on marking. The results of each quiz can also be analyzed in real time to provide important feedback or discussions about the quiz or course material.

# 8.0 References

1. "Back to School... by the Numbers." Government of Canada, Statistics Canada. Web. 7 Dec. 2015.

<http://www.statcan.gc.ca/eng/dai/smr08/2014/smr08\_190\_2014>.

2. "Education in Canada." Education in Canada: An Overview. CEMC. Web. 7 Dec. 2015.

<http://www.cmec.ca/299/Education-in-Canada-An-Overview/>.

3. "PCB Cost Comparison Calculator." LadyAda. Web. 7 Dec. 2015.

<http://www.ladyada.net/library/pcb/costcalc.html>.

4. "PCB Pricing and Specifications." OSH Park. Web. 7 Dec. 2015.

<https://oshpark.com/pricing>.

5. "DA14580 Beacon reference application." Dialog Semiconductor. Web. 5 Jan. 2016.

6. "Operating System Market Share." *Operating System Market Share*. Feb 2016. Web. 10 Apr. 2016.

<https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8>.

7. Wright, Joshua. "Security Laboratory." *Dispelling Common Bluetooth Misconceptions*. SANS

Technology Institute. Web. 11 Apr. 2016. <https://www.sans.edu/research/security-laboratory/article/bluetooth>.

8. "Safety of Wi-Fi Equipment." *Health Canada*. Minister of Health, 1 Dec. 2011. Web. 11 Apr. 2016.

<http://www.hc-sc.gc.ca/hl-vs/iyh-vsv/prod/wifi-eng.php>.

9. "Occupational Health and Safety Policy." *Occupational Health and Safety Policy*. Web. 11 Apr. 2016.

<http://secretariat-policies.info.yorku.ca/policies/occupational-health-and-safety-policy/>.

# 9.0 Appendix

## 9.1 Appendix A (UML Diagrams)

The following are the UML class diagrams of all the Java classes used in developing the administrator software the stage 3 of the scanner firmware.

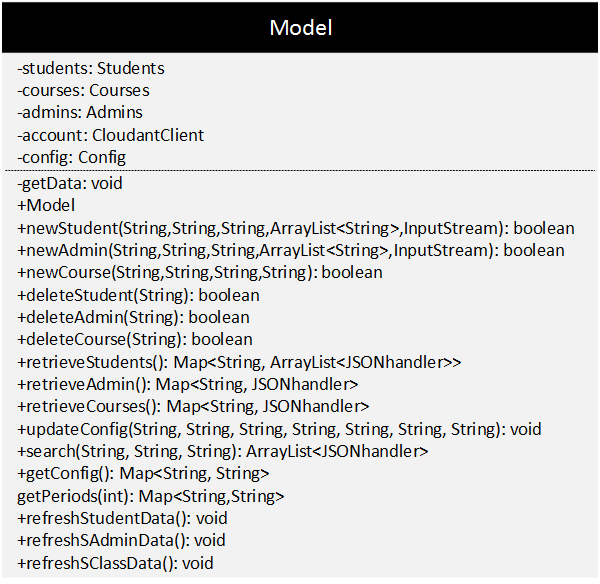


Figure 28: UML diagram for the Model class which resides in the Model component of the administrator software

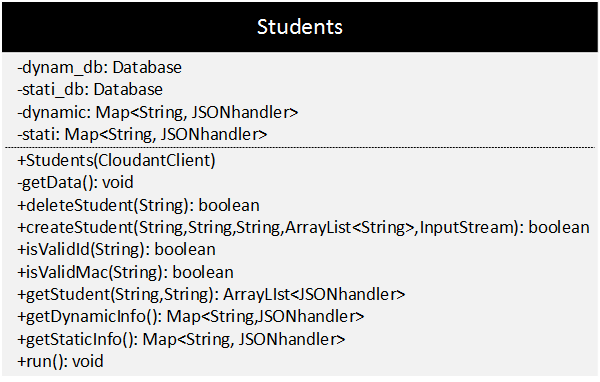


Figure 29: UML diagram for the Students class which resides in the Model component of the administrator software

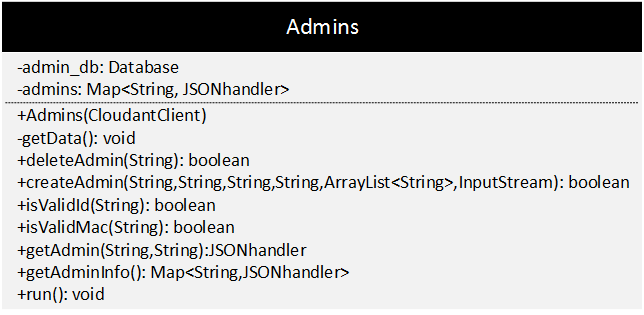


Figure 30: UML diagram for the Admins class which resides in the Model component

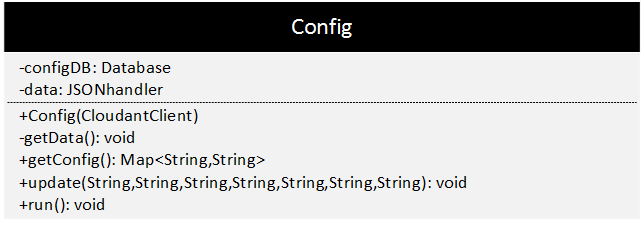


Figure 31: UML diagram for the Config class which resides in the Model component

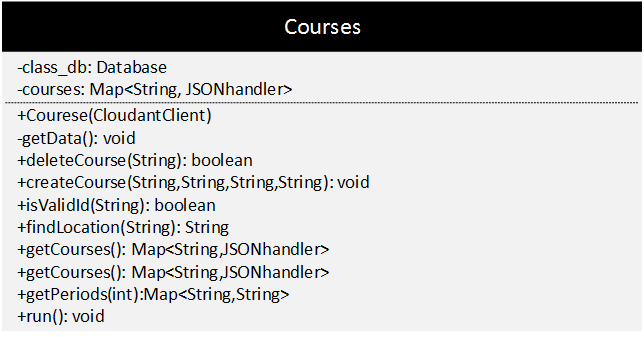


Figure 32: UML diagram for the Courses class which resides in the Model component

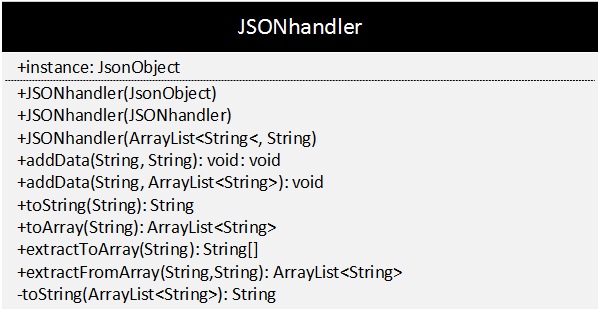


Figure 33: UML diagram for the JSONhandler class which resides in the Model component

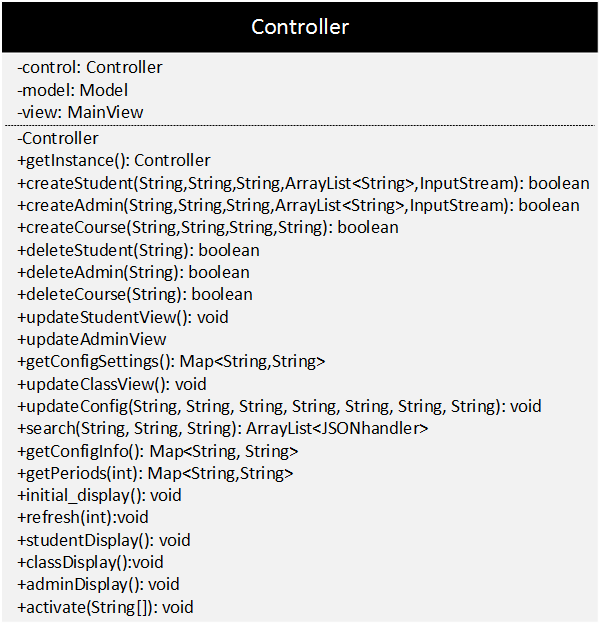


Figure 34: UML diagram for the Controller class which resides in the Controller component

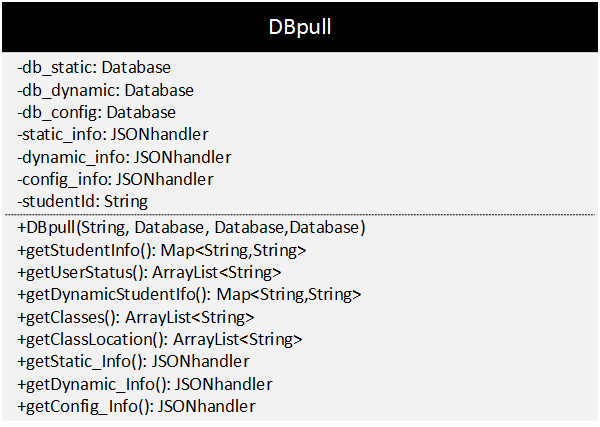


Figure 35: UML diagram for the DBpull class used in the Scanner Firmware

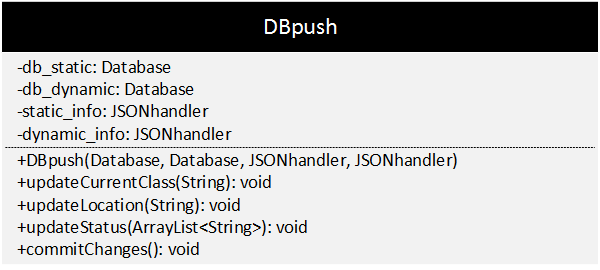


Figure 36: UML diagram for the DBpush class used in the Scanner Firmware

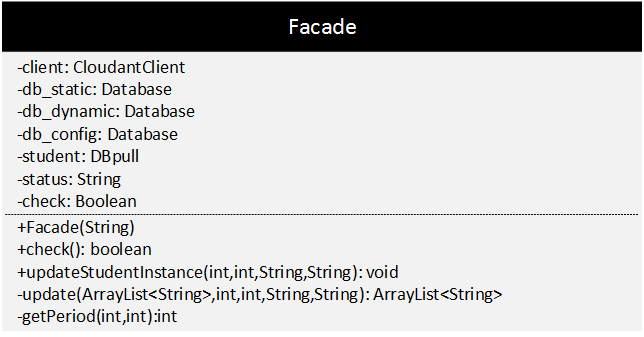


Figure 37: UML diagram for the Facade class used in the Scanner Firmware