Attendance Management System

Maan Alaulaqi *201610814*

A project report submitted in partial fulfillment of the

Requirements for the award of the degree of

Bachelors of Computer Science.

College of Engineering

Al Ain University of Science and Technology, Abu Dhabi, U. A. E

2019

Acknowledgement

Approval for Submission & Declaration

I certify that this project report, entitled “Attendance Management System”, prepared by Maan Alaulaqi, has met the required standard for submission in partial fulfillment of the requirements for the award of Bachelor of Computer Science, College of Engineering, at Al Ain University of Science and Technology, Abu Dhabi.

Approved by,

Signature : _____

Supervisor : Dr. Tarik El Amsy

Date : _____

Acknowledgement

First and foremost, I would like to thank God for giving us the knowledge and strength to work on this project. Without His guidance, I'd have been in more trouble.

I am grateful for the teaching staff of Alain University.

Patient friends, who had to deal with my talking to myself all night about different systems, implementation methods, and technical gibberish that they couldn't care less to remember or actually listen to.

And finally, the amazing, relentless, and forever fact checking community of the internet, especially Reddit and StackOverflow.

Thank you to all.

Abstract

Managing the attendance of students in educational institutes governs more importance than simply stating that they have made an appearance. It is a confirmation to both the instructors and parents that their student/child is in class when they're supposed to be. Something of this importance is highly prone to errors. The Attendance Management System (AMS) will address that by automating the logs of each students, their sick leave documents, and all other related aspects.

The AMS is, in its essence, a timekeeping software. It can communicate with databases of students and faculty. In every classroom there are readers that read the students' university ID cards. At command, it can generate reports for either the student, or the faculty members directly related to the student.

Table of Contents

List of Tables

List of Figures

List of abbreviations

AMS	Attendance Management System
APDU	Application Program Data Unit
API	Application Program Interface
GUI	Graphical User Interface
ISO	International Organization of Standardization
JDBC	Java Database Connectivity
Kbps	Kilobit Per Second
LSB	Least Significant Bit
MHz	Megahertz
MSB	Most Significant Bit
NFC	Near Field Communication
OS	Operating System
PC	Personal Computer
RFID	Radio-Frequency Identification
SQL	Structured Query Language
UC	Use Case
UID	Unique Identifier
USB	Universal Serial Bus
Wi-Fi	Wi-Fi Alliance – Tradegroup for promoting interoperability of IEEE 802.11 equipped devices

Introduction

Background

Radio Frequency Identification (RFID) technology is a wide spread technology that is currently being used worldwide by many companies and individuals. It can be used to keep track of things, such as packages and time, and can be used as access control and identification of different things, such as test tubes and animals, etc. RFID tags are read by RFID readers. Each tag consists of a serial number and a manufacturer's identifier, and optionally other information.

Near Field Communication (NFC) is a growing technology that has seen growth in the market. NFC takes RFID technology as the base, and extend it to be a more complete technology with increased security; NFC operates at 13.56 MHz, while RFID may operate on many different frequencies. As such, the range of operation for NFC tags is limited to a few centimeters.

At Alain University of Science and Technology, RFID is a part of the everyday life of the students and employees/faculty members. We use RFID cards in order to gain access to the main campus, and in some cases, access to the laboratories of the campus. There is potential for more use to come of our RFID enabled university ID cards. This projects aims to put some of that potential into the spotlight and possibly pave a way for more uses.

Problem definition

Most educational institutions are obliged to take the attendance of the students that have signed up to the instructors' courses. That in itself can take up some time due to the current nature of

taking the attendance manually using a list of the students, time varying based on the number of students in the course.

Managing the log of a few students is a relatively easy task; the instructor would need to simply find the current date from a log sheet, find the student's names, and mark them off with either "attended", "absent", "late", or "excused", and that would be it. The real issue is when we have over students that are above the average of a controlled number, when it reaches to 40 students or sometimes a 100 within a lecture hall. Each and every one of these students have to go through the roll and listing their names one by one, or have a paper that goes through the class to have every student sign for themselves. This process can easily take up to 10-15 minutes, and that is running on a good record. Some attendance taking can take up a solid amount of time from the classroom, which takes away from the academic's time where they could be teaching, whether it's an hour-long class, or a three-hour long lecture, that still a solid percentage of the class lost in efficiency due to attendance. To be a successful college begin by enforcing and ensuring that students have highest records of attendance will be directly proportional to making successful students into ensuring they fulfill their regular attendance to school.

This project proposes an automated system that will be able to record data according to attendance, record information needed over a periodic time through the different courses and duration of their studies as well creating statistics and reports that could be used as a source of evaluation for both the student and teacher. The proposed system is centralized, with different input devices in every classroom to take their logs. The most important part of the proposed solution is to create a database for analysis to precisely look at how an increase efficiency will positively affect the students and faculty in containing all the information that is required.

Purpose

The purpose of this project is to develop an application that will make it possible for students to mark their presence in classes. It has the potential to expand beyond the limitations of class, and be used in events and presentation, which would increase the usability of such an application.

The credibility of an attendance system can, in turn, be increased due the fact that the student will need to physically be there to mark their presence. Initially, the purpose was to also tackle the idea of buddy punching, but that was addressed by adding a delay in the application that will only allow one card at a time to be read.

Privacy of students will also be taken into account, as only the instructor will be able to view the attendance list.

Goals

The main goal of this project is to design and develop an Attendance Management System that utilizes RFID and NFC technology, while also expanding my own personal knowledge of these technologies. The project will, hopefully, pave potential pathways for other ideas to be explored by like-minded students.

Research Methodology

This project started off with a literature study and research in order to gain a solid understanding of the different aspects the application will need in order to operate. Prior to starting this project, my knowledge was strictly limited to basic java command line prompts and

a theoretical understanding of databases. The literature search primarily focused on understanding databases, and deep dive into RFID and NFC technology.

The development of the application was focused on incremental and iterative production or partial systems in order to test and understand them better. There was no solid, or established, methodology that was followed, simple incremental partial prototyping and testing.

Delimitations

The project and application will be developed and made for Windows (OS) computers. The idea to develop a mobile application was dropped in favor of saving time and that I don't have any prior experience in developing mobile applications.

The application will be developed using an NFC reader "ACR122U" that will the capabilities of reading both RFID and NFC tags. Testing of the application will be on a smaller scale to ensure functionality.

Background

This chapter is geared towards providing some background essential information about RFID, NFC, and other aspects that will be used on this application. In addition to that, some related works will be explored.

RFID

RFID is a technology based on communicating through electromagnetic signals (through coils) that enable identification and tracking of RFID tags. The RFID tags themselves can either be attached or embedded into different objects. In conjunction with RFID tags, RFID readers

enables for the automatic identification of objects, as opposed to manually scanning the object with using a barcode scanner. The RFID enabled objects need only be in close proximity, aligning the objects (such as aligning barcodes) is not needed. Different RFID systems have different frequencies at which operate, based on the nature of the transmission, and their needs. Most common frequencies operate at a low frequency (125-135 kHz) high frequency (13.56 MHz), and ultra-high frequency (433 and 858-930 MHz). [1][2]

NFC

Near Field Communication (NFC) us a communication technology that enables contactless sharing of data between devices. The communication range varies depending on the specifics of the card technology being used, generally, however, the working range is less than 10 cm with a maximum write/read speed of 424 kbps. Because of its short range, another party will find difficulty in intercepting the traffic, as they have to be very close to their target. This is helpful from a security's point of view, but for the purposes of this project, it will be helpful in preventing students to buddy punch since they'll need to be really close to the reader and near the instructor, who will be able to verify the integrity of the cards being read by the reader. NFC based systems operate at 13.56 MHz (high frequency RFID also operates in this frequency). [3]

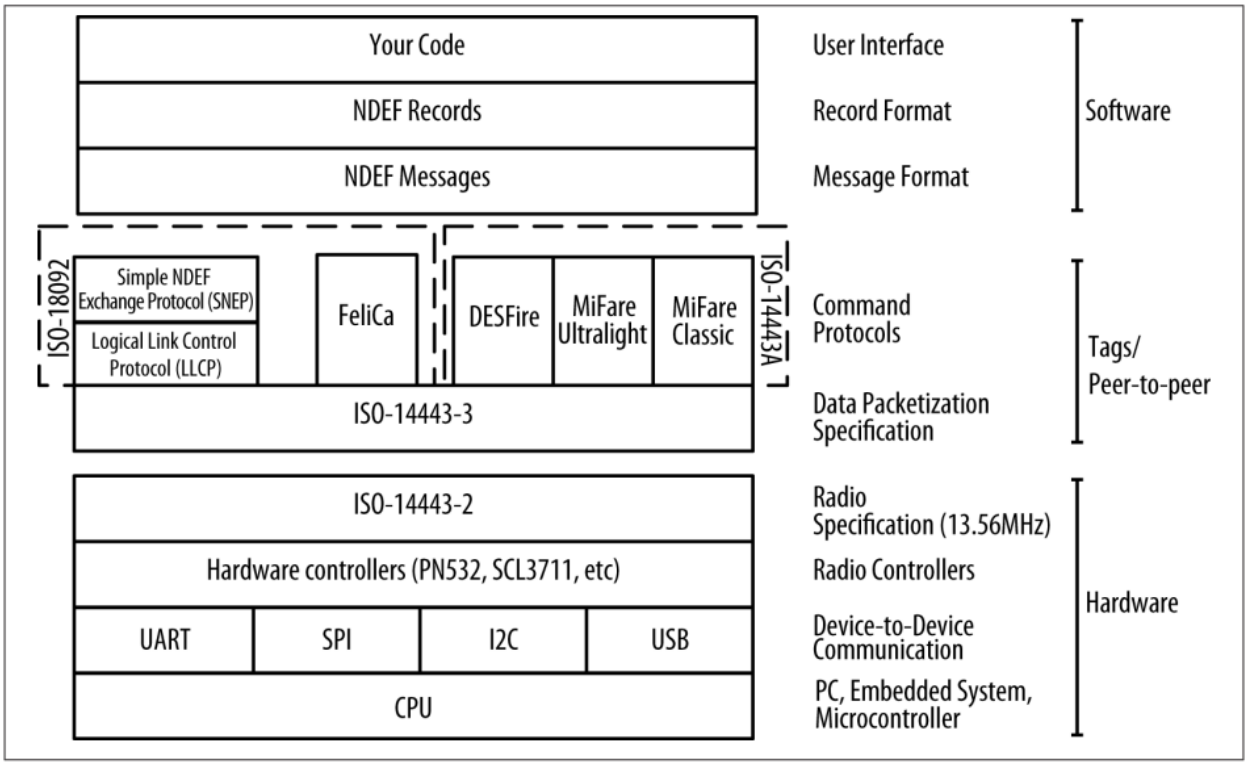

Figure 1 The NFC protocol stack [4]

Communication Modes

NFC devices have to modes of communication: active or passive. The first mode, the active mode, both NFC devices will charge their own radio frequency signal, allowing either device to initiated a transmission. In the second mode, the passive mode, the communication occurs between an active devices and a passive device, where the passive devices derives its power from the active device (the initiator of the communication) to charge its own radio frequency signals.

Comparisons

Since NFC technology is based on RFID technology, they both can operate at the same frequency (although RFID may operate in different frequencies depending on the specific

device). That aside, based on the architecture of the NFC (Fig. 1), NFC has two ISO 18092 based protocols; Logical Link Control Protocol, and Simple NDEF Exchange Protocol (SNEP). These two protocols allow NFC devices to perform two-way communication, in contrast to RFID, where they may only read or write data to tags.

JDBC

Java Database Connectivity (JDBC) is an API that allows code written in Java access to data that is stored in popular databases, such MySQL. JDBC allows the application to encode Structured Query Language (SQL) statements and pass them to the database. The results are returned through a similar interface. One of the advantages of using JDBC is that it utilizes the internet's file addressing scheme. When using JDBC to access a remote database, its SQL statement would look familiar to that of a URL (ie. "jdbc:odbc://www.BestCompany.com:1337/dbfile"). On a personal level, I used this interface mostly because it related to Java and it felt as though my application will be able to flow smoother with a familiar API such as JDBC.

Methodology

Research process

This section describes some of the steps taken to ensure that the application functions as intended with little to no faults/bugs. It will describe the environment in which this application was built on, and some of the steps taken during the development phase.

Information gathering

This phase of the research process was geared towards obtaining all the required information in order to develop this application. It involved reading different reports on similar topics written

by other people and getting inspiration by different ideas to utilize in this project. It also involved interacting with the StackOverflow(stackoverflow.com) community in order to clear up any doubts I have had during the development phase of the project.

Developing individual parts of the project

Opposed to the original plan of developing the whole project in one smooth sequence of coding and implementing, I had to resort to writing each module (class) separately, and testing them individually. The reason behind this was that there were different components to each module that I have little to no knowledge or experience (CardReader, dbConnect, etc.) with, and in order to be sure that the application functioned as intended each module was developed and tested on their own using test modules that utilized each module.

Data collecting

After the development of the project, I had a few friends outside of university test it and play around with the application. This was in order obtain any feedback or bugs that I might have missed. Since this application is of a minor and local nature, there was no actual need to have more than a couple of testers and their feedbacks.

Testing environment

The testing and development of this application will occur on a laptop computer running Microsoft's Windows 10. Since the application is written in Java, Java Runtime is needed. Besides the laptop, a USB NFC reader (ACR122U) is connected to it and will interact with, but not limited to, Mifare 1k test cards.

Hardware and Software specs

The hardware used in the development of the application are illustrated in table 1 and 2. The decision to use an ACR122U reader was due to its ability to read both RFID and NFC tags, and it was the most accessible to me at the time. The laptop environment is my own personal laptop.

Model/make	Lenovo ThinkPad P51
CPU	i7-7700HQ @ 2.81 GHz
RAM	32GB @ 2400 MHz
GPU	Intel® HD Graphics 630
HDD	ST1000LM035-1RK172 (1TB)
OS	Windows 10 ver. 1809 build 17763.292

Table 1 Laptop hardware specifications

Dimensions	98.0 mm (L) x 65.0 mm (W) x 12.8 mm (H)
Weight	70.0 g
Compliance / Certifications	ISO 18092, ISO 14443, PC/SC, CCID, EN60950/ISO 60950, CE, FCC, MIC, KC, VCCI, RoHS 2, USB Full Speed, Microsoft® WHQL
Interface	USB Full Speed
Operating Distance	Up to 50 mm (depends on the tag type)
Supply Voltage	Regulated 5V DC
Supply Current	200mA (operating); 50mA (standby); 100mA (normal)

Operating Temperature	0-60 °C
Operating Frequency	13.56 MHz
Smart Card Interface Support	ISO14443 Type A & B MIFARE FeliCa 4 types of NFC (ISO/IEC18092) tags
Operating System Support	Windows® Win CE 5.0 and 6.0 Linux® Mac OS® Android™ 3.1 and above

Table 2 ACR122U technical specifications [5]

Netbeans IDE	Java-written platform used for software development
Notepad++	A text editor geared towards to coding
Java SDK	A development environment for Java

Table 3 Software specifications

Design

This section will talk about some of the decisions I made with some of the components of the application, specifically why I chose to use Java and JDBC.

Java

Being a platform independent language, Java was a solid choice of a language to use for this application. It can be run on most, if not all, hardware and be implemented in most software platforms (Windows, macOS, Linux, etc.). Building a GUI is fairly simple due to the fact that Java has a lot of frameworks (AWT, Swing, etc.) and the frameworks provide easily accessible components, such as labels and buttons. These reasons were what drove me to develop the application and GUI using Java, as opposed to HTML/CSS on a web app.

Argumentatively, web apps can use Java and JavaScript, but my previous experience and knowledge are geared towards standalone Java applications, and thus I didn't opt to develop a web app. Furthermore, if in the future this application will become widely used, implementing it into a web app shouldn't be too difficult a task as Java and JavaScript are both accessible in HTML5.

JDBC

JDBC is a set of Java APIs used to access data in a database, such as MySQL. Other APIs, such as ODBC, are accessible, JDBC, however, was made for Java and is easily integrated into my application. The database schema can still be accessed by both APIs though, so it was more of a personal preference.

The Database

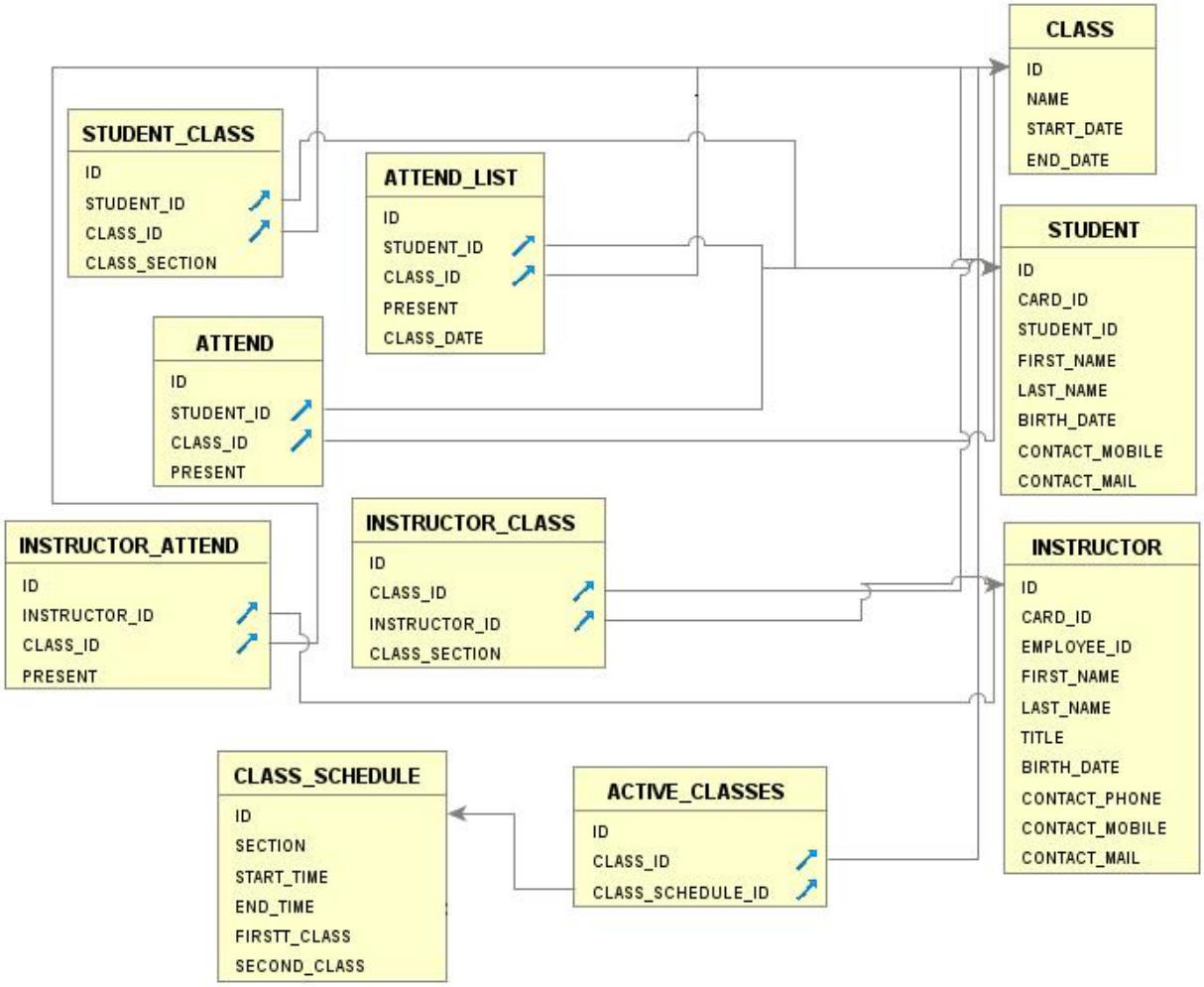

Figure 2 Main database schema

AMS will be running with the support of a Derby database. This was chosen for its ease of access from the NetBeans IDE. It allowed for quick referencing when playing with the data of the database.

Each table in the database references at least one of the three tables “CLASS”, STUDENT”, and “INSTRUCTOR”. These three tables are where most of the student, instructor, and class data will be stored.

The “ATTEND”, “INSTRUCTOR_ATTEND”, and “ATTEND_LIST” will store the attendance status of each student/instructor, “ATTEND_LIST” being the more permanent list where it will obtain its data from “ATTEND” and “INSTRUCTOR_ATTEND”.

Tables

The following are the list of tables used in the database.

class

Column name	Type	Properties
ID	INTEGER	PK
NAME	VARCHAR	
START_DATE	DATE	
END_DATE	DATE	

student

Column name	Type	Properties
ID	INTEGER	PK
CARD_ID	VARCHAR	
STUDENT_ID	VARCHAR	
FIRST_NAME	VARCHAR	
LAST_NAME	VARCHAR	
BIRTH_DATE	DATE	nullable
CONTACT_MOBILE	VARCHAR	nullable

CONTACT_MAIL	VARCHAR	nullable

instructor

Column name	Type	Properties
ID	INTEGER	PK
CARD_ID	VARCHAR	
EMPLOYEE_ID	VARCHAR	
FIRST_NAME	VARCHAR	
LAST_NAME	VARCHAR	
TITLE	VARCHAR	nullable
BIRTH_DATE	DATE	
CONTACT_PHONE	VARCHAR	nullable
CONTACT_MOBILE	VARCHAR	nullable
CONTACT_MAIL	VARCHAR	nullable

class_schedule

Column name	Type	Properties
ID	INTEGER	PK
SECTION	VARCHAR	nullable
START_TIME	TIME	nullable
END_TIME	TIME	nullable
FIRSTT_CLASS	VARCHAR	
SECOND_CLASS	VARCHAR	

attend

Column name	Type	Properties
ID	INTEGER	PK
STUDENT_ID	INTEGER	nullable
CLASS_ID	INTEGER	nullable
PRESENT	BOOLEAN	nullable

attend_list

Column name	Type	Properties
ID	INTEGER	PK
STUDENT_ID	INTEGER	nullable
CLASS_ID	INTEGER	nullable
PRESENT	BOOLEAN	nullable
CLASS_DATE	DATE	nullable

active_classes

Column name	Type	Properties
ID	INTEGER	PK
CLASS_ID	INTEGER	nullable
CLASS_SCHEDULE_ID	INTEGER	nullable

instructor_attend

Column name	Type	Properties
ID	INTEGER	PK
INSTRUCTOR_ID	INTEGER	nullable
CLASS_ID	INTEGER	nullable
PRESENT	BOOLEAN	

instructor_class

Column name	Type	Properties
ID	INTEGER	PK
CLASS_ID	INTEGER	nullable
INSTRUCTOR_ID	INTEGER	nullable
CLASS_SECTION	VARCHAR	nullable

student_class

Column name	Type	Properties
ID	INTEGER	PK
STUDENT_ID	INTEGER	nullable
CLASS_ID	INTEGER	nullable
CLASS_SECTION	VARCHAR	nullable

```java
/**
 * Connects to the database.
 * @param username = "test"
 * @param password = "test"
 * @return  con to carry out the connection to other methods
 */
public static Connection doConnect(){
        try{
            //Connect to the db
            String host = "jdbc:derby://localhost:1527/ams_test5/";
            String username = "test";
            String password = "test";
            Connection con = DriverManager.getConnection( host, username, password );
            //System.out.println(con.getSchema());

            return con;
        }catch (SQLException error){
            System.out.println("Oopsies, you gone get goofed, here's the error message: ");
            System.out.println(error.getMessage());
            return con;
        }

    }
/**
 * Closes the connection in the agreed upon safe pattern:
 * ResultSet > Statement > Connection.
 * Should generally be added to the finally block after the try-catch.
 * Based on ( https://stackoverflow.com/questions/2225221/closing-database-connections-in-java )
 */
public static void doClose(){
    try { rs.close(); } catch (Exception e) { /* ignored */ }
    try { COMMANDTHEE.close(); } catch (Exception e) { /* ignored */ }
    try { con.close(); } catch (Exception e) { /* ignored */ }
```

Figure 3 Early code snippet: How JDBC can be used to set up a connection to a database (locally in this snippet)

In an early phase of the application, I set up a local database using JDBC and a database interface called Derby through NetBeans. Displayed here are two methods, a “doConnect()” and a “doClose()” method. They both open and close a connection to a database. These two methods will surround any SQL command in order to close the connection after each command, as opposed to leaving the connection open, which may lead to a loss of resources and a malfunction.

Useful diagrams

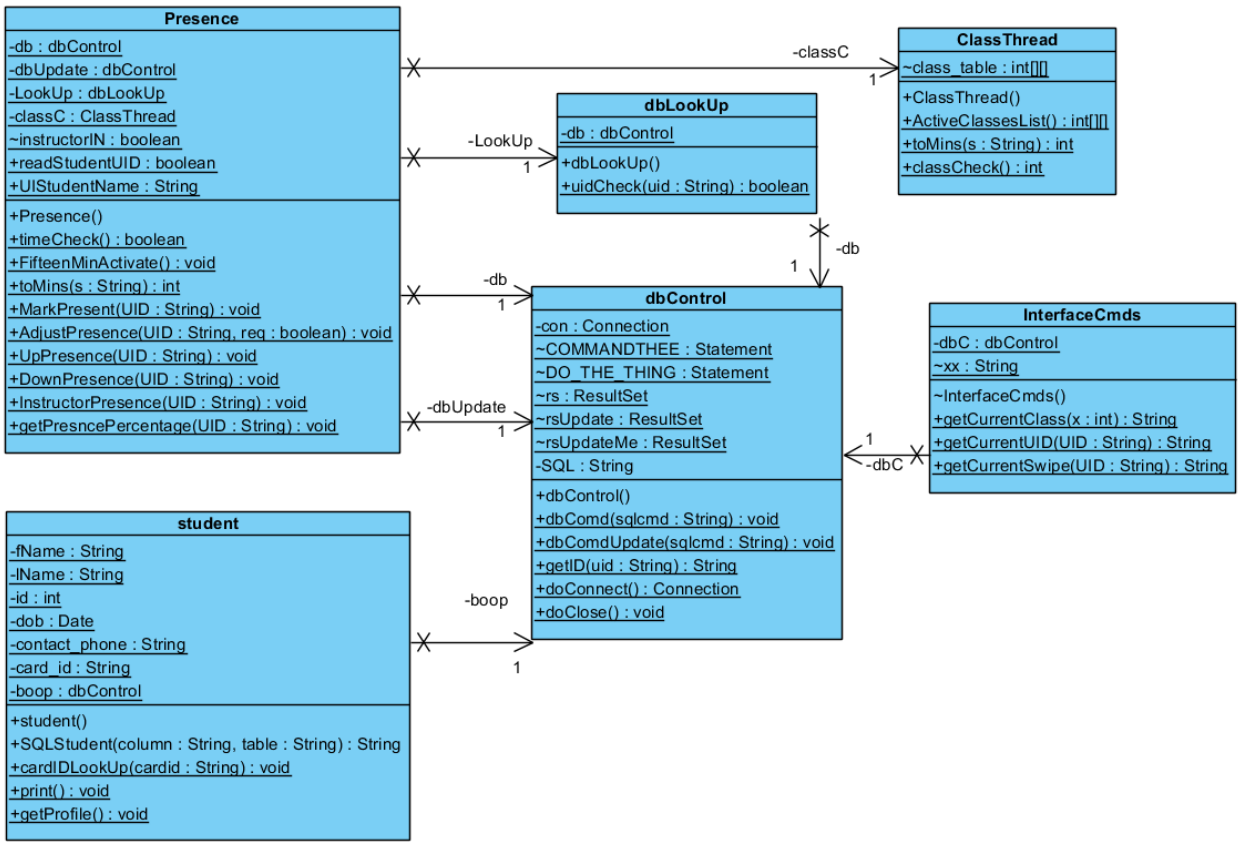

Figure 4 Class diagram of classes with relations

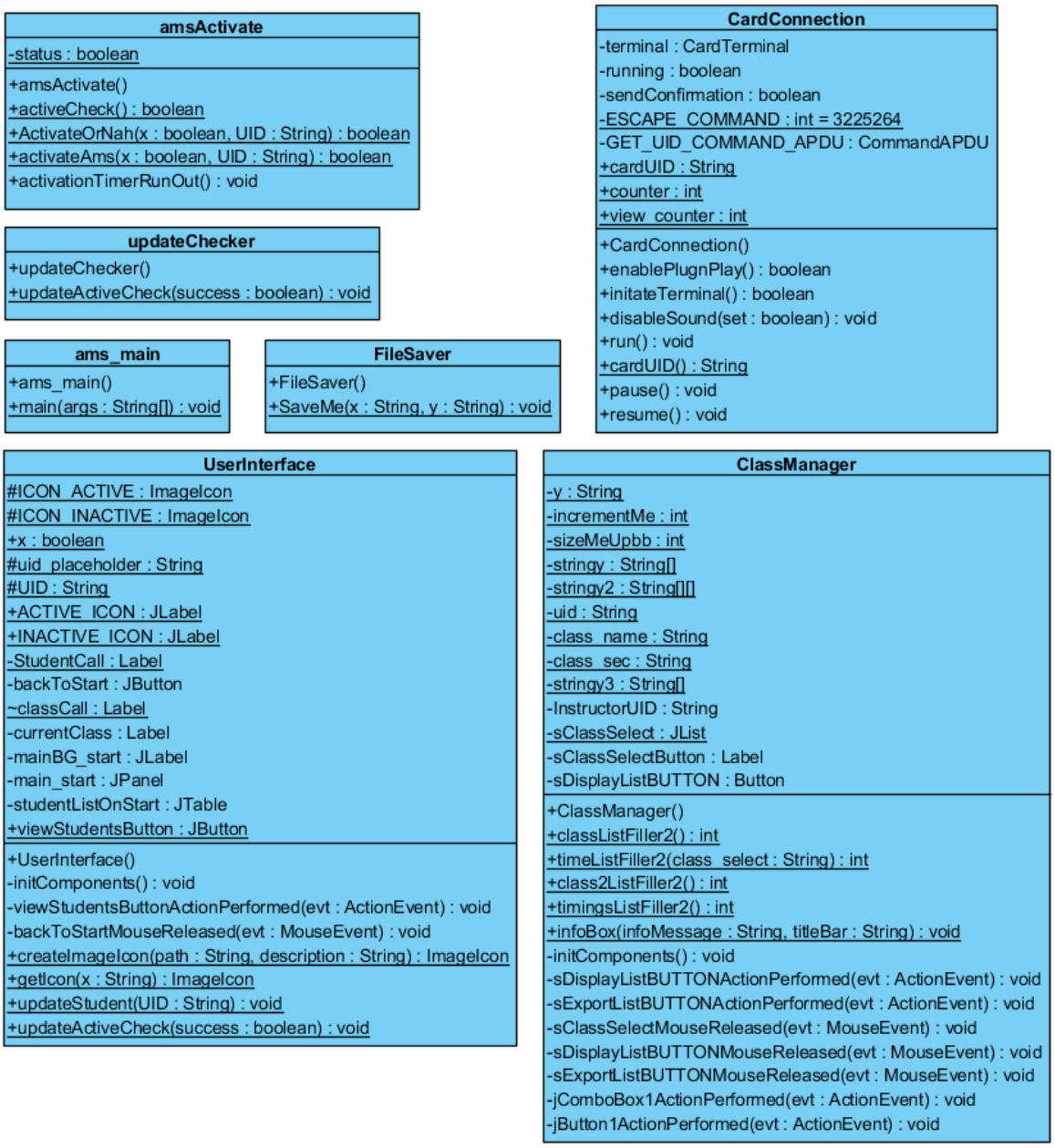

Figure 5 Class diagrams of independent classes

Functional requirements

Reading a card

This was achieved using the library "javax.smartcardio.*".This library consists of methods that allows the application to detect and read cards, and in the application, the Unique Identifier (UID) of the cards. Each card consists of a UID and a manufacturer's serial number that identifies them. These fields are not supposed to be rewritable. The application will read the UID of the cards and look them up in the database to locate the individual that UID is assigned to.

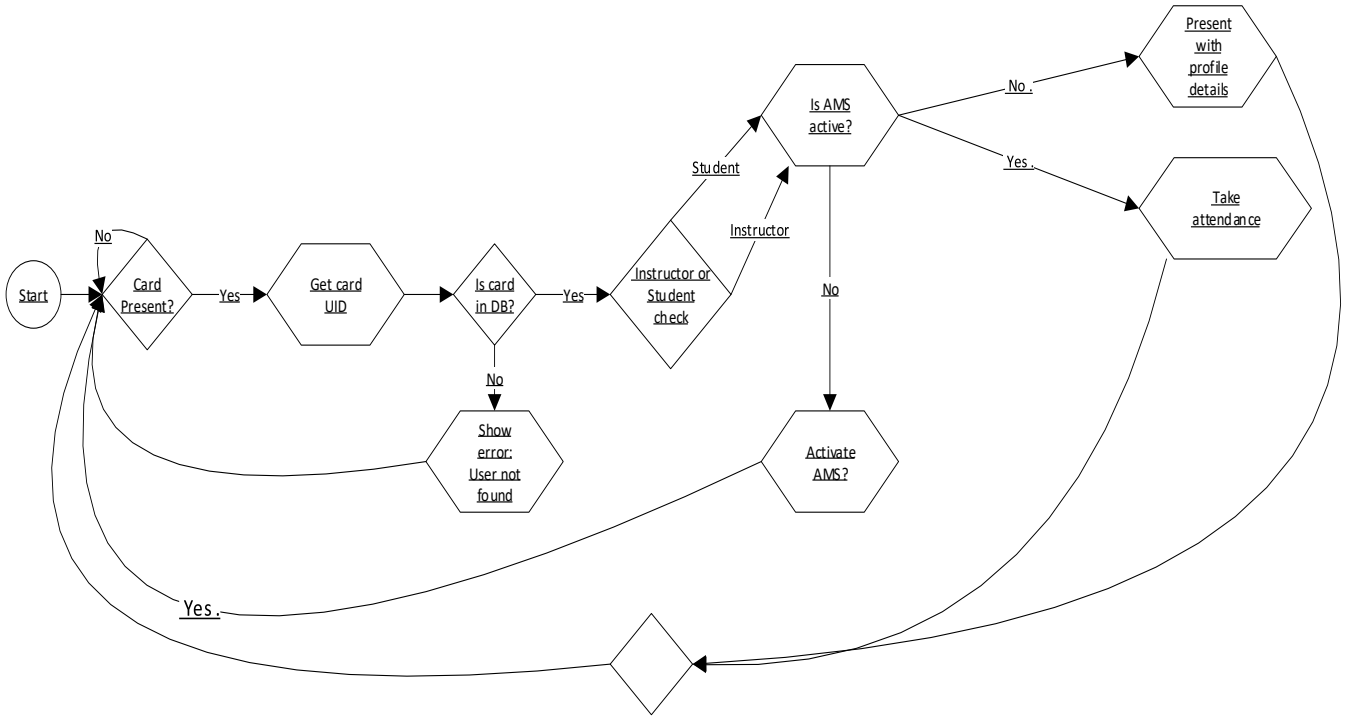

Figure 6 Flowchart showing the main flow of actions of the application

Figure 6 illustrates a common flow of actions and reactions that will occur during the life cycle of the AMS. At the end of all the activities the "cursor" goes back to checking for new cards.

This function will be achieved using the Thread method of Java. (See figure 4 below). Shown here is also "Thread.sleep(500)", which allows the card reader to take in ONE card every half a second.

What I've learned about Threads may admittedly be incomplete, however, it has helped me acquire the desired results and functionality. Further research and studies of this method will be applied to completely understand the concept of threading in Java.

```
public class CardHandlerTest{
    public static void main(String[] args){
        CardHandler cl = new CardHandler();
        cl.enableHotplug();
        cl.initateTerminal();
        cl.disableSound(true);
        new Thread(cl).start(); //This is to keep the reader running
    }
}
```

```
        // Main-loop to keep the thread running.
        while(true){
            while(!running) {
                try {
                    Thread.sleep(500);
                } catch (InterruptedException ex) {
                }
            }
            while(running){
                try{
                    // Checks if a card is present, if not it'll sleep and th
                    if(!terminal.isCardPresent()){
                        Thread.sleep(500);
                    }
```

Figure 7 Earlier snippet of code showcasing the Thread in java

As the scope of this project is mainly geared towards implementing an attendance management system using card scanners, some common functionalities won't be available. Such functionalities may include, but not limited to course registration of students. This functionality can be implemented, but it wouldn't serve this particular application much. The idea of course registration would require that the student apply as a new student in a certain

course, which in this case is as simple as adding their name to a table in the database. The cases in which this functionality would be useful would be in cases where certain questions need to be answered. Some of these questions are:

- Is the student clear to be admitted into a particular course? As in, are they currently registered to the course

These questions can be answered with the registration's own system, but not with the AMS, whose current sole purpose is to simply take attendance of the already registered students.

View attendance list

One function of the AMS is the ability to view the attendance, in the student's case, their own attendance, and in the instructor's case, a list of the students and their attendance records.

Development

The development of this application was done almost entirely in NetBeans. Most of the testing and iterations occurred in Notepad++. The reasoning behind this is that although NetBeans is a very powerful tool in the programmer's hands, I've grown accustomed to the nature of Notepad++ and its simplicity in coding. I am fully aware that I do not use NetBeans to its full potential, but it's as the saying goes: you don't need a full featured toolbox for a job that a Phillip's screw can accomplish with a little bit of imagination.

Some afterthoughts regarding the development of this application: I wish I had invested more into the idea of GitHub. I haven't been able to figure it out completely and only by accident was I able to "use it". Had I become accustomed to using it I would have saved a lot of time due to HDD crashes and loss of data.

During development, a high emphasis was placed in the Java library known as “SmartCardIO”, which is the smartcard API that allows the communication of the card reader with the Java program. This library enabled the application to be programed and allow it to connect and communicate with the ACR122U card reader and use the APDU commands found in its API.

```
public void run() {
    Card card;
    CardChannel channel;
    ResponseAPDU answer;
    byte[] uid;
    while(true){
        while(!running) {
            try {
                Thread.sleep(500);
            } catch (InterruptedException ex) {
            }
        }
        while(running){
            try{
                // Checks if a card is present, if not it'll sleep and then check again.
                if(!terminal.isCardPresent()){
                    Thread.sleep(500);
                }
                else{
                    // Connects to the card
                    card = terminal.connect("*");
                    // Opens a channel
                    channel = card.getBasicChannel();
                    // Transmits the get UID command
                    answer = channel.transmit(GET_UID_COMMAND_APDU);
                    // Takes the response and puts it in a byte array.
                    uid = answer.getData();

                    if(Integer.toHexString(answer.getSW()).equals("9000")) { //90 00 = success, 63 00 = error
                        String out = "";
                        // Gets the UID from the array. MSB at [3] and LSB at [0]
                        for(int i = 3; i > -1; i--) {
                            //The next line is based partially on https://stackoverflow.com/questions/11380062,
                            //tl;dr = the "& 0xFF" prevents the result being something like fffffff8, and keep
                            out += Integer.toHexString((int) uid[i] & 0xFF);
                            System.out.print(" " + Integer.toHexString((int) uid[i] & 0xFF));
                        }System.out.println();
                        System.out.println(out);
                        // Checks if the UID is in our database
                        //TO-DO Add comparator of UID and DB entries(student card id)
```

```
// Gets the UID from the array. MSB at [3] and LSB at [0]
for(int i = 3; i > -1; i--) {
    //The next line is based partially on https://stackoverflow.com/q
    //tl;dr = the "& 0xFF" prevents the result being something like f
    out += Integer.toHexString((int) uid[i] & 0xFF);
    System.out.print(" " + Integer.toHexString((int) uid[i] & 0xFF));
}System.out.println();
```

Figure 8 Snippet of the SmartCardIO implementation

The reason the for loop looks the way it does is because of the Response APDU. table 4 illustrates the structure of the response when a Command APDU is sent to get the UIS of the currently connect card, SW1 and SW2 is the status code. This is adapted from the ACR122U manual. The least significant bit (LSB) is first and the most significant bit (MSB) is last, therefore in order for the UID to be read in its correct format, it needs to be read backwards. This is not a general rule, it depends on the specific reader and what its specification imply.

Response	**Data Out (in bytes 0x00-0xFF)**					
Result	UID (LSB)	-	-	UID (MSB)	SW1 (Status code)	SW2 (Status code)

Table 4 Structure of Response

Main start/run

The AMS is designed to run on a thread that reiterates itself every half a minute. A thread is basically a way to allow code to re-run according to a set time rate. Figure 9 showcases some of the function calls, and how they're threaded to one another, that occurs in this thread.

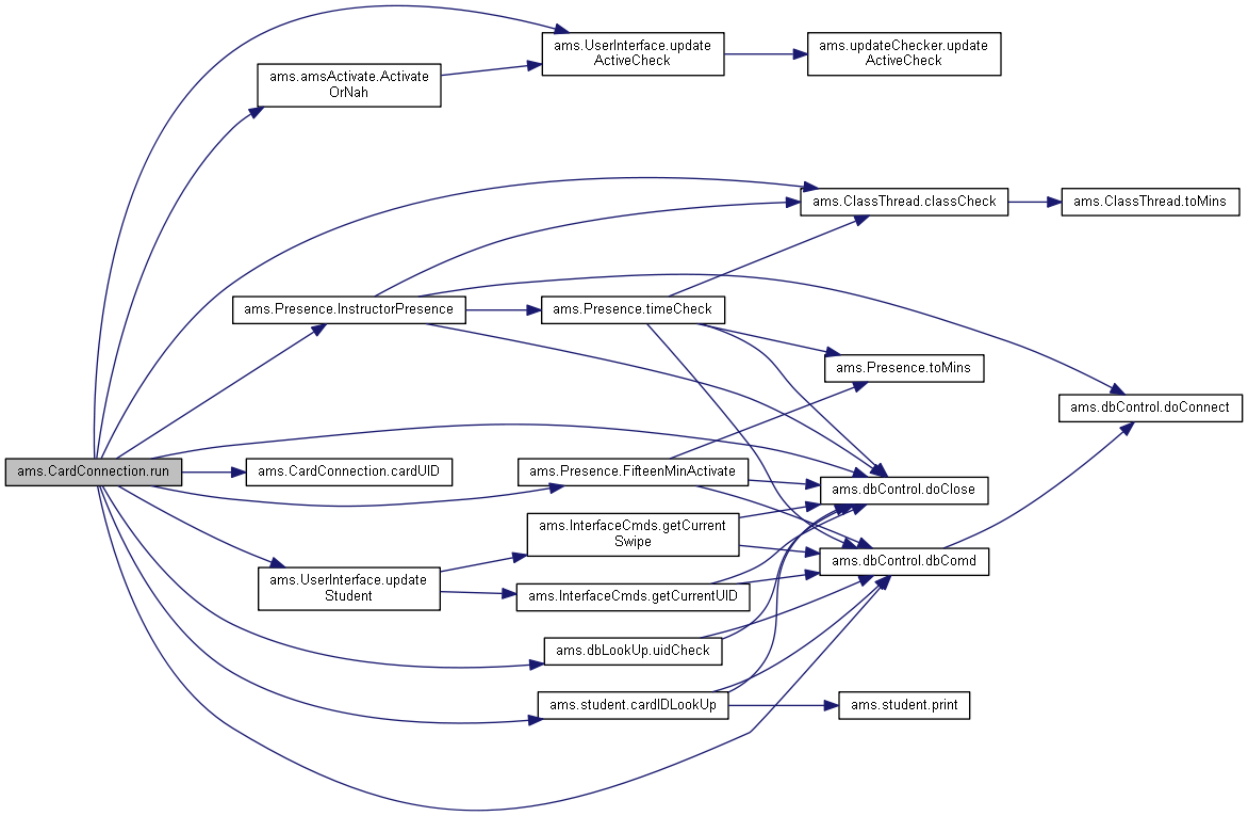

Figure 9 Some of the main calls running on loop

More details of the threads can be found in the Javadoc folder associated with this report.

Testing

Testing was done in two main parts: testing of individual modules (classes), and testing of the modules together.

Testing individual modules

When developing the application, often time I needed to make a copy of the module and paste it on to Notepad++ and create a separate main class in order to test their individual functionalities. The introduction of different forms of modules (JDBC modules and SmartCardIO modules) needed to be met with strict prejudice in terms of testing their functionalities. These are environments that I have had no prior experience or knowledge with, so I needed to be extra thorough in my testing.

Testing modules together

Testing the modules together proved to be somewhat of a challenge as there were different parts at play at this stage, such as the database running while the card reader is running. A few things could've gone wrong, which they did in the early stages of testing.

Testing: NetBeans vs. Notepad++

My Computer Science career began with notepad.exe that's included in Microsoft's Windows OS. From there I went into Notepad++ and remained there for years. It was only in the summer 2018 that I started using NetBeans for a graphics course. All my testing happened in code, happened with "System.out.println"s. I've personally found that this method works best for me as I have grown accustomed to it. NetBean's debug service is a powerful tool for testing, but it isn't a tool I fully understand, and I had no interest in understanding at this time since my current method of testing works fine. "If it ain't broke, don't fix it".

Final prototype

Figure 10 "Home screen" of the AMS

The way the AMS functions is that it is a standalone point of access for students and instructors. Minimal interaction with the system is intended. The less interaction the smoother the interactions are. The idea here is not to create a system that manages any courses or student profiles. Its *main* purpose is to log in the attendance of the students for that class on that particular day.

With that said, some functionalities have been added at the request of my supervisor. The function to allow the instructor to view their current student list for any of the courses that particular instructor has. (Figure 8 and 9)

Figure 11 Once an instructor swipes their card, the AMS is activated and the "View students" button is also available to them.

Shown in figure 9 is the window the instructor is greeted with once they select “View Students”. The numbered sections are as follows

1. The instructor will be able to view all the courses they are assigned to for that particular semester. If they wish to view the students, they’ll need to select one of the courses in the list

a. In the case that they are assigned multiple sections, they may select it from this table

2. The "Display student list" will list all the students in that particular course (and section) on the table located on the right half of the window.
3. The list of currently enrolled students.
4. If they so choose, they may export the student list as an Excel (.xls) spreadsheet.

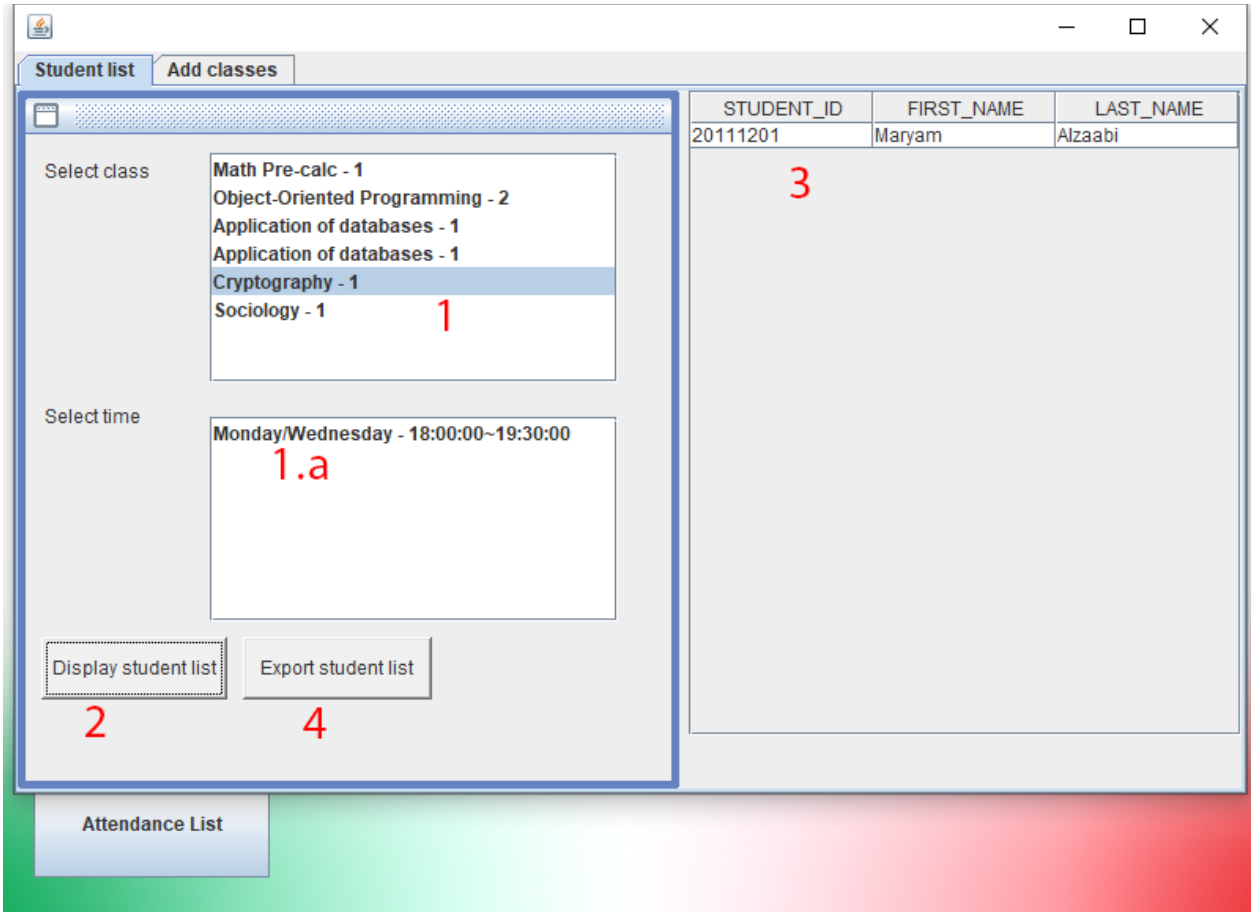

Figure 12 The "View Students" window

Going back to the home screen, whether a student/instructor or not, the attendance list of the current class will be available by selecting the "Attendance List" button, at which point the user will be able to view the current student list, and their attendance status for that particular class on that particular day. (Figure 10)

Figure 13 Attendance list of a particular day and time

Conclusion

Going through with this project, and while developing the application, I've noticed that there were features I might have been able to add, such as emailing a list of attendees, or sending the list to a printer to have it printed out. However these functions could be patched in at a later time if this application gains traction.

However, while loitering through this project, through different libraries, different reports, a vast sea of various of media geared towards outputting information, I was able to obtain a few things:

- My knowledge and understanding of the Java has drastically increased. Before this project, I haven't written much code except for class assignments, so this change in environment really helped delve deeper into the Java environment.
- My understanding for databases has definitely become more than just theoretical. I'm hoping to possibly utilize this experience in future projects.
- I have learned how to effectively use different libraries and incorporate them into my coding.
- Although I feel I only scratched the surface, I've gained some knowledge regarding the infrastructure of RFID technology. I hope in the future I'm able to go deeper into that world and understand it more, and be able to quickly implement code that will effectively utilize RFID technology.

At the end of it all, I was able to delve into aspects of software development where I learned many different aspects of it, from failed features to successful tests. It was a lovely journey.

References

[1] J. Landt, "The history of RFID," IEEE Potentials, vol. 24, no. 4, pp. 8–11, Oct. 2005.

[2] Notes, Electronics. "RFID Frequency Bands & Spectrum." Electronics Notes, www.electronics-notes.com/articles/connectivity/rfid-radio-frequency-identification/frequency-bands-spectrum.php.

[3] Patauner, C., et al. "High Speed RFID/NFC at the Frequency of 13.56 MHz." EURASIP Journal on Advances in Signal Processing, 2007, www.eurasip.org/.

[4] "The Architecture of NFC." Beginning NFC near Field Communication with Arduino, Android, and PhoneGap, by Tom Igoe et al., O'Reilly, 2014, pp. 15–16.

[5] Advanced Card Systems Ltd., "ACR122U USB NFC Reader," Advanced Card Systems Ltd. [Online]. Available: http://www.acs.com.hk/en/products/3/acr122u-usb-nfc-reader/

Appendix A: Source code

This appendix provides a listing of the java files and dependency library files that must be associated with this document.

Java files:

- ams_main.java
- amsActivate.java
- CardConnection.java
- ClassManager.java
- ClassThread.java
- dbControl.java
- dbLookUp.java
- FileSaver.java
- InterfaceCmds.java
- Presence.java
- student.java
- updateChecker.java
- UserInterface.java

Library dependencies

- AbsoluteLayout.jar

- beansbinding-1.2.1.jar
- derbyclient.jar
- eclipselink.jar
- javax.persistence_2.1.0.v201304241213.jar
- org.eclipse.persistence.jpa.jpql_2.5.2.v20140319-9ad6abd.jar
- rs2xml.jar