DEGREE PROJECT IN COMMUNICATION SYSTEMS, FIRST LEVEL
STOCKHOLM, SWEDEN 2015

Automating Interactions with Web Services

NFC based attendance software in Java

CARL JOHANSSON & SOREN KAVOSI

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Automating Interactions with Web Services

NFC based attendance software in Java

Carl Johansson & Soren Kavosi

2015-06-02

Bachelor's Thesis

Examiner and Academic adviser
Gerald Q. Maguire Jr.

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

Today we use an obsolete way of handling information regarding which student and/or teacher is attending which class/lab/seminar, attendance is written down on a piece of paper and collected so that an administrator can manually enter this information to some data processing system.

This method is far from optimal and demands a lot of time and resources from administrators, teachers, and students. Correct gathering of attendance is important since it is required for specific parts of some courses. We propose to automate the collection of this attendance data, thus enabling students and teachers to simply swipe their NFC-enabled KTH access card in order to enter their name on an attendance list. This will be achieved by creating an application that adds a student to an attendance list by reading information using a NFC/RFID reader and mapping the card's UID to a KTHID (a locally unique identifier used within the university) using a database. The resulting attendance list should be formatted in such a fashion that it can easily be uploaded to systems such as KTH Social and Daisy. Ideally these systems will be extended so that instructors/teachers can use this attendance list to automatically create the appropriate entries in these systems to record the student's participation in the indicated activity – in the process avoiding a lot of manual labor and improving the accuracy of the process.

An additional problem is that there is currently no unified system that connects the KTH access card database (BRAVIDA) to the KTH LDAP database (which stores information about KTH students, faculty, and staff). This means that each student's access card UID must manually be added to a database together with the student's KTHID. However, once this database entry has been made, we can then map from a card number to a KTHID (or the reverse).

The purpose behind and expected result of this thesis is a functional prototype of an application that creates an attendance list by reading data from the student or teacher's access cards using a NFC reader. This will hopefully stimulate further digitalization in KTH and also encourage more courses to utilize such access card based attendance lists. The result should be less manual effort by students, faculty, and staff, as well as more accurate and timely filing of attendance information for courses.

Keywords

student attendance, application, access card, NFC reader, prototype

Sammanfattning

I dagsläget använder vi en föråldrad metod för att hantera information kring vilken student och/eller instruktör som närvarar vid vilken föreläsning/laboration/seminarie, närvaron skrivs ner på en bit papper som samlas ihop och skickas till en administratör som sedan manuellt får mata in den här informationen i de olika databehandlingssystemen.

Denna metod är långtifrån optimal och kräver en massa tid och resurser från administratörer, lärare och elever. Att den insamlade informationen är korrekt är viktig eftersom den är ett krav vid vissa kurser. Vårt förslag är att insamlingen av närvaroinformation automatiseras, genom att studenter och lärare enkelt kan dra sina KTH access kort för att mata in sitt namn på en närvarolista. Detta kommer att genomföras genom utvecklandet av en applikation som lägger till en student i närvarolistan genom att läsa av kort genom en NFC/RFID läsare och mappning av kortens UID till ett KTH användarnamn (användarnamnet är unikt inom KTH) med hjälp av en databas. Närvarolistan som genereras som ett resultat av programmets körning skall vara formaterad på ett sådant sätt att den enkelt kan laddas upp till system som KTH Social och Daisy. Idealt skall applikationen vidareutvecklas så att instruktörer/lärare kan använda närvarolistan till att automatiskt lägga till rätt post i de systemen för att lagra information om studentens närvaro vid en viss aktivitet - med mål att undvika mycket manuell inmatning samt öka noggrannheten kring processen.

Ett ytterligare problem är att det i nuläget inte finns något system som kopplar KTH:s databas för accesskort (BRAVIDA) till KTH LDAP databasen (som lagrar information om KTH studenter, fakultet och personal). Detta betyder att varje användares accesskorts UID måste läggas till i en databas manuellt tillsammans med studentens KTH användarnamn. Emellertid är det så att när posten väl är inlagd i databasen, så kan vi mappa mellan accesskorts UID till KTHID(eller motsatsen).

Detta examensarbete har resulterat i en fungerande prototyp av en applikation som skapar närvarolistor genom att läsa av data från studenter och lärares accesskort med hjälp av en NFC läsare. Detta kommer förhoppningsvis att stimulera ökad digitalisering inom KTH och dessutom motivera fler kursansvariga att använda accesskortsbaserade listor. Resultatet bör förhoppningsvis bli mindre manuellt arbete för studenter, fakultet och övrig personal samt mera precis och snabbare insamling av närvaroinformation.

Nyckelord

studentnärvaro, applikation, accesskort, NFC-läsare, prototyp

Acknowledgments

We would like to thank O'Reilly Media Inc. for allowing us to use a figure from the book: Beginning NFC (ISBN-13: 978-1-4493-0851-3).

We would like to thank Victor Häggqvist and Peter Lundberg for their valuable input during the development of this prototype.

We would like to thank those who allowed us to use their KTH usernames during the testing of the prototype.

Stockholm, June 2015
Carl Johansson & Soren Kavosi

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

APDU	Application Program Data Unit
CN	Common Name
DC	Domain Component
DIT	Directory Information Tree
DN	Distinguished Name
GUI	Graphical User Interface
ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
Kbps	Kilobit per second
KTH	Kungliga Tekniska Högskolan
LDAP	Lightweight Directory Access Protocol
LLCP	Logical Link Control Protocol
LSB	Least Significant Bit
MHz	Megahertz
MSB	Most Significant Bit
MVC	Model-View-Controller
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
OS	Operating System
OU	Organizational Unit
PC	Personal Computer
POS	Point-of-Sale
PUL	Personuppgiftslagen - The Personal Data Act
RFID	Radio Frequency Identification
SL	Storstockholms Lokaltrafik AB - Stockholm Public Transport
SNEP	Simple NDEF Exchange Protocol
UID	Unique Identifier
USB	Universal Serial Bus
Wi-Fi	Wi-Fi Alliance – tradegroup for promoting interoperability of IEEE 802.11 equipped devices
WWW	World Wide Web

1 Introduction

This chapter introduces the background and problem definition that motivates the execution of this thesis. Also, it explains the goals, research methodology, delimitations and structure of this thesis.

1.1 Background

Radio frequency identification (RFID) technology is a widely used technology and has been used for many years for tracking packages, access control, identification of animals, etc. RFID tags are read by RFID readers. These tags have both a manufacturer's identifier and a serial number, but can also store other information. Further details about RFID are given in Section 2.1.

In contrast, Near Field Communication (NFC) is an emerging technology with an expanding market. NFC takes RFID technology as a base, then extends it to be a more complete communications technology and to increase security it reduces the range at which this communication may take places to several centimeters. Additionally, RFID technology can use one of many different frequencies, while NFC operates at 13.56 MHz. According to the research firm Berg Insight, the annual growth rate of NFC-enabled mobile handsets will be about 50 % by year 2017 [1] and nearly 30 % by year 2019 for NFC-enabled Point-of-Sale (POS) terminals [2]. This, in turn, will increase the demand for NFC applications. Further details about NFC are given in Section 2.2.

There is currently a broad spectrum of uses in society when it comes to NFC. One example is access control, such as implemented in ticketing systems for public transportation and locking/unlocking doors in buildings. Another example is simplified payment systems where smart phones implementing NFC can be used for secure transactions. Google has made progress in NFC payments with an application called Google Wallet [3] which serves as a digital wallet where credit cards, coupons, and such are stored digitally.

At KTH School of Information and Communication Technology (ICT), RFID is a part of everyday life for students and teachers due to the fact that it is implemented in the access control system used throughout the university for locking/unlocking doors. All students, staff, and faculty possess a unique RFID equipped access card of the type Mifare Plus 4k [4], which is currently solely used for locking/unlocking doors. However, these cards have the potential for more than just locking/unlocking doors. This thesis project aims to automate the process of registering attendance information. This problem is discussed more thoroughly in Section 1.2.

Figure 1-1 is a system diagram that illustrates the relationship between the different elements of the application proposed in this thesis project as a solution to the problem of registering attendance at events at the university. The system is based on the RFID cards (in this case a Mifare Plus 4k card – further described in Section 2.2), a NFC reader which is capable of reading the card number contained in the RFID card, an instructor’s PC running our application, access to the existing KTH LDAP database, access to our new database, and access from the instructor’s PC to one of the KTH systems that is used to keep track of attendance.

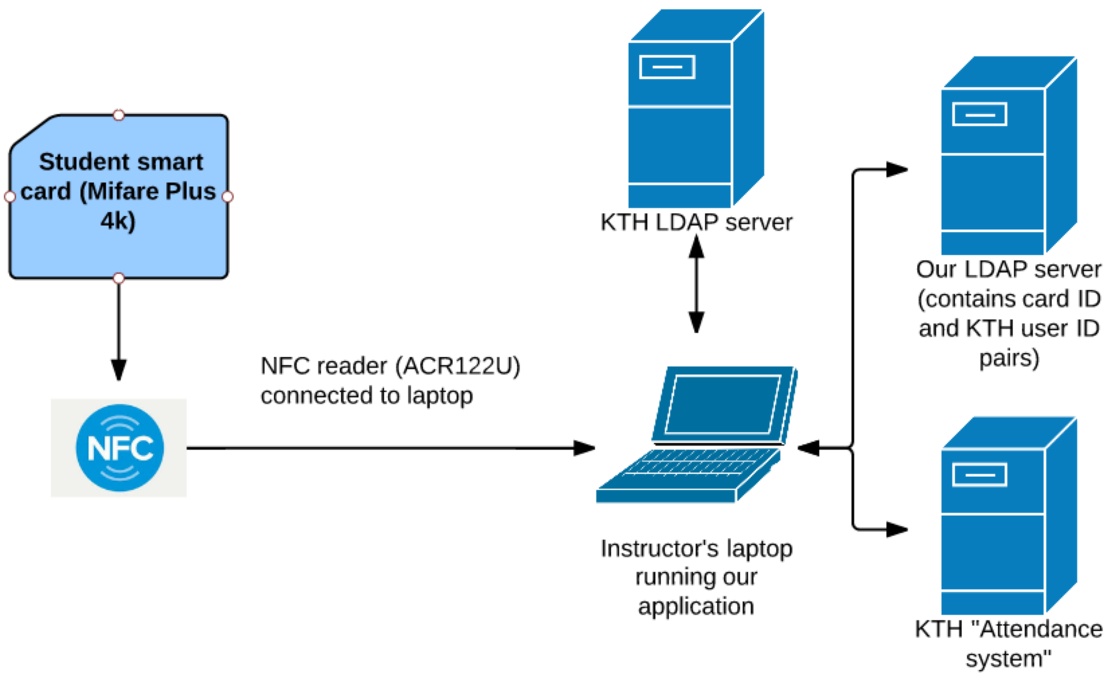

Student smart card (Mifare Plus 4k)
NFC
NFC reader (ACR122U) connected to laptop
KTH LDAP server
Instructor's laptop running our application
Our LDAP server (contains card ID and KTH user ID pairs)
KTH "Attendance system"

Figure 1-1: System diagram of the attendance registration system

1.2 Problem definition

In KTH, there is currently no effective way of registering attendance at classes, labs, and such; as this process is done manually using pieces of paper[*]. Students are usually required to write their signature next to their name on a pre-printed attendance list. This method is far from optimal with regard to accuracy, trustworthiness, privacy, and amount of labor. Forged signatures indicating that a student has been present, when in fact they have not, is always a risk when passing around an attendance list in class. Furthermore, as emphasized by the recent decision [5] by Ander Lundgren, KTH's University Director, in his "beslut V-2015-0309: Vidarebefordrar beslut 'Anvisning om hantering av förteckningar över studenter' – there are also privacy and personal integrity issues with regard to lists of students. Gathering correct and accurate information about attendance is important since attendance is required to receive credit for parts of some courses. Accuracy is a problem both due to clerical errors in reading the lists after they have been signed, but also due to missing names that have been manually added to the list – as these entries are frequently unreadable, incomplete, or not unique.

Another issue regarding the current handling of attendances lists in KTH is the amount of administrative work that needs to be done. Paper attendance lists are typically passed on to an administrator who needs to manually enter the information from the list into data processing systems such as Daisy, KTH Social, and Bilda[†].

[*] Information from professor Gerald Q. Maguire Jr.
[†] These are three of the many systems used at KTH for course administration.

There are several problems relevant to the design and implementation of the proposed application, namely:

- How to connect the KTH access card database (BRAVIDA) [4] to the KTH LDAP database. Or if they are not connected, how to generate a new database that links an access card number to a KTHID (a locally unique identifier used within the university).
- How and when will the client and the server interact (e.g., what protocols should be used for: (a) entering new users, (b) entering attendance information, and (c) providing the collected data to some other data processing system). The last of these protocols concerns how to make the attendance information available to systems such as KTH Social and Daisy.
- User friendliness (how to inform each user that their attendance has been recorded and that the mapping has been done correctly/incorrectly).

1.3 Purpose

The purpose of this degree project is to develop an application that makes it possible to create digital attendance lists for occasions such as seminars, labs, and thesis presentations and oppositions. In turn, this will facilitate and streamline monitoring of attendance, thus it may increase the use of attendance lists.

We expect that there will be benefits with regard to ethical and social issues. Registration of mandatory attendance will increase in credibility, due to the fact that a student must be physically present to swipe his/her access card. This will in turn hopefully diminish attendance forgery that occurs when students falsely register other students whom are **not** present. The proposed system should also provide greater personal privacy as the list of attendees need not be visible to anyone other than the instructor.

There will also be sustainability benefits due to the reduced use of paper and the internal transportation of this paper from place to place. Since all students and teachers at KTH ICT already possess access cards (and each person is already issued such an access card for room access), there will not be any increase in the number of these plastic cards.

1.4 Goals

The main goal of this project is to design, develop, and evaluate an application prototype that creates an attendance list by using a NFC reader to read access card data from the access card which each students already has. The attendance list should be able to be easily transferred by the instructor to administrative systems such as KTH Social and Daisy.

To achieve our main goal, we need to achieve the following sub-goals:

- Increase the trustworthiness of attendance,
- Stimulate students' level of ambition (in terms of actually attending the events that they are required to attend), and
- Reduce paper and time consumption for these administrative tasks.

1.5 Research Methodology

This thesis project began with a literature study in order to gain a solid base of knowledge of relevant topics. This literature search primarily used the KTH Library in order to find and access journals, articles, books, reports, student theses, and other reliable sources.

The development of the application will be iterative and incremental to facilitate the process of producing a partially complete system [6]. This will be necessary due to the need for a functioning system in the early stages of the project to allow for iterative system design, testing, and evaluation.

1.6 Delimitations

This project will only focus on developing an application compatible for Windows computers, rather than an application for mobile devices (e.g., smartphones and tablets), because of the fact that we have no earlier experience with mobile application development. Also, there is no unified way of writing an application for multiple mobile operating systems (since they use different API:s), which is another factor for why a mobile application would not be suitable for this thesis project.

The system will only implement reading of KTH access cards and will do this reading using a USB connected NFC reader (specifically the ACR122U – see Section 3.3.2). There will be no large scale real world tests of the application, instead we will focus on small tests to ensure functionality.

1.7 Structure of the thesis

This first chapter introduced the background, problem definition, and goals of this thesis. Chapter 2 will present relevant background information that is needed to understand this thesis project, e.g. NFC and LDAP. Chapter 3 presents the research methodology and methods used in this thesis project. Chapter 4 describes the design, testing, development and final prototype of the application . Chapter 5 is an analysis of our test results. Finally, the thesis concludes with Chapter 6 which summarizes our conclusions and suggest some future work.

2 Background

This chapter provides some essential information about RFID, NFC, and LDAP. These three areas are pivotal to understand the system developed in this thesis project. Furthermore, some related work will also be described and analyzed.

2.1 Radio Frequency Identification (RFID)

RFID is a technology based on communication through electromagnetic signals that enables identification and tracking of RFID tags. These tags may be attached to (or even embedded inside) objects. The communication supported by RFID can occur between movable objects or between stationary and movable objects [7]. An RFID reader enables automatic identification based upon RFID tags, rather than manual scanning used together with barcode technologies. Different frequency band allocations are used for different types of RFID systems, based on what transmission ranges are needed. The most common frequencies are: low frequency (125 – 134 kHz), high frequency (13.56 MHz), and ultra-high frequency (433 and 858-930 MHz) [8].

There are three main components of an RFID system: tags, interrogators, and controllers. A tag transmits data that is has stored to an interrogator when it receives a signal from the interrogator, as long as it is within that interrogator’s electromagnetic field [9]. Tags can either be active, meaning that they have their own power source and thus they are able to transmit data up to a range of 100 meters, or passive, indicating that there is no power source, instead the tag is powered by the interrogator [10]. The controller, which is generally a PC, processes the information received from the interrogator via some interface.

2.2 Near Field Communication (NFC)

Near Field Communication (NFC) is 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, but is generally less than 10 cm [11] with a maximum read/write speed of 424 kbps [12]. From a security perspective, this short range makes it difficult for another party to intercept the traffic as they have to be very close to their target. NFC is based on RFID technologies, such as ISO/IEC 14443 and ISO/IEC 18092, and operates on 13.56 MHz (i.e., high frequency RFID) signals [13]. NXP Semiconductors MIFARE is a smart card technology based on ISO/IEC 14443 and is commonly used in access control. FeliCa is another smart card technology based on IOS/IEC 18092 that is popular in Japan where it is used in ticketing for public transportation [14]. The architecture of NFC is illustrated in Figure 2-1.

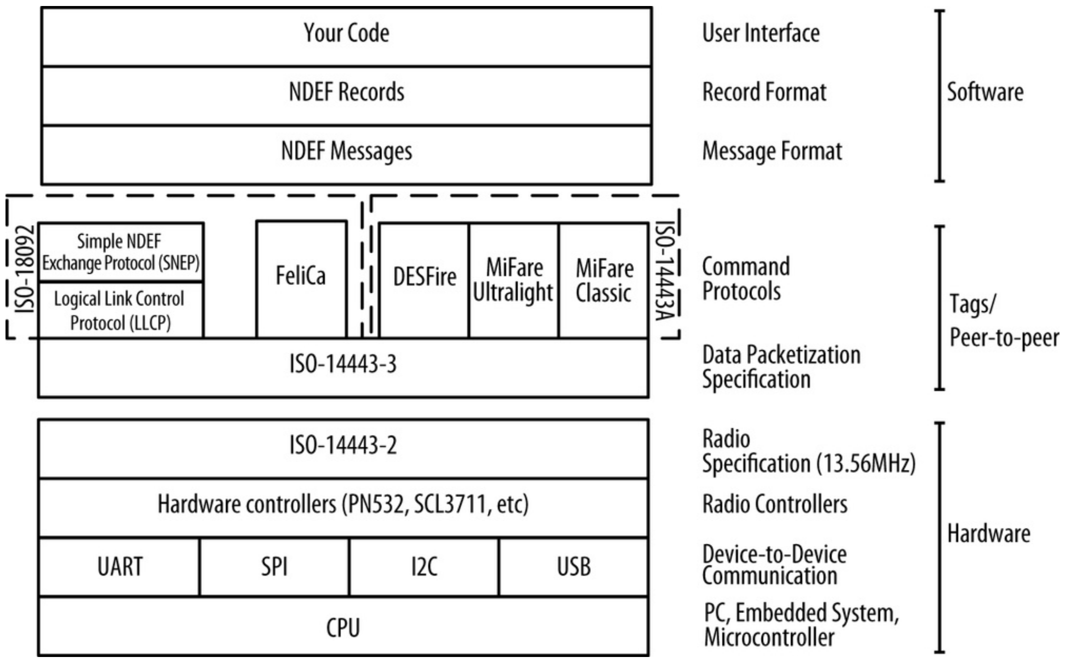


Figure 2-1: The NFC protocol stack [13]. (It appears here with permission from O'Reilly Media, Inc.)

2.2.1 Communication modes

A NFC device can either be active, which means it has a power supply that enables it to generate its own radio frequency (RF) signal, or passive, meaning that it does not have a power supply and thus has to be powered by another device. A passive device has the advantage of not needing a battery or other built-in source of power.

NFC communication has two different modes: active or passive. In active mode, both communicating devices generate their own RF signals, thus allowing either of the devices to initiate transmission, i.e., either device can be a initiator or a target. The initiator generates a RF signal and the target responds to the initiator when it receives the signal. In passive mode, the communication occurs between an active and a passive device, with the passive device deriving its power from the initiator [15].

2.2.2 Operating modes

Based upon the article "NFC Essentials"[14] by Coskun, Ok, and Ozdenizci, and the *NFC Forum*'s "What It Does"[15] NFC implements three different modes of operation: read/write mode, peer-to-peer mode, and card emulation mode [16][17]:

- In read/write mode, the active NFC device can read and write data in a passive device, such as a NFC tag. Schemes such as ISO 14443 and FeliCa are compliant with the read/write mode.
- Peer-to-peer mode allows communication and data exchange between two active NFC devices. This data exchange could for example be files shared between two NFC-enabled mobile phones. The peer-to-peer mode is standardized in ISO/IEC 18092.
- The card emulation mode allows devices to emulate tags and smart cards when communicating with an external reader. This mode enables two-way communication and thus enabling NFC devices to perform contactless payments, access control and ticketing. The standard used is ISO 14443.

2.2.3 Type of devices

An NFC enabled mobile phone, NFC reader, and NFC tag are the three different types of NFC devices [16]. NFC tags are passive devices with unique identifiers (UIDs) and are normally access cards or stickers (such as those found on advertisement posters). NFC tags can be of different types (type 1, 2, 3, or 4) as defined by the NFC Forum [18]. These tag types differ in memory capacity, transmission speed, price, and compatible products. Communication between two NFC tags is not possible, as tags must communicate with active devices such as NFC readers.

NFC readers are active devices and can be either internal or external. An external reader, such as a NFC POS terminal or an USB NFC reader, can read information from another NFC device. Internal readers, such as used in NFC-enabled mobile devices, are integrated into a device. This integration allows NFC-enabled mobile devices to be active devices and to communicate with passive devices such as tags [16].

2.2.4 NFC vs. RFID

Since NFC is an extension of high frequency RFID and they both operate on 13.56 MHz. (As mentioned in Section 2.1, RFID can also operate in other frequencies – depending upon the specific device). Figure 2-1 on page 6 illustrated the NFC architecture and some unique features for NFC that are not available in RFID. Two of these features are the NFC Data Exchange Format (NDEF) Message and Record field. In addition, there are two ISO 18092-based protocols: Logical Link Control Protocol (LLCP) and Simple NDEF Exchange Protocol (SNEP) [13]. These fields allow NFC to perform two-way communication as opposed to RFID where the initiator can either write to or read from the tag. The two-way communications are peer-to-peer and card emulation, as mentioned in Section 2.2.2.

2.2.5 Uses

Due to the simple usage model that NFC provides, there are currently many uses of NFC devices in society and the adoption of NFC is expected to continue growing during the forthcoming years [19] and there is. Stockholms Lokaltrafik (SL- Stockholm Public Transport) implements NFC/RFID access cards in their ticketing system, allowing users to simply hold their access cards in front of a reader rather than using paper tickets with magnetic stripes, barcodes, or manually applied stamps. Transport systems in London also utilize NFC/RFID access cards. The local government body “Transfer for London” (TfL), responsible for the transporting systems in London, offers access by NFC-enabled credit cards and mobile phones [20]. KTH ICT utilizes NFC/RFID access cards, allowing students and teachers to lock/unlock doors inside the university. One common property of the access cards used by SL, TfL and KTH ICT is that they are MIFARE type cards.

Today it is quite common to connect two devices, such as a mobile phone and a wireless headset; unfortunately, this is not a simple process. However, using NFC this pairing can be done simply by tapping the devices together – once they are in range of each other the devices can exchange the information needed for pairing. Note that subsequently NFC is **not** used as the actual data transfer technology for sending data between these devices as its transmission speed is quite slow, instead the data transfer is done via faster wireless technologies, such as Bluetooth or Wi-Fi.

2.3 Lightweight Directory Access Protocol (LDAP)

The Lightweight Directory Access Protocol (LDAP) is a protocol for accessing and managing directories over an IP network. LDAP is used to access directories that store and organize information. According to IETF’s RFC 4511: "LDAP provides access to distributed directory services that act in accordance with X.500 data and service models. These protocol elements are based on those described in the X.500 Directory Access Protocol (DAP)."[21]

LDAP directories use a directory information tree (DIT) as a data structure. Entries in a DIT are based on the Distinguished name (DN) of an object and structured in a tree hierarchically. An example of a DN is "cn=Andreas Andersson,ou=users,dc=example,dc=com". This is an entry in the tree shown in Figure 2-2.

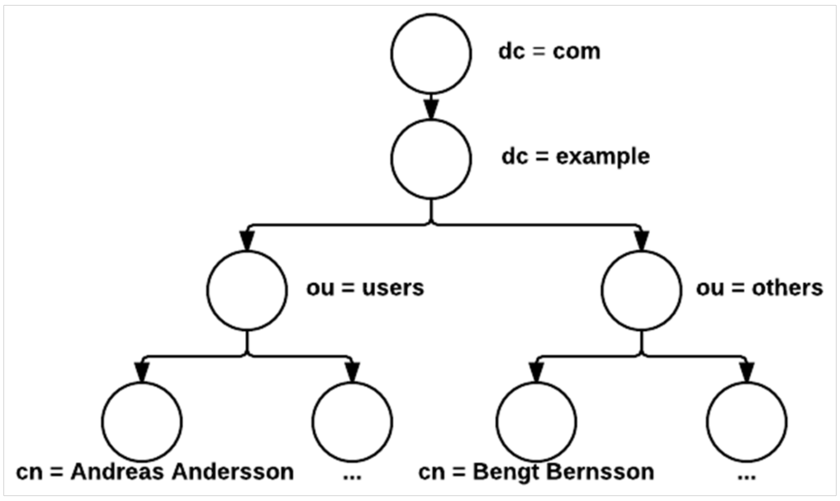


Figure 2-2: Example of a LDAP directory tree structure

The content that can be stored in an LDAP database is based on schemas. The schema defines object classes and attributes that can be used inside the database, e.g. if a schema defines an object class A containing one or more attributes, then a user can add one (or more) objects of type A to the database.

2.3.1 Schema

A schema is a type of document used in LDAP that describes attributes and/or object classes. For LDAP to be able to create an object of a certain class that class must first have been defined in a schema. The attributes used by the object must also be defined in a schema. Schemas are written as normal documents, then converted and imported to the LDAP database. If an implementation is defined in a schema, but unknown by the LDAP server - the object classes and attributes it describes will be unusable [22]. Each element in a schema must be identified by a globally unique Object Identifier (OID), these OIDs are hierarchical and a organization can create as many branches as they want from their root OID. An example of an OID is: 1.3.6.1.4.1.4203.666.* that is OpenLDAPs experimental branch.

Figure 2-3 is an example of the schema used in this project. This is an initial version and should not be thought of as the final product, but is here to give the reader an idea of what a schema looks like. This should help the reader when reading this subsection and following two subsections about attributes and objects.

```
attributetype   ( 1.3.6.1.4.1.4203.666.9.999.1
        NAME 'cardid'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024}
    )

attributetype   ( 1.3.6.1.4.1.4203.666.9.999.2
        NAME 'kthuser'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024}
    )

objectClass     ( 1.3.6.1.4.1.4203.666.9.999
    NAME 'carduser'
        DESC 'CardUser'
    STRUCTURAL
        MUST   ( cardid $ kthuser )
    )
```

Figure 2-3: An early example of the schema used in this project

2.3.2 Attribute

An attribute has a unique name (this is only required to be unique within a specific system) and contain some type of data,. Each attribute is a member of one or more object classes. The attribute itself has a type that defines what it can contain, e.g. it is an Integer type- this means that it can contain a 32-bit numeric value. In the schema for an object class containing this attribute the schema specifies whether or not the attribute *may* occur or *must* occur, i.e. whether this attribute is optional or mandatory. The value of an attribute can be specified as being single valued or multi valued. The latter indicates if more than one value can be stored in a specific attribute. For example, multiple e-mail attributes might be associated with one person. An attribute can be part of a hierarchy, if so they inherit all of the properties from their parent. [22]

2.3.3 Object classes

Object classes could be seen as containers for attributes, but are considered attributes themselves. There are some predefined object classes in LDAP and a user can add new classes through a schema. The object class must have a globally unique name or identifier. As an attribute can be part of a hierarchy, so can an object class, i.e. an object class can inherit attributes from its parent as well as add new attributes, effectively extending the original object class. [22]

2.4 Related work

This section present some related work on student attendance systems utilizing NFC. The purpose of these systems works are generally the same, but they differ in their designs and implementations.

2.4.1 Student attendance monitoring using NFC and fingerprint readers

Benyo et al. [23] developed a NFC based attendance system to increase student attendance at lectures. They used a scalable back office and contactless terminals together with not only access cards (MIFARE Desfire), but also fingerprint identification. In order to prevent students from using another student's card, there was a binding between the student's identification and their fingerprint. The sensitive fingerprint data were stored in the MIFARE Desfire cards since they were considered secure, thus avoiding unnecessary storage in any part of the monitoring system. The back office collected and stored attendance information gathered from the terminals via wireless communications.

2.4.2 TouchIn: a web based application

Ayu and Ahmad developed an NFC supported web based attendance system called “TouchIn” [24]. The system was implemented as a web based application using PHP, JavaScript, and MySQL. The system consisted of one reader unit and one web server unit. The reader unit is an application installed on an NFC-enabled device, while the web server unit is a computer hosting web services. Three different scenarios were implemented: a student with an NFC-enabled device and internet connectivity, a student with an NFC-enabled device but without internet connectivity, and lastly a student without either a device or internet connectivity but equipped with a NFC-tag. In the first scenario, NFC communication between the mobile device and a tag attached to a poster will load and submit the student's ID, the device's ID, and a course code. The second scenario requires the lecturer to set the course code in an application on his/her NFC-enabled mobile device, thus allowing attendance registration through communication between the lecturer's mobile device and the student's device. The last scenario requires NFC tags containing the student's ID. These tags are read by the lecturer's mobile device. Attendance requests and responses are handled in the web server unit.

2.4.3 Control of attendance through mobile NFC technologies

Fernandez et al. [25] developed a mobile application to reduce the time wasted doing manual student attendance registration. After comparing different technologies (such as Bluetooth, Infrared Data Association, RFID, and NFC), they came to the conclusion that NFC would be the best solution due to its speed and wide availability in smartphones. The application allows students to sign in and register their attendance by tapping their NFC-enabled smartphone (acting as an active device) on an NFC tag (a passive device). The institution manages (through a management application) the time frames that determine the limits for a students' arrival to be considered as attending an event. The application has the ability to give calendar information specific to each student, for example, showing attended/missed classes, dates for upcoming classes, current attendance percentage, and much more.

2.4.4 Easy Attendance: Location-based authentication for students integrated with Moodle

M. Bucicoiu and N. Tapus describe a simple location-based student registration system that is optimized to fast, easy to use, and integrated with existing technology; thus overcoming system complications and high cost [26]. The system requires that the student and teacher first identify themselves to an aggregation platform, in this case a Moodle [27] plug-in running on a back-end server. This back-end server then creates a session enabling the student to touch the teacher's NFC-enabled phone with their own NFC-enabled phone. A picture of the student appears on the teacher's phone, giving the teacher an opportunity to grant attendance only if the picture matches the student.

PHP was used as the programming language for the development of the Moodle plug-in that provided web-services. The communication between the NFC-enabled phones and the back-end server was done through XMLRPC over HTTPS.

2.5 Summary

The NFC technology is growing and thereby increasing the demand on NFC-compatible applications and systems. NFC is used in public transportation, access control, payments, etc. and there has been some studies and related work on using NFC to take attendance of visitors at events (e.g. at university lectures).

Table 2-1 gives a summary of related work and notes the advantages and disadvantages of these earlier systems. These related works will be kept in mind when forming our own functional requirements in Section 4.1.3.

Table 2-1: Summary of related work

Related work	***Advantages***	***Disadvantages***
Benyo et al.	Binding between student ID and fingerprint thus increasing reliability.	Wireless communication between the terminals and the back office is considered unreliable.
Ayu and Ahmad	The system supports three different scenarios and has a minimal hardware requirement of a NFC-tag and a NFC-enabled phone	None
Fernandez et al.	Additional information provided by the application, such as; the minimum and actual attendance percentage, the upcoming classes in the subject, etc.	Students are required to possess a NFC-enabled phone (active device) since the device in the classroom is a NFC tag (passive device).
Bucicoiu and Tapus	The system integrates with an existing platform (Moodle) used by the university.	Students are required to possess a NFC-enabled phone

3 Methodology

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes the research process. Section 3.2 focuses on the data collection techniques used for this research. Section 3.3 describes the experimental design. Section 3.4 explains the techniques used to evaluate the reliability and validity of the data collected. Finally, Section 3.5 describes the framework selected to evaluate the application.

3.1 Research Process

Upon completion of this process a prototype of an NFC based attendance application will be present, we have decided to split our method into four phases as it would give us a chance to go back and repeat steps taken. Since each phase is run through multiple times we hope to weed out any bugs as well as find flaws in functionality and what other functionality might be needed to get a prototype that can show the potential of such a system.

3.1.1 Phase 1: Information gathering phase

In this phase information were supposed to be gathered and researched to collect the necessary knowledge for developing the application. This involves reading reports on similar subjects written by other people and collecting good ideas for this project. It also involves gathering general knowledge and freshening up on our programming skills to prepare for the practical part of the project.

3.1.2 Phase 2: Developing individual parts of the application

This phase consists of three sub phases:

1. Research in order to acquire the knowledge and skills needed to create the necessary functionality for each part of the application.
2. Create each part of the application using pair programming to ensure the utilization of both persons' knowledge.
3. Test of each part of the application to ensure that functionality is what we looked for, this involved looking for minor bugs as well as code optimization.

3.1.3 Phase 3: Create and test the Graphical User Interface

In this phase a Graphical User Interface (GUI) is created and then integrated with all the parts from phase 2 to achieve the desired functionality. This phase also includes testing to ensure that the application responds as desired and that none of the parts clashed with each other.

3.1.4 Phase 4: Evaluation

In this phase we evaluate our application and the functionality it provides to determine if anything is missing or obsolete. If something is considered obsolete, then it should be removed; while if anything is still missing, then return to phase 1 to start gathering new information and add the missing functionality.

3.2 Data Collection

The initial data collection will utilize a set of test cards, while subsequent testing will use KTH access cards (assigned to the authors and their classmates). This subsequent testing with actual users will be limited to trials with a small number of instructors (as described in Section 3.2.2). The initial target population is the students and instructors at KTH ICT (as described in Section 3.2.3). Future testing on a university scale is outside the scope of this thesis, but our design should scale to tens of thousands of students and thousands of instructors.

3.2.1 Sampling

A host computer with a virtual machine running an OpenLDAP server acts as the information gathering station running our application. This host computer has a NFC-reader connected to it via USB. Currently we have 8 MIFARE test cards, as well as 2 backup test cards. These test cards are used together with the host computer to generate test attendance lists.

The test cards are linked to KTH users in the LDAP database (with the cardid <-> kthuser connections), the KTH usernames connected to the test cards were initially chosen at random from the Winnie-the-Pooh universe's Swedish characters. After communicating with Robin Roy (KTHs Personal data representative) we were informed that even if the information used is not sensitive some form of consent were preferred, we then switched to using usernames of other students at KTH ICT that consented to us using their usernames for demonstration purposes.

Due to the lack of test cards a function has been created that generates random first name and last name combinations (together with fake e-mail addresses and usernames), this has been done to populate the attendance list during test. We decided to keep these names random instead of getting them from the KTH LDAP database as some people might object to us using their names while testing the application.

3.2.2 Sample Size

Due to the fact that there is no maximum number of students that can attend a course in KTH (it is limited to the resources) we have no idea how many people will be registered in one event where the application is present. Since we have access to 10 test cards, we will also simulate 10 extra users, giving us a total of 20 users when testing the application. The sample size of instructors will be 5, which is the optimal number for usability tests based on multiple articles[*][†][‡].

3.2.3 Target Population

The target population is the number of students participating in courses at KTH ICT with the need for signing attendance lists and the number of instructors who have sections in such courses. Estimating the number of access cards currently deployed is straightforward since all students, staff, and teachers each possess one. In 2014, the total number of students taking courses at ICT was 1301 and the total number of employees were 321 [28].

The legal requirements of Personuppgiftslagen (PuL), "The Personal Data Act", are examples of some of the ethical concerns that should be taken into consideration during the data collection part of this thesis project. The purpose of PuL is to protect people's integrity when personal data is collected and manipulated, for instance, stored in a database [29]. Our system will handle names

[*] http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/
[†] http://www.measuringu.com/blog/sample-size-problems.php
[‡] http://www.measuringu.com/five-users.php

which is considered personal information. This information is not as harmless as one might think, since searching for someone’s name on the Internet could provide other information about the person. The use of such a system in conjunction with registering attendance is not expected to cause specific PUL issues, since the users of the application (both instructors and students) have already given approval for the handling of the necessary personal information (such as their name) in the course of their activities (work or studies) at KTH and the only information that is being collected when an instructor uses the application is the same information that would be collected via a paper attendance list. The application collects only the data that the instructor would normally collect, hence there is no change it the exposure of data to the instructor, but as noted earlier there is a *decrease* in the exposure of the data to other student in the classroom, laboratory, seminar, etc.

There are questions regarding PuL with respect to the collection of the binding between the KTH access card number and the user’s KTHID – since this information is currently not kept outside of the BRAVIDA system. There will be PuL questions as to whether the new database will need to be integrated and managed by the KTH (should it go into production use), as the database should be given the same protections as the existing LDAP database and presumably the same rules will apply to its use.

3.3 Experimental design and Planned Measurements

The testing carried out in conjunction with this thesis project has two main components: initial testing during phases 2 and 3) and testing on a small scale with a limited number of instructors (during phase 4). The test environment is described in Section 3.3.1 and the hardware and software are described in Section 3.3.2.

3.3.1 Test environment

The testing of the software will initially be performed on a laptop computer running Microsoft’s Windows 7. Additionally, Java Runtime is needed on this laptop as the prototype of our application is written in Java. The laptop has a ACR122U NFC reader connected to it through USB and will interact with the Mifare 1k test cards used to simulate student access cards. This same laptop hosts a guest OS (Linux) running OpenLDAP to allow lookups in our new database.

The exact specifications for the laptops used in this thesis project are described in Table 3-1. The NFC reader’s specifications is shown in Table 3-2, observe that the application program data unit (APDU) (used by the application to communicate with the reader) is for ACR122U and this same APDU might not work with other readers. The test cards used are Mifare 1k, instead of the Mifare 4k Plus KTH uses. However, this will not have any negative impact on this work since both our test cards and the KTH student access cards both use 4 byte UIDs [4].

3.3.2 Hardware and Software to be used

Two test systems will be used during this thesis project. Their specifications are given in Table 3-1. A NFC (or RFID) card reader is necessary for the test environment. For this thesis project we have used an Advanced Card Systems Holdings Limited ACR122U. These specifications for this NFC card reader are given in Table 3-2. Table 3-3 specifies the software needed and their versions.

There is a guest server running on the ASUS K55V to act as LDAP server to enable the lookup of KTH usernames from card IDs. The server is running OpenLDAP 2.4.31 on Ubuntu (64-bit) Server 14.04.1 LTS in Oracle VirtualBox 4.3.22 r98236 with 1 core and 1024 MB RAM, the kernel version is Kernel 3.13.0-32-generic. The LDAP server has the standard schemas with one addition: the schema shown in Figure 2-3 is also installed through the ldapadd[30] utility (it was first converted using the slaptest[31] utility). Details on the installation of the LDAP server is available in Appendix A.

Table 3-1: Technical specifications for test computers

Test system	***System 1***	***System 2***
Model/make	ASUS K55V	ASUS G551JM
CPU	I7-3610QM @ 2.30 GHz	I7-4710HQ @ 2.50GHz
RAM	8 GB @ 1600 MHz	16 GB
GPU	GeForce GT630M	GeForce GTX860M
HDD	1TB @ 5400 rpm	250 GB SSD
NIC	Realtek PCIe GBE Family Controller	Realtek PCIe GBE Family Controller
OS	Windows 7 Service pack 1	Windows 8.1

Table 3-2: Technical specifications for USB NFC-reader ACR122U [32]

Interface	USB Full Speed
Operating Distance	Up to 50 mm
Supply Voltage	Regulated 5V DC
Supply Current	200mA (operating); 50mA (standby); 100mA (normal)
Operating Temperature	0-50 ˚C
Operating Frequency	13.56 MHz
Smart Card Interface Support	• ISO 14443 Type A & B • Mifare • FeliCa • 4 types of NFC (ISO/IEC 18092) tags
Operating System Support	• Win 98, Win ME, Win NT 4.0, Win 2000, Win 2003, Win 2003 R2, Win XP, Win Vista, Win 2008, Win 7, Win 8 • Win 2003 x64, Win 2003 R2 x64, Win XP x64, Win Vista x64, Win 2008 x64, Win 2008 R2 x64, Win 7 x64, Win 8 x64, Win 2012 x64 • Win CE 5.0 and 6.0 • Mac OS® • Linux® • Android™ 3.1 and above

Table 3-3: Software specifications

Program	*Version*	*Description*
NetBeans IDE	8.0.2	Java-written platform used for software development
ACS Unified Driver (PC/SC Drivers)	4.0.0.4	Smart card reader driver, needed for development with ACR122U
Java SDK 7	Update 51 (64-bit)	A development environment for Java

3.4 Assessing reliability and validity of the data collected

In order to acquire reliable results, the number of test cards communicating with our application must be equal to the average number of students usually attending lectures. Due to the fact that we only possess 10 test cards, we will create virtual “fake”-users. These fake users have random first and last names.

The reliability in regard to user-functionality (the user in our case is assumed to be an instructor) will be tested by manually simulating a normal user’s behavior. This normal behavior is defined as:

1. starting the application
2. setting up the event
3. starting the card-scanning procedure
4. scanning a card (which looks up information via LDAP and presents this information to the user)*
5. stopping the card-scanning procedure
6. saving the attendance list as a file
7. sending the attendance list as an e-mail attachment, and
8. closing the application

The simulation of an event will be done by having four test users test the application and then interviewing these people. This will increase the validity of our results since we will obtain “human” input concerning the usability of the application.

3.5 Evaluation framework

User-friendliness is a very important element of our application. We will test the user-friendliness by evaluating the feedback and input gathered from our test users, as mentioned in the previous section. Based on this feedback, adjustments will be made to the application.

* This procedure is repeated for each student’s access card.

4 The application

The purpose of this chapter is to give the reader an insight into the application and consists of the following parts; the major design decisions made for the application (in Section 4.1), it will describe how we worked to create the aforementioned application (in Section 4.2), what we tested and how the tests were conducted (in Section 4.3), and finally describe the finished prototype and show some of the functionality (in Section 4.4).

4.1 Design

This section describes the background of why we made certain decisions in our thesis project. Specifically we will explain why we chose Java, LDAP, and certain functionality for the app.

4.1.1 Java

There are a number of reasons for why we chose Java as our programming language. First of all, Java is platform independent, meaning that Java programs can run on all hardware and software platforms. One example of a hardware platform is a PC, and software platforms include: Windows, Linux, and Mac OS X. Building a GUI is easy and effective due to the fact that Java has a lot of frameworks (e.g. AWT and Swing) and these frameworks provide many components, such as; buttons, text fields, labels, etc. These reasons motivated the choice of java as a our programming language, especially as our application development was mainly focused on creating a very user friendly GUI.

Another reason for our choice of Java was the fact that both of us have had previous experience with Java in courses at KTH, thus we had some basic knowledge of java and how to develop java programs. We knew that we had to learn a number of concepts, but felt that Java would offer a relatively short learning process compared to other programming languages. Another determinant factor in choosing Java was that we decided to implement our system using the Model-View-Controller (MVC) pattern. Both of us have had previous experience regarding MVC in Java during the earlier stages of our education at KTH.

4.1.2 LDAP

LDAP was selected as the database to use to store the information connecting each student to an access card. To do this we created an object called “carduser”, see Figure 2-3 for the schema. This would enable KTH’s Information Technology staff to use this solution in the future by simply adding the attribute “cardid” to their current LDAP object class that represents students in the KTH LDAP server. This would require only a small change in the code of our application, as the application would use only one LDAP server, rather than two separate LDAP servers. We initially considered using an SQL database, but almost immediately discarded this idea - since if we used LDAP the possibility to merge the two databases offers a large advantage.

If at a later date KTH wants to add support for the use of cards other than the student access card, they could add a new attribute such as “slcardid” (short for SL card ID) so that students could use their SL card to register their attendance. It would also be possible to add a attribute “custcardid” (Custom card ID) so that the student can register and then use whatever NFC device they want, assuming that the NFC device they use has a GUID and not just an UID.

4.1.3 Functional requirements

This section will describe the different decisions made to realize the necessary functionality of the application and why they were made. The different functionalities are; reading a card (described in Section 4.1.3.1), loop for card handling (described in Section 4.1.3.2), storing the attendance list (described in Section 4.1.3.3), special case: course registration (described in Section 4.1.3.4), user-friendly display (described in Section 4.1.3.5) and handling missing information (described in Section 4.1.3.6).

4.1.3.1 Reading a card

We used the javax.smartcardio library to detect the presence of a card near the NFC reader and to read the UID of such a card. The key functions are checking if a card is present and getting the UID from that card. In the event of an error we get an error indication and display it via a message on the screen such as "User not found!" or "User not in KTH LDAP!" with red background and a cross. If the reading of the card succeeded and we were able to find the corresponding information in the LDAP database, then a message with the user’s CN will be displayed on a green background with a checkmark.

4.1.3.2 Main loop for card handling (CardHandler.java)

The most important functionality needed for our program to be able to achieve our goals was a way to read the student’s access card’s RFID chip. While the application is running it keep checking whether a card is present, if so then it will read the card’s UID and use this to lookup the corresponding KTHID and record this user as being present for the event.

As shown in the flowchart in Figure 4-1, the loop continuously looks for a card until one is present. Upon finding a card it reads the UID from the card and checks if this UID is present in our LDAP database. If this UID exists, then the application looks up the corresponding user information via the KTH LDAP server. However, if the UID does not exist then the application generates a prompt asking the user for his or her KTH username. Using this username the application queries the KTH LDAP server for the corresponding user information and extracts the user’s KTHID. The application can now register a new binding with its LDAP server. Once this user’s card ID and KTHID information has been stored, then the application can send a confirmation e-mail to the user about this registration. Finally, the application records the user’s attendance and loops to look for a new card.

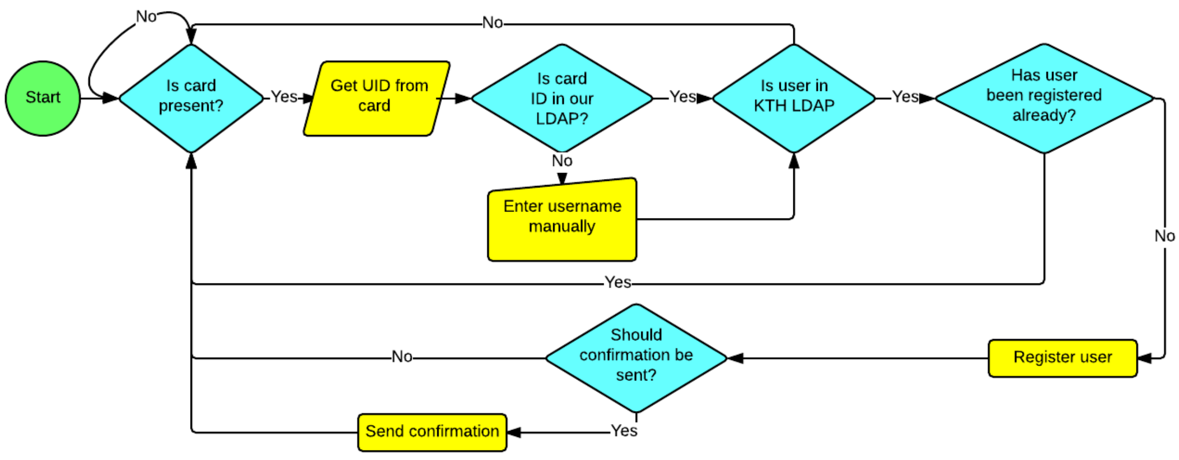


Figure 4-1: Flowchart showing the card reading loop of the app.

4.1.3.3 Storing the attendance list

Another important requirement was to give the users of the application several ways of handling the attendance lists. The required functionalities are listed below, and enable users to:

- Store the attendance list locally, either as a txt-file or XML-file,
- Send this file to his/her e-mail own address as a file attachment, and
- Upload the attendance list directly to one of the KTH course administration systems.

Unfortunately, uploading the attendance list to KTH administrative systems was not implemented due to lack of time.

4.1.3.4 Special case: Course registration

One of the Daisy developers stated that some usages of the application such as course registration could not be automated for entry into Daisy, as multiple factors(in the list below) affects whether a student can be registered or not. [33] This does not affect the part of taking attendance only the ability to take course registrations through the application.

- Is the student admitted to the course?
- Is the student registered for the current semester?
- Is the student eligible to take this course?

These factors complicate our application and requires additional development which is outside the scope of this thesis project. Also, KTH is planning to allow students to register for courses by themselves directly via KTH's administrative systems, which makes further developments into this special case functionality for the application unnecessary [33].

4.1.3.5 Visual presentation of information

The output from the application should be user-friendly and simple to understand. An attendee scanning his/her KTH access card via the NFC USB-reader should be able to see if the registration of their attendance was successful or not. To do this, we chose to make the display show the user's name and KTH-ID (the CN of the LDAP object class for the student) of the latest scanned attendee in green if registration for the event was successful, and in red if their registration attempt was unsuccessful. The colors green and red are optimal since they often resemble success and failure respectively. Additionally, in order to avoid problems for colorblind users we display a checkmark if the registration was successful and a cross if the registration was unsuccessful.

4.1.3.6 How to handle missing information

The scenario of a student scanning their card resulting in an unsuccessful registration was taken into consideration. This unsuccessful registration could occur because a card is not yet registered (not connected with a specific user in our LDAP database). Our application provides the possibility for instructors to manually enter the KTH username of the student in a dialog window that appears when there is an unsuccessful card registration. After the username is entered, our application will query the KTH LDAP database for the specified username and subsequently successfully register the student and show their CN on the screen.

Previously in this thesis project, there were thoughts of implementing a feature that would allow users of the application to add a student and/or card in the LDAP database, but this was not implemented due to the fact that we believe that this functionality should be done in a centralized fashion via an administrative unit of KTH ICT. As attendance is most likely taken during the first 15 minutes of the lecture, seminar, etc, the process of adding user and card combinations to the LDAP database directly in the application might slow it down(in terms of registering other attendees). As the application is expected to be fast and simple, we believe that adding people to the database is best done via the IT helpdesk or another administrative unit.

4.2 Development

The coding of the application was done exclusively in NetBeans IDE 8.0.2. By deciding upon a single IDE we were able to use the exact same settings while coding and were able to push everything including our configurations to Bitbucket* using git. This simplified our work as we did not need to waste time trying to setup two different IDEs in the same way.

Early on a decision was made to do most of the coding using the agile software development technique of pair programming. Pair programming has two roles that the users frequently switch between. The first role is the driver - who writes all of the code while in the driver role. The second role is the observer who reviews each line of code as it is written and considers how to proceed as well as considers potential flaws in the current code. The decision to use pair programming was mainly based on the fact that we both wanted to write equal amounts of the code, while neither of us needed to individually understand all of the details of all of the code. However, we both have an understanding of the code as a whole.

An important part of the project was searching, mainly on stackoverflow posts, for multiple examples of working (and non-working) ideas when trying to understand how to develop a certain function of the application. By looking at other people basic ideas and examples we were able to incorporate their knowledge into our knowledge, hence extending the good parts while discarding the unnecessary parts.

The main part of the Java library that were used for this project was the Java Smartcard I/O API (the package javax.smartcardio) which "… defines a Java API for communication with Smart Cards using ISO/IEC 7816-4 APDUs. It thereby allows Java applications to interact with applications running on the Smart Card, to store and retrieve data on the card, etc."[34]. This package enabled us to connect to and use the USB NFC Reader and to communicate with smartcards using the APDU commands found in the ACR122U API[35]. An example of code that uses the ACR122U API and javax.smartcardio package is Figure 4-2 below. If this code is compiled and run it will print the UID of any card it can communicate with.

* https://bitbucket.org/ - A web-based hosting service for projects that use the revision control systems Mercurial or Git.

```
try {
    TerminalFactory terminalFactory = TerminalFactory.getDefault();
    List<CardTerminal> cardTerminals = terminalFactory.terminals().list();
    CardTerminal terminal = cardTerminals.get(0);
    Card card;
    CardChannel channel;
    ResponseAPDU answer;
    byte[] uid;
    while(true){
        if(!terminal.isCardPresent()){
            Thread.sleep(500);
        }
        else{
            card = terminal.connect("*");
            channel = card.getBasicChannel();
            answer = channel.transmit(
                    new CommandAPDU(0xFF, 0xCA, 0x00, 0x00, 0x04));
            uid = answer.getBytes();
            String out = "";
            for(int i = 3; i > -1; i--) {
                out += Integer.toHexString((int) uid[i] & 0xFF);
            }
            System.out.println(out);
        }
    }
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Figure 4-2: Sample code that with an ACR122U USB NFC Reader connected to the computer would print out the UID of a card placed near the reader*.

The reason that the for-loop looks the way it does is that the Response APDU has the format shown in Table 4-1, the Least Significant Bit (LSB) is first and the Most Significant Bit (MSB) is last, therefore the UID value has to be read backwards. This same process is performed our application so that the UID is recorded the correct way.

Table 4-1: Shows the structure of the Response when a Command APDU is sent to get the UID of the currently connected card, SW1 and SW2 is the status code. Adapted from a table in ACR122U API section 4.1.[35]

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

* *Note: Do not forget to add "import javax.smartcardio.*" in the beginning of the file if you try this example.*

4.3 Testing

Testing was done in 3 parts. Section 4.3.1 and Section 4.3.2 were done by us. Section 4.3.3 was done by volunteers, under our supervision. Section 4.3.4 describes how we tested the time needed to make the LDAP queries and get a response.

4.3.1 Testing of individual modules

Testing of individual modules (classes) was done to check that they performed as they should in the form of functionality rather than efficiency. The testing itself varied from class to class, but in this section we will use the SaveHandler class as an example, but overall we tested that all of the methods performed the functionally they were built for.

Some classes required us to create fake data for testing, the SaveHandler is one of these classes that required fake data for testing. First of we created a Test() method inside a class called Test. In this class we created the objects that we needed to test our class, such as a SchemaEvent, an ArrayList containing Visitor(s), and in the case of SaveHandler a target file. Testing was performed by repeatedly saving information to different files to see if the code would save the values properly, even when SchemaEvent was empty, when the ArrayList was empty, when no file was specified, and similar tests cases.

If the method, in this case saveToTxtFile(ArrayList<Visitor> visitors, SchemaEvent se, File file) in SaveHandler.java, was able to save to a file when all necessary information was provided and if it returned the correct value (true/false), then this function was considered functional. If the tests broke, then we would look at the code and print debugging information to find where the code failed and fix that part. Then we tested this code once again. Once a module were considered functional, i.e., it did what it was supposed to under normal conditions, then the code was integrated together with the other modules and the GUI for further testing (as described in Section 4.3.2).

4.3.2 Testing the modules together

First of all, the application was tested to see if it would start and stop as it should. These tests were based on the normal behavior specified in Section 3.4. Then we created the CardHandler.java class in order to have a working module that successfully communicates with the NFC USB Reader and reads the UID of a student access card. This enabled us to test the module with the GUI to see that it still provided the same functionality as when tested independently, while now also updating the display. Following this we could test the normal behavior, including parts of the application that relied on the CardHandler.

The next step was to implement the LDAPQuery class to see if we could resolve a cardid to KTH usernames and then get the relevant information about the student (based on the user's KTH username). At this point the CN of the student associated with this card is supposed to be displayed. After this basic testing we continued to add new modules. This procedure was repeated, adding a module and testing as much of the normal behavior scenario as possible, until all the modules had been added to the application. At this point we could start testing the full application as it was supposed to work as it has shown all of the necessary functionality.

Changes to the code were made based on testing the application as a whole to ensure user-friendliness and to optimize the application's functionality. One of such changes was the addition of an icon next to the name of the most recently scanned attendee in the form of a checkmark/cross based on successful/unsuccessful registration for the even in order to ensure that the difference was distinct for those who are unable to differentiate between the two colors that were used.

4.3.3 Testing the GUI with Volunteers

In order to improve the application and make any changes necessary to achieve our goals, we had volunteers test the application and then ask them some questions. Table 4-2, Table 4-3, and Table 4-4 summarizes the questions ask of the volunteers, their answers, and our actions. In order to gather relevant answers, the volunteers simulated the normal behavior (as described in Section 3.4) with some modifications. These modifications were made since different users perform these tasks in different ways.

Table 4-2: Answers and measures regarding unexpected behavior when using the application

Did you experience any unexpected behavior when using the application?		
Tester	**Answer**	**Our Actions**
A	NullPointerException caused when trying to save an event after a failed search.	Implemented fail check that prevents the save function from being run if no search results were returned.
B	The display-text in the start screen does not change when the start/stop option is clicked.	We changed the display-text so that it clearly says that the card scanning has not started yet. The display-text now changes with regard to whether it is running the card scan or not.
C	The application did not register the card when the card was removed too quickly during a scan.	There is nothing we can do to prevent this. The card should not be removed too quickly.
D	Card scanning does not work directly after the application is started	This behavior is intentional and is now properly communicated to the user.

Table 4-3: **Answers and measures regarding potentially required application clarifications**

Is there something in the application that needs to be clarified?		
Tester	**Answer**	**Our actions**
A	It is difficult to see if the application card scanning is on or off (the menu option “start/stop”).	We changed the text on the menu option “start/stop” so that it would say “Start scanning” when the card scanning is stopped, and vice versa.
B	It is not possible to save a file if no schema event has been added.	If no schema event is added, then the application will only save the attendee’s information, rather than not saving anything at all.
D	It is difficult to see: 1. What you actually save when clicking on “File → Save → As text/XML” 2. What you send when clicking on “File→ Send as e-mail” 3. What the “e-mail notifications” option does.	1. We changed the text to “Send attendance list” to further clarify the save button. 2. Same measure as the previous point, but changed the text to “Send attendance list as e-mail”. 3. Same measure as the previous point, but changed the text to “E-mail notifications for registrants”

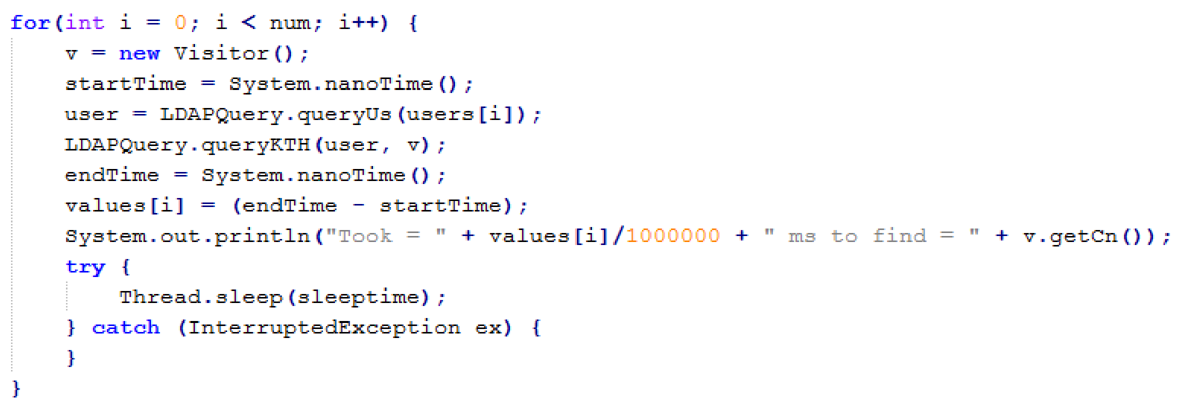
Table 4-4: Answers and measures regarding suggestions on improvements of the application

Do you have any suggestions on how to improve the application?		
Tester	**Answer**	**Our actions**
A	You need to specify what kind of input is needed in the "setup scheduled event"-option in a better way. An example is where the user needs to enter the period of the course, there you should have a placeholder and not a hover text.	The period text field is now prefilled with 2015VT, no placeholder were added due to time restraints.
B	You should not have case-sensitive input forms (e.g. where the user enter the period of the course).	Changes were made in the code in order to meet this requirement. Input is now casted to upper case.
D	1. The "attendance taken-from" when setting up a scheduled event should be filled in automatically based on when the current time. 2. Make the font size smaller in the "current attendees"-display, it is unnecessarily big 3. It would be nice to have an option to remove registered attendees 4. The "current attendees"-display should either have a close-button or be integrated in the main screen instead of being a separate screen 5. When a user that has already been registered tries to register again the screen color should change to yellow and not continuing to be green.	1. We changed the code to meet this requirement. 2. We made the font size smaller. 3. No measures will be made for this since we do not think that this functionality is needed 4. A close button has been added. 5. We'll keep it green as more colors might make it harder for those that are colorblind.

The results from these tests will be analyzed in Section 5.1.1.

4.3.4 Time to perform an LDAP query

The module for LDAP (LDAPQuery.java) was tested to measure the time it takes to complete both of the LDAP queries: the one to our own LDAP database and the KTH LDAP database. This means that we will subtract the time from just before the first LDAP query from the time just after the second LDAP query response arrives in order to compute a total query response time (in milliseconds – written below as "ms"). The main part of the test code is shown in Figure 4-3.



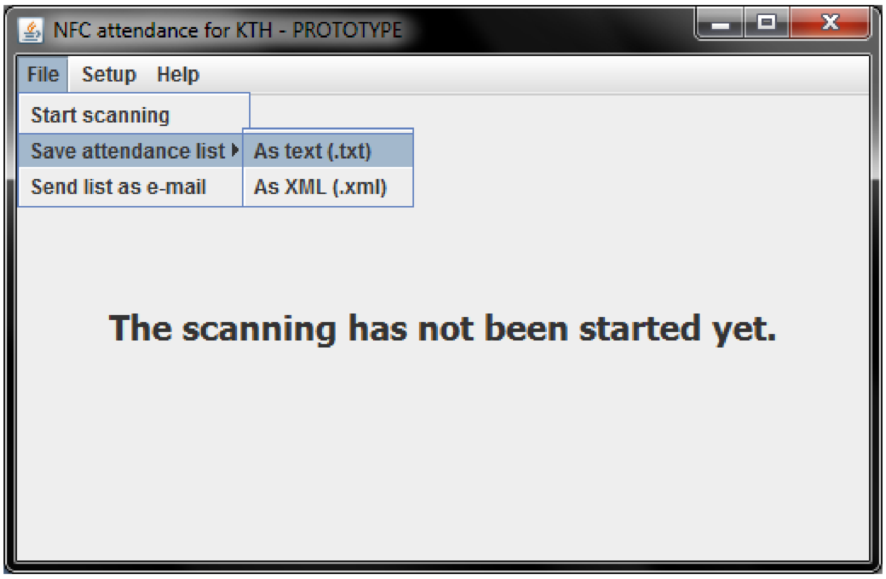
```
for(int i = 0; i < num; i++) {
    v = new Visitor();
    startTime = System.nanoTime();
    user = LDAPQuery.queryUs(users[i]);
    LDAPQuery.queryKTH(user, v);
    endTime = System.nanoTime();
    values[i] = (endTime - startTime);
    System.out.println("Took = " + values[i]/1000000 + " ms to find = " + v.getCn());
    try {
        Thread.sleep(sleeptime);
    } catch (InterruptedException ex) {
    }
}
```

Figure 4-3: Part of the code to measure the time taken for LDAP queries.

The for-loop will be run executing 5 queries (for different users each loop), the number of queries was restricted to 5 as that was the number of students that at the time had allowed us to use their usernames. The time for each query to complete is recorded and used to calculate average time, minimum time, and maximum time. The test code will be executed multiple times with different delays between the queries, the planned delays are: 1000 ms, 1 0000 ms, 2 0000 ms, and 60 000 ms. These delays are used to simulate possible delays between adding each new student. The first value 1 000 ms was chosen as a starting point as any less time would mean that the student would not even have time to check if their registration succeeded or not. The last value of 60 000 ms was chosen as the maximum delay due to the fact that most students will register attendance one right after another at the beginning of a lecture, lab, or seminar. Additionally, it is still possible to register your attendance after a longer time in order to handle the occasional straggler.

4.4 Final prototype

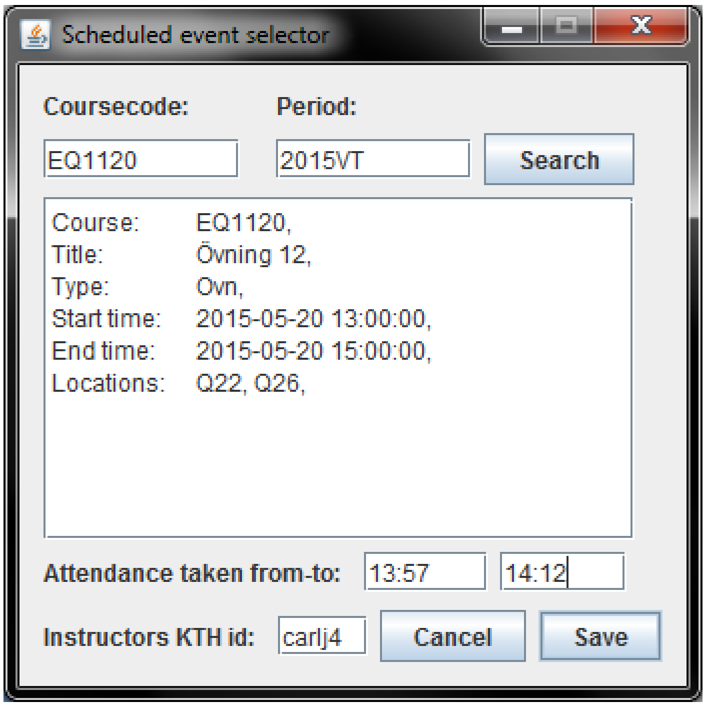
Figure 4-4 shows the start screen of the application. One of the menu options is called “File” (shown as selected in Figure 4-4). This provides the options to start/stop the card scanning, save the current attendance list, and send the current attendance list as an e-mail attachment. The save-option allows the user to save the current attendance list either as a txt-file or a XML-file. After choosing to save the file, the destination path and name of the file can be chosen via a pop up file chooser dialog (jFileChooser). When choosing the option “Send list as e-mail”, a pop up window will appear and prompt the user to enter the recipient’s e-mail address. An e-mail containing the attendance list as an attached file will be sent to the specified e-mail address. The sender should be the instructor’s own e-mail address. During development we used our own e-mail addresses for testing purposes.



The scanning has not been started yet.

Figure 4-4: Start-screen with the “File”-option selected

Another option in the menu is “Setup”, where the user has the options to setup a scheduled event, setup a manual event (when an event is not present in the KTH scheduling system), and to enable/disable the option to send a confirmation e-mail to each successfully registered attendee. Figure 4-5 shows the dialog window that appears when the user chooses to setup a scheduled event. The user is prompted to enter the course code (“EQ1120” is an example, as shown in the figure) and the current term (entered in the field period) in order to search for an event. The time interval during which attendance will be taken and the instructor’s KTH id need to be entered via the dialog window.



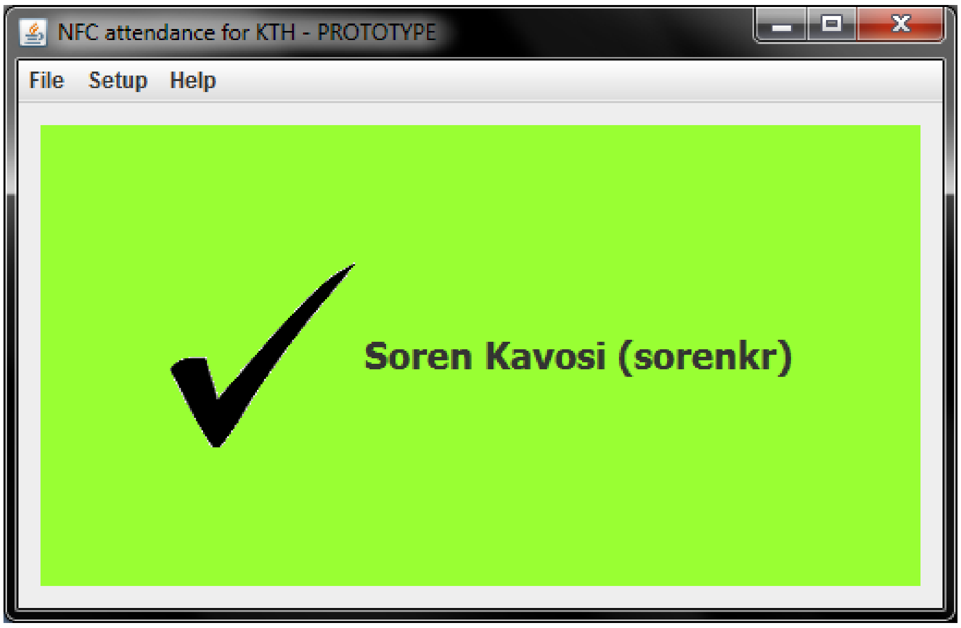
Coursecode: EQ1120
Period: 2015VT

Course:	EQ1120,
Title:	Övning 12,
Type:	Ovn,
Start time:	2015-05-20 13:00:00,
End time:	2015-05-20 15:00:00,
Locations:	Q22, Q26,

Attendance taken from-to: 13:57 14:12
Instructors KTH id: carlj4

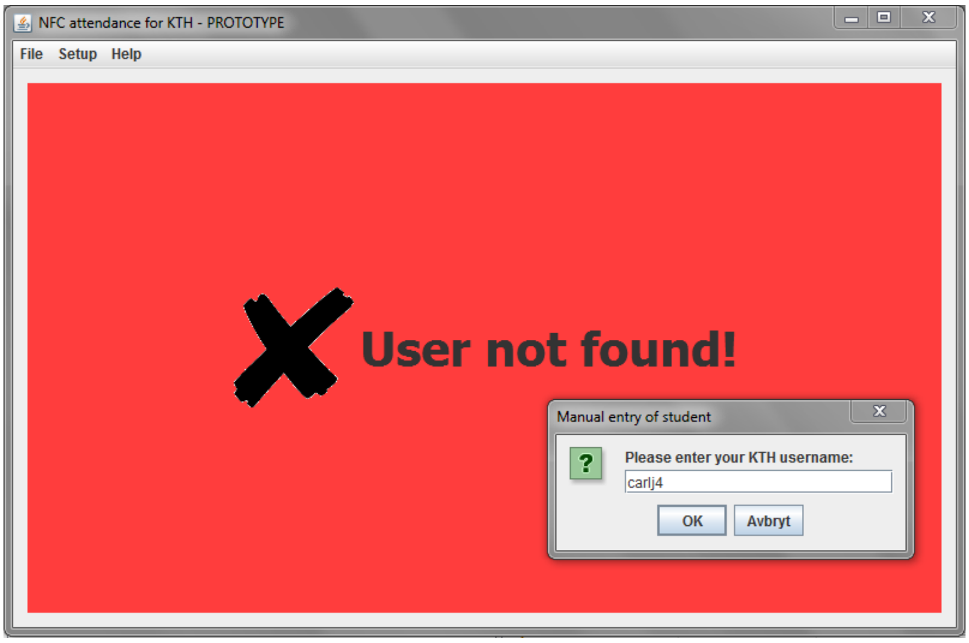
Figure 4-5: Dialog window for setting up a scheduled event

When the user has set up an event, he/she can proceed with the attendance registration process by clicking on the “Start scanning”-option under “File”, as illustrated in Figure 4-4. Once the start-option has been selected, the application and the connected NFC USB-reader are ready to scan RFID cards. Clicking on the “Stop scanning”-option , which replaces the “Start scanning”-option as a menu option when running the scan, again will stop the scanning (not shown in the figure). Pressing start again after stopping the application will start the scanning again from the stopped point, it will not reset any information gathered prior to pressing stop. Figure 4-6 illustrates an successful student registration, while Figure 4-7 illustrates an unsuccessful registration resulting in a dialog box popping up to enable the instructor to manually entry a student’s the KTH-ID (as described in Section 4.1.3.6).



Soren Kavosi (sorenkr)

Figure 4-6: Successful card registration



User not found!

Manual entry of student

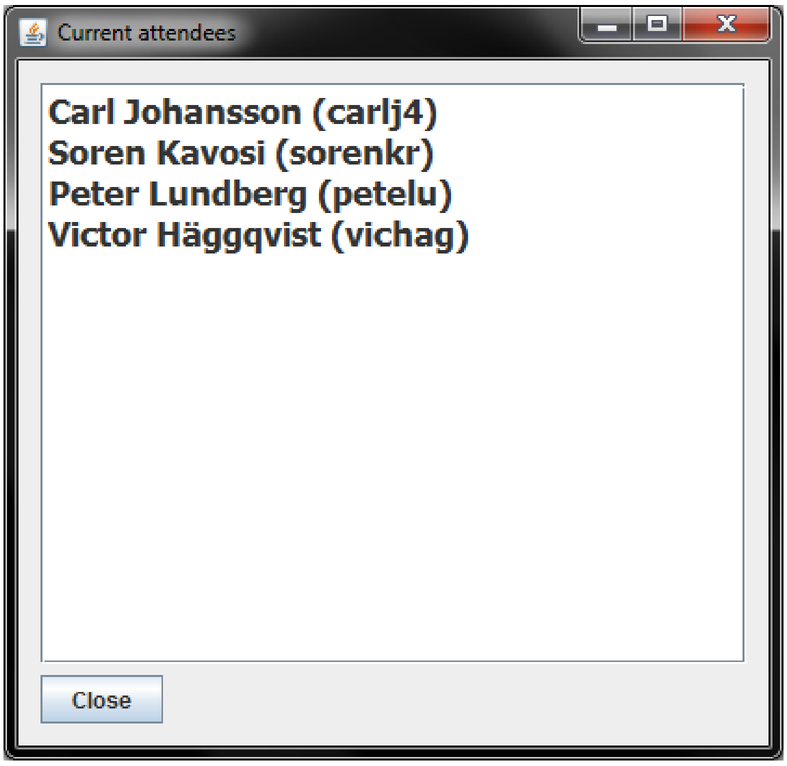
Please enter your KTH username:

carlj4

OK Avbryt

Figure 4-7: Unsuccessful card registration

The last menu option is called “Help” and allows the user to view the current list of attendees in a separate window. Figure 4-8 shows the “current attendees”-display with three registered students as an example. The order of the students is based on their attendance registration time.



Carl Johansson (carlj4)
Soren Kavosi (sorenkr)
Peter Lundberg (petelu)
Victor Häggqvist (vichag)

Figure 4-8: Display list of current attendees*

* These users have explicitly given their permission for their name and KTHID to be shown in this thesis.

5 Analysis

In this chapter, test results are presented and discussed.

5.1 Major results

The results can be divided into two parts: user tests and results of the LDAP lookup time test.

5.1.1 User tests

The feedback gathered from our user tests (presented in Section 4.3.3) were very valuable as they identified some flaws in our application that we had missed during our own testing. One common flaw that was pointed out by different test users was the fact that it was difficult to see if the application was scanning for cards or not. This occurred because the display-text on the start screen did **not** change after the "Start/stop" option was chosen, thus confusing the user. This was corrected by changing the text that is displayed after the Start/stop option is chosen.

Another flaw was that incomplete text in certain parts (menu options, input forms, etc.) of the application resulted in the test users being uncertain as to what kind of information should be entered. A clear example of this problem was the "Save → as txt" option which did not specify what it would save. We assumed that this information was implied, but some test users did not understand that the application would save the attendance list. Another example was the text describing what to type in input forms presented during the processing of the "setup scheduled event"-option. The hover text giving examples of what to enter was not sufficient, perhaps these test users were not actually instructors and thus were not used to entering course codes and semester periods.

The results of our user tests varied a lot depending on the user's background. Test users with less programming and testing knowledge did not provide as much constructive feedback. Moreover they did not test the application that thoroughly and thus did not find many flaws. On the other hand, test users with more programming and testing knowledge provided more explicit feedback. These user were able to push the application to its limits and thus revealed more flaws.

For some of the suggested improvements regarding our application no actions were taken because we did not agree with those particular improvements. One example is the suggestion about changing the display color to yellow when a student who has already been registered tries to register again. We felt that it would be better to keep the two colors green and red, as more colors would potentially complicate the situation for colorblind users. Another example is the suggestion of being able to remove attendees from the current attendance list. We reckoned that this functionality could potentially be misused by individuals with intentions of damaging others (e.g. when there are insufficient reasons for removing a student).

5.1.2 LDAP lookup time test

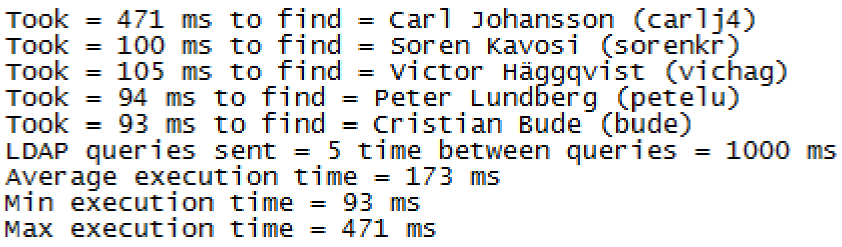
As can be seen in Table 5-1 the LDAP test results varied slightly for the different delay times. The biggest discovery from these tests is the fact that the maximum value is so far above the median, which in turn pushes up the average value to almost twice the median value. This tells us that the distribution of query times has a long tail.

Table 5-1: The test results from the LDAP lookup time tests. All values are in milliseconds (ms) and 5 lookups were done for each "time between lookups" value.

Time between lookups:	1000 ms	10000 ms	20000 ms	60000 ms
Sum	863	850	968	1053
Average	172.6	170	193.6	210.6
Minimum	93	89	109	99
Maximum	471	473	481	616
Median	100	93	127	110

The fact that the first lookup out of the 5 (as can be seen in Figure 5-1) for each delay time always was several times slower than the following 4 lookups. This could potentially mean that the first card that is scanned will not be properly processed and displayed before the next card is scanned.

If other external factors, such as the network being congested, can increase the round trip time of the packets thus increasing the probability that the second student’s card will be displayed only shortly after the first student’s has scanned his/her card. There is code in the application that prevents a new card from being scanned while the prior one is still connected, which means that if the student waits before removing his or her card there will not be a problem.



```
Took = 471 ms to find = Carl Johansson (carlj4)
Took = 100 ms to find = Soren Kavosi (sorenkr)
Took = 105 ms to find = Victor Häggqvist (vichag)
Took = 94 ms to find = Peter Lundberg (petelu)
Took = 93 ms to find = Cristian Bude (bude)
LDAP queries sent = 5 time between queries = 1000 ms
Average execution time = 173 ms
Min execution time = 93 ms
Max execution time = 471 ms
```

Figure 5-1: Sample test output from the LDAP query rate test with a delay of 1000 ms.

As can be seen in Figure 5-2 the time to complete each LDAP lookup follows the same pattern in which the first query takes on average 500 ms, while the second to fifth lookup takes on average 100 ms each. If this pattern were to continue there will be no problem with our application, although the limited number of usernames that we had to use in queries limited the number of subsequent consecutive LDAP lookups that we could made. If we were to have many more usernames to use for the lookups there might be another pattern revealed.

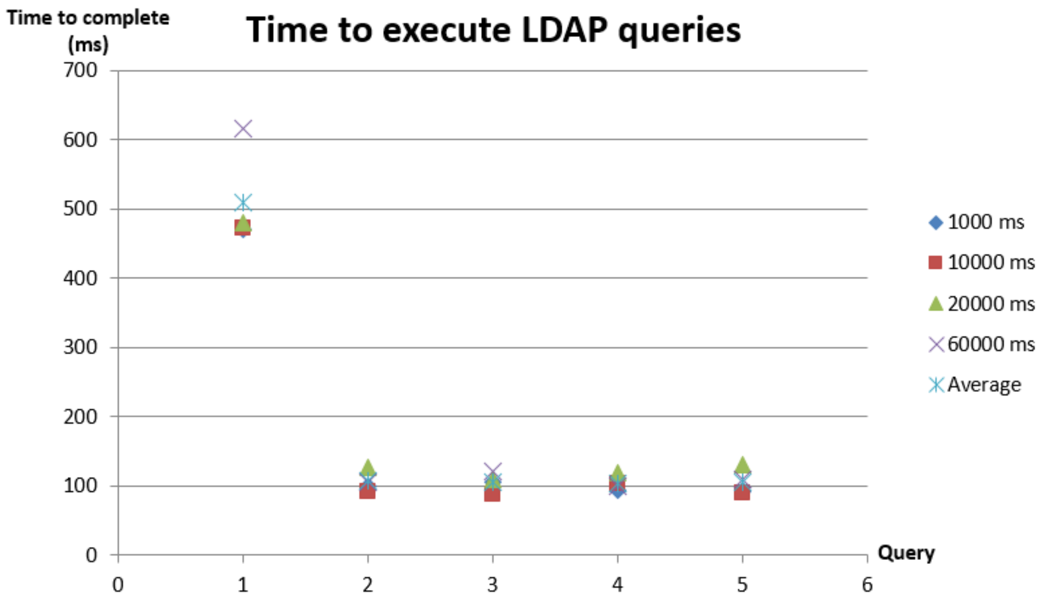


Figure 5-2: The time to complete LDAP lookups with a delay of 1, 10, 20, 60 seconds between lookups

Below in Figure 5-3, one can see that most packets in the first and second lookup takes the same amount of time, one thing to be noted is that the packets in which the first lookup takes longer are all sent by the LDAP server. This gives further material to the idea that it is the authentication to the LDAP server that makes the first lookup take a longer time.

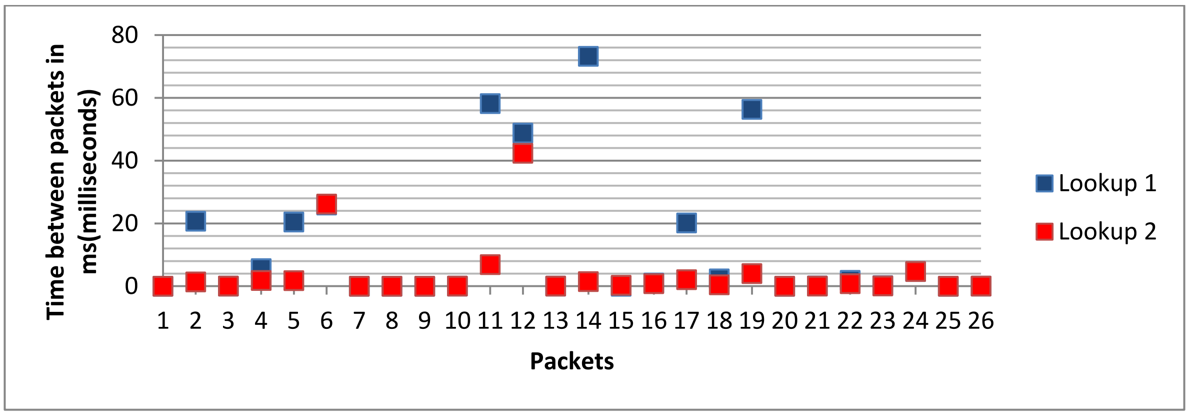


Figure 5-3: Figure depicting the 26 packets sent and the time between them for two LDAP lookups performed after each other (lookup 1 took 340 ms and lookup 2 took 96 ms). Note that this only depicts the lookup versus the KTH LDAP database.

An interesting thing to note is that there seems to be no limit on the number of LDAP queries that can be performed. This was discovered when we tried to do 50 lookups in a row with 1 000 ms between each lookup, all lookups were completed successfully. Although a limit was discovered when tried to extract multiple user's information in a single query – as we once got an error stating that the "Size limit exceeded".

5.2 Reliability Analysis

There are some factors that could affect the query time. These factors include the round trip latency of the connection between the test computer and the LDAP server, the test computer's CPU, the current load on KTH's LDAP server, and the time of day when the tests were made. These are only some of the factors and there are other factors such as network distance which might also affect the tests.

Since the tests were done on a computer connected to an access point inside KTH ICT's wireless local area network, the time difference between queries to the KTH LDAP server should have been routed completely within KTH's network, hence they did not have to go through external nodes to reach their destination. Our LDAP server was running locally on the same machine as the tests were conducted, hence the time to do a local lookup and transfer the data should be minimal, hence the overall test results should describe a future scenario where the KTH LDAP server stores the access card's UID and only one lookup is needed.

Four consecutive tests were made starting with a delay of 1 000 ms and ending with 60 000 ms *without* the GUI. As mentioned in Section 4.3.4, the test-method used our java methods that executed LDAP queries with an access card UID to lookup the KTH username based upon this UID and then a LDAP query was made to lookup information about the student based on their KTH username. The UID was **not** obtained via scanning of card, but instead was built into the test script, hence there were no delays due to accessing the NFC reader or the RFID card. This means that the additional delay of the communication between the reader and the cards were excluded from the test. For reliability purposes the tests were done with the same five users in all the tests.

The user tests of the application also considered reliability since the users performed the normal behavior as described in Section 3.4.

5.3 Validity Analysis

The data gathered are considered valid since, as previously described in Section 5.1.2, a time stamp is collected before and after the LDAP queries and then compared to the time stamp after the responses have been received. This time is recorded and printed to the console. As we pointed out in Section 4.3.4, the time between queries is recorded, thus we can calculate the average, minimum, and maximum query response times. The output containing this information is considered valid as the time measurements were in units of nanoseconds, and then stored in units of milliseconds to allow for easier comparisons.

5.4 Discussion

One interesting question is: Will the system scale for other universities with larger numbers of students and faculty? The number of students and faculty can vary quite a lot depending on the university. For example, Stockholms University has 70 000 students and 5 000 staff[*], New York University has 58 547 students and 4 572 faculty[†], and California State University has 460 000 students and 47 000 faculty and staff[‡]. These numbers are much larger that the corresponding numbers for KTH ICT, where the total number of students was 1 301 and the total number of faculty & staff were 321, in 2014 (as described previously in Section 3.2.3).

[*] http://www.su.se/english/about/facts-figures
[†] http://www.nyu.edu/about/news-publications/nyu-at-a-glance.html
[‡] http://www.calstate.edu/PA/2015facts/

The high numbers of students and faculty at other universities should not be a problem with regard to scalability because these numbers will not affect the LDAP queries made from our application, unless there are limitations on how many nearly simultaneous LDAP queries can be made. The number of attendees at any lecture/lab/seminar is normally affected by the maximum number of students that can participate in a given course during a certain period. This is in turn affected by the specific school's regulations and room capacities. As mentioned in Section 5.1.2, we made a test with 50 lookups in a row with 1 000 ms between each lookup and the application continued to behave in the expected manner. It is clear that 100 or 200 lookups per second could occur in schools with more students and faculty.

The pattern of lookup times shown in Figure 5-2, illustrated and described in Section 5.1.2, supports our argument of the application being able to operate with larger numbers of students. An average lookup time of about 100 ms should not be a problem since it is sufficiently fast so that most humans will not notice any delay. We doubt that two subsequent card scans and lookups could take place in under 100 ms. Of course there could be a similar student registration taking place in one of KTH's 264 rooms[*].

Compared to some of the other relevant work in Table 2-1, our prototype has the advantage of not requiring students to have an NFC-enabled phone or any application installed. It only requires students to have a KTH access card, which is not a problem at KTH ICT since all students possess one. The downside is that this solution assumes that student have access cards with unique IDs, which they have at KTH ICT, but this might not be the case at other universities. Our solution has the major disadvantage of not being integrated with any administrative system and only being able to save attendance lists locally as a txt or xml-file. Although the list can be send as an e-mail attachment.

[*] http://www.kth.se/places/room

6 Conclusions and Future work

This final chapter summarizes and concludes this thesis project as well as describes the limitations of this thesis project and gives suggestions for future work. At the end of the chapter, reflections on relevant economic, social, environmental, and ethical aspects of our work are presented.

6.1 Conclusions

Over time we noticed that we would be unable to reach all our goals (as specified in Section 1.4) within the planned time period, hence we choose to focus on the first part of our goals: "... develop and evaluate an prototype application that creates an attendance list by using a NFC reader to read access card data.". This goal was reached and a functional prototype was created. This prototype can read the UID of an access card. Given this UID a lookup can be made to get information about the student associated with this card. The second major goal, to enable uploading of the attendance list to other KTH services remains for future work (see Section 6.3).

We have gained multiple insights during this project that have further increased our knowledge, some of these insights are:

- Our knowledge of Java has been increased as we learnt about LDAP communication from within Java. We also learned how to communicate with an USB NFC Reader using Java with APDUs, how to send mail using Java with javax.mail, and how to read JSON in Java with javax.json.
- We have gained new knowledge and extended our knowledge of RFID and NFC during this thesis project.
- We have learned how to setup OpenLDAP within a Linux environment and how to create and add new schemas to the LDAP database. Additionally, we gained some knowledge about how to use LDAP related utilities, such as slaptest and ldapadd.

The most important suggestion we have for others working in this area in the future would be to do a major background study - as there are many other people doing similar things and you should be able to gain valuable information and inspiration from their work.

As the project nears its end we only have one regret: we did not reach out to all the people that could help us within the first week. Waiting to ask until when we needed their help leads to delays in the project and eliminated the chance to change directions during the thesis project. For example, if we had gotten the information about Daisy earlier (see the first paragraph of Section 6.3) we might have changed direction to focus on the upload of the attendance information to the system, rather than retrieving attendance information from the students.

6.2 Limitations

One of the main goals of this thesis was to create a functional prototype of an application that could create an attendance list and easily transfer it to administrative systems such as Daisy (described in Section 1.4). Unfortunately, the developers of Daisy have not implemented such a function and we would have needed to learn how to integrate the application with the Daisy API ourselves. Unfortunately, this was not possible due to lack of time and a lack of knowledge. This will be further described in the next section.

There were other limitations as well, one of them is that we only had 10 test cards. These cards included our own two KTH access cards, two additional cards generously donated to us by Fidesmo[*], and the last 6 cards were a part of a development kit bought with our own money. Since we both are students with tight budgets, the option to buy additional cards was unavailable. The small number of test cards limited our tests in terms of the number of unique queries that we could make.

Other limitations of our results included:

- The application was not properly tested on operative systems other than Microsoft's Windows.
- The application requires a PC and a NFC USB ACR122U reader.

These limitations will further be described in the next section.

6.3 Future work

The most important future work would be the ability to directly upload attendance lists from the application into KTH's course administration systems, such as Bilda, Daisy, Social, etc. Initially, there were plans to upload the attendance lists into Daisy with the help of the Daisy developers; however, they did not consider this a high priority to add to Daisy [33]. According to one developer we could have gotten access to the private Daisy API and built a JavaScript function to automate the checking of the checkboxes that represent course registrations in Daisy. If someone were to look at this in the future, we would suggest that they start there. One of our hopes is given that the development of this prototype as a proof-of-concept, KTH administrators will support the required extension to Bilda, Daisy, Social, etc.

As the application has been built in Java it should theoretically run on both Linux and Mac OS X. However, the application has not been tested in these environments due to lack of access to these operating systems. The application has been tested once on Linux, but has not be tested again since the last changes were made. If KTH were to decide to use this prototype as a base for an attendance system we would recommend that the application should be properly tested in both Linux and Mac OS X environments before changes to the code are made.

Something that would simplify user's life in the future would be an Android, iOS, and/or Windows Phone port of the application to enable staff to take attendance without needing a laptop or stationary computer and a USB connected NFC Reader. This would be a useful feature as many of the staff might not want to drag around extra equipment. The parts of the application that would need to be modified or re-implemented would then be the GUI. Additionally, some adjustments would have to be made to the CardHandler.java class to make it use the Android/iOS/Windows Phone API.

Currently, the application only supports one specific USB NFC Reader, the ACR122U. A clear future effort would be to extend the application to check what type of NFC reader is currently connected. Then the application would be able to choose the correct method for reading cards for that type of reader. This is necessary as different readers use different command APDUs to communicate with the cards and the format of the response might differ between different readers. This would not be needed if KTH adopted a standard hardware platform which they provided to all instructors for collecting attendance lists through NFC.

[*] http://fidesmo.com/sv/

An interesting adaptation of the application would be to add support for the RFID locks currently deployed on the doors on the lecture halls in the Electrum building (in the rooms used by KTH ICT). This could possibly enable a teacher to swipe his/her card and have the RFID lock take attendance automatically and then e-mail the attendance list to the instructor. This would require either KTH to rebuild the RFID lock system from scratch or work out some kind of arrangement with the vendor to enable the use of plugins for these locks.

6.4 Reflections

If our prototype were to be further developed and integrated with appropriate administrative systems at KTH, there could be economic benefits for the school - as such a system would reduce the time and resources needed by instructors and administrators at KTH. It could also reduce the time needed to register attendance manually from sheets of paper, as currently that paper has to circulate around the classroom (with the risk of not be properly passed to all students) and reduces the focus on the current lecture, lab, or seminar.

The prototype as it is currently implemented (i.e., without integration with any administrative system) could possibly improve several ethical aspects regarding attendance registration. To passing around a piece of paper to collected attendance at events has the potential risk of forgery - as it is hard to detect that one student is signing the name of his/her friend who is not present at the event. By using our prototype, students would simply place their KTH access card on the NFC reader connected to the instructor's PC. If a student attempts forgery, he/she would have to possess his or her friend's KTH access card and would need to scan two different cards (once for themselves and once for their friend) and this is easier for the instructor or other attendees to notice.

The environmental aspects regarding this thesis are also interesting. Implementation of our prototype would require purchasing of NFC USB readers which has a negative impact on the environment since production and transportation of these device's requires additional energy and material. However, the fact that our prototype would be used rather than paper lists (which require a lot of paper) could possibly be a benefit from an environmental point of view. Also, running our application would not increase computer usage since instructors at KTH ICT generally use computers in some way to present information at lectures, labs, seminars, etc.

References

[1] D. Graziano, “NFC-equipped smartphone sales ballooned 300% in 2012 even without Apple’s support,” *BGR*. 20-Jun-2013 [Online]. Available: http://bgr.com/2013/06/20/nfc-smartphone-adoption-2012/. [Accessed: 10-Apr-2015]

[2] R. Boden, “Berg Insight reports on NFC POS terminal growth • NFC World+,” *NFC World+*. Feb-2015 [Online]. Available: http://www.nfcworld.com/2015/02/24/334209/berg-insight-reports-on-nfc-pos-terminal-growth/. [Accessed: 10-Apr-2015]

[3] Google, “An easy way to pay, purchase, and save – Google Wallet.” [Online]. Available: https://www.google.com/wallet/. [Accessed: 10-Apr-2015]

[4] M. Tapper, “FW: Genomförande av exjobb - accesskort,” 03-Feb-2015 [Online]. Available: https://www.dropbox.com/s/r5u2hqihdxh14q9/FW_%20Genomf%C3%B6rande%20av%20exjobb%20-%20accesskort%20-%20Soren%20Kavosi.pdf?dl=0

[5] A. Lundgren, “Anvisning om hantering av förteckningar över studenter.” KTH Universitetsförvaltning, 16-Apr-2015 [Online]. Available: https://www.dropbox.com/s/lbujsq7ix580ou1/SUF_RH_PLAN15042012190.pdf?dl=0

[6] C. Larman, “Iterative Development,” in *Agile and Iterative Development: A Manager’s Guide*, Addison-Wesley Professional, 2003, ISBN-10: 0-13-111155-8 ISBN-13: 978-0-13-111155-4 [Online]. Available: http://proquest.safaribooksonline.com.focus.lib.kth.se/book/software-engineering-and-development/agile-development/0131111558/iterative-and-evolutionary/ch02lev1sec1. [Accessed: 14-Apr-2015]

[7] J. Landt, “The history of RFID,” *IEEE Potentials*, vol. 24, no. 4, pp. 8–11, Oct. 2005.

[8] I. Poole, “RFID Frequencies | RFID Frequency Bands & Spectrum | Allocations,” *Radio-Electronics.com*. [Online]. Available: http://www.radio-electronics.com/info/wireless/radio-frequency-identification-rfid/low-high-frequency-bands-frequencies.php. [Accessed: 27-Mar-2015]

[9] V. D. Hunt, M. Puglia, and A. Puglia, *RFID - A Guide to Radio Frequency Identification*. Hoboken John Wiley & Sons, Inc, 2007, ISBN-13: 978-1-4493-7206-4 [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/9780470112250.ch1/summary

[10] J. Thrasher, “RFID versus NFC: What’s the difference between NFC and RFID?,” *RFID insider*. 11-Oct-2013 [Online]. Available: http://blog.atlasrfidstore.com/rfid-vs-nfc. [Accessed: 24-Mar-2015]

[11] NXP Semiconductors, “MIFARE smart card ICs :: NXP Semiconductors.” [Online]. Available: http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/. [Accessed: 02-May-2015]

[12] NFC Forum, “What Is NFC? - About the Technology,” *NFC Forum*. [Online]. Available: http://nfc-forum.org/what-is-nfc/about-the-technology/. [Accessed: 01-Jun-2015]

[13] T. Igoe, D. Coleman, and B. Jepson, “The Architecture of NFC,” in *Beginning NFC*, O’Reilly Media, Inc., 2014, ISBN-10: 1-4493-0851-1 ISBN-13: 978-1-4493-0851-3 [Online]. Available: http://shop.oreilly.com/product/0636920021193.do. [Accessed: 02-Apr-2015]

[14] Sony, “Sony Global - FeliCa - Overview of FeliCa - What is FeliCa ?” [Online]. Available: http://www.sony.net/Products/felica/about/. [Accessed: 14-Apr-2015]

[15] T. Igoe, D. Coleman, and B. Jepson, “How NFC Operates,” in *Beginning NFC*, O’Reilly Media, Inc., 2014, ISBN-10: 1-4493-0851-1 ISBN-13: 978-1-4493-0851-3 [Online].

Available: http://shop.oreilly.com/product/0636920021193.do. [Accessed: 19-Apr-2015]

[16] Vedat Coskun, Kerem Ok, and B. Ozdenizci, “NFC Essentials,” in *Professional NFC Application Development for Android*, John Wiley & Sons, 2013, ISBN-13: 9781118380543, pp. 10–23 [Online]. Available: http://site.ebrary.com.focus.lib.kth.se/lib/kth/detail.action?docID=10684960

[17] NFC Forum, “What Is NFC? - What It Does,” *NFC Forum*. [Online]. Available: http://nfc-forum.org/what-is-nfc/what-it-does/. [Accessed: 21-Apr-2015]

[18] NFC Forum, “NFC Forum Specification Architecture - Tag Type Technical Specifications,” *NFC Forum*. [Online]. Available: http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/tag-type-technical-specifications/. [Accessed: 01-Jun-2015]

[19] C. Kern, “New Report Forecasts Major Growth In NFC Enabled Handsets And NFC Transaction Market By 2019.” [Online]. Available: http://www.retailsolutionsonline.com/doc/new-report-forecasts-major-growth-in-nfc-enabled-handsets-and-nfc-transaction-market-by-0001. [Accessed: 02-Apr-2015]

[20] Transport for London, “Contactless - Transport for London.” [Online]. Available: http://www.tfl.gov.uk/fares-and-payments/contactless?cid=contactless. [Accessed: 21-Apr-2015]

[21] Jim Sermersheim <jimse@novell.com>, “Lightweight Directory Access Protocol (LDAP): The Protocol.” [Online]. Available: https://tools.ietf.org/html/rfc4511. [Accessed: 30-Mar-2015]

[22] Zytrax, “Open Source Guide - LDAP for Rocket Scientists.” [Online]. Available: http://www.zytrax.com/books/ldap/. [Accessed: 30-Mar-2015]

[23] B. Benyo, B. Sodor, T. Doktor, and G. Fordos, “Student attendance monitoring at the university using NFC,” presented at the Wireless Telecommunications Symposium (WTS), 2012, London, 2012, ISBN-13: 978-1-4577-0579-3, pp. 1–5 [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6266137

[24] M. A. Ayu and B. I. Ahmad, “TouchIn: An NFC Supported Attendance System in a University Environment,” *Int. J. Inf. Educ. Technol.*, vol. 4, no. 5, pp. 448–453, Oct. 2014.

[25] M. J. L. Fernandez, J. G. Fernandez, S. R. Aguilar, and B. Selvi, “Control of attendance applied in higher education through mobile NFC technologies,” *Expert Syst. Appl.*, vol. 40, no. 11, pp. 4478–4489, Sep. 2013.

[26] M. Bucicoiu and N. Tapus, “Easy attendance: location-based authentication for students integrated with Moodle,” presented at the 2013 11th RoEduNet International Conference, 2013, ISBN-13: 978-1-4673-6114-9, pp. 1–4 [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6511761

[27] Moodle, “Moodle - Open-source learning platform.” [Online]. Available: https://moodle.org/. [Accessed: 27-Apr-2015]

[28] KTH, “KTH | Skolan i siffror.” [Online]. Available: https://www.kth.se/ict/om/skolan-i-siffror-1.7724. [Accessed: 03-May-2015]

[29] Datainspektionen, “Personuppgiftslagen - Datainspektionen.” [Online]. Available: http://www.datainspektionen.se/lagar-och-regler/personuppgiftslagen/. [Accessed: 14-May-2015]

[30] OpenLDAP, “Manual Pages - ldapadd(1),” 24-Nov-2011. [Online]. Available: http://www.openldap.org/software/man.cgi?query=ldapadd&sektion=1&manpath=OpenLDAP+2.4-Release. [Accessed: 22-May-2015]

[31] OpenLDAP, “Manual Pages - slaptest,” 24-Nov-2011. [Online]. Available: http://www.openldap.org/software/man.cgi?query=slaptest&manpath=OpenLDAP+2.4-Release. [Accessed: 22-May-2015]

[32] 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/. [Accessed: 28-Apr-2015]
[33] N. Dimitrakas, “Re: Hjälp med exjobb - implementation av Daisy,” 07-May-2015 [Online]. Available: https://www.dropbox.com/s/0tsmddfh9lwhmjw/Re_%20Hj%C3%A4lp%20med%20exjobb%20-%20implementation%20av%20Daisy%20-%20Soren%20Kavosi.pdf?dl=0
[34] Oracle, “Java™ Smart Card I/O API,” *javax.smartcardio (Java Smart Card I/O)*, 26-Sep-2014. [Online]. Available: https://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html. [Accessed: 18-May-2015]
[35] Advanced Card Systems Ltd., “ACR122U Application Programming Interface V2.02.” Advanced Card Systems Ltd., 20-Dec-2012 [Online]. Available: http://downloads.acs.com.hk/drivers/en/API-ACR122U-2.02.pdf. [Accessed: 18-May-2015]
[36] J. Ellingwood, “How To Install and Configure OpenLDAP and phpLDAPadmin on an Ubuntu 14.04 Server,” *DigitalOcean*, 05-Jun-2014. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-openldap-and-phpldapadmin-on-an-ubuntu-14-04-server. [Accessed: 24-May-2015]
[37] stezz, “somethingMeaningful: How to add a new schema to openLDAP 2.4+.” [Online]. Available: http://stezz.blogspot.se/2012/05/how-to-add-new-schema-to-openldap-24.html. [Accessed: 25-Mar-2015]

Appendix A: Creating the new LDAP database

A.1: Installing OpenLDAP (on Ubuntu 14.04)

```
sudo apt-get update
sudo apt-get install slapd ldap-utils
sudo dpkg-reconfigure slapd
```

- Omit OpenLDAP server configuration? **No**
- DNS domain name? **ldap.local**
- Organization name? **ldap**
- Administrator password? **ldap**
- Database backend? **HDB**
- Remove the database when slapd is purged? **No**
- Move old database? **Yes**
- Allow LDAPv2 protocol? **No**

A.2: Creating the OU used with ldapadd

create a file with: "vim carduserou.ldif" and add:

```
dn: ou=carduser,dc=ldap,dc=local
objectclass: organizationalUnit
ou: carduser
```

Use "ldapadd -xWvD cn=admin,dc=ldap,dc=local -f carduserou.ldif" to add the OU to LDAP.

A.3: Adding a schema with ldapadd utility and slaptest utility

Create the schema and name it carduser.schema, in our case we used the one shown in the paper.

```
mkdir /tmp/lidf/
echo "include /etc/ldap/schema/carduser.schema" > schema_conv.conf
slaptest -f ./schema_conv.conf -F /tmp/ldif/
vim /tmp/ldif/cn\=config/cn\=schema/cn\=\{0\}carduser.ldif
```

Change the following lines to this:

```
dn: cn=carduser,cn=schema,cn=config,
objectClass: olcSchemaConfig
cn: carduser
```

Also remove the following lines from the file (* is any text):

```
structuralObjectClass:*
entryUUID: *
creatorsName: *
createTimestamp: *
entryCSN: *
```

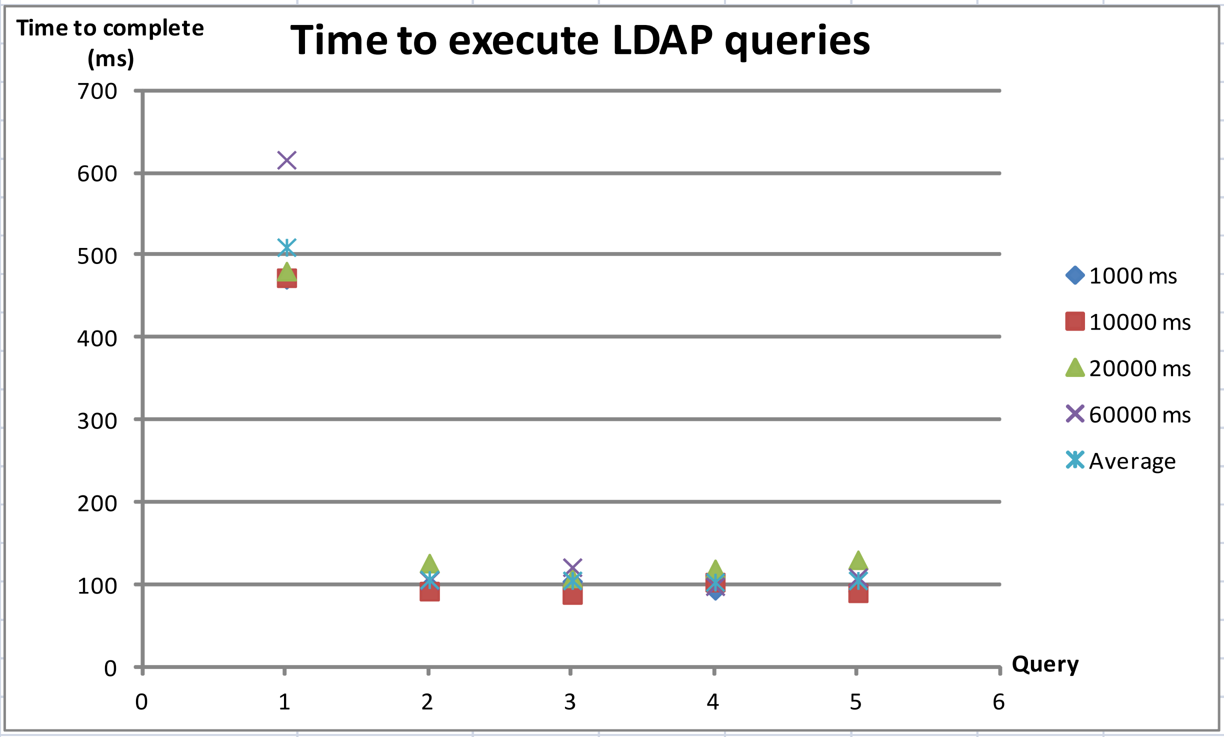
```
        modifiersName: *
        modifyTimestamp: *
cp /tmp/ldif/cn\=config/cn\=schema/cn\=\{0\}carduser.ldif /etc/ldap/schema/carduser.ldif
ldapadd -Q -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/carduser.ldif
```

Based on information from a community tutorial on Digital Ocean[36], and a blogpost on SomethingMeaningful(stezz.blogspot.com)[37].

Appendix B: Detailed results

Lookup:	1000 ms	10000 ms	20000 ms	60000 ms	Average
1	471	473	481	616	510
2	100	93	127	106	107
3	105	89	109	122	106
4	94	104	120	99	104
5	93	91	131	110	106

Time between lookups:	1000 ms	10000 ms	20000 ms	60000 ms
Sum:	863	850	968	1053
Average:	172,6	170	193,6	210,6
Min:	93	89	109	99
Max:	471	473	481	616
Median:	100	93	127	110



Appendix C: Source code

This appendix gives a listing of the source code. The source code is available for download from the DiVA repository where this thesis is stored.

The .zip-archive contains the following classes:

- CardHandler.java
- Controller.java
- CurrentAttendeesDisplay.java
- CustomEventConfigurator.java
- LDAPQuery.java
- MailHandler.java
- Precense.java
- SaveHandler.java
- ScheduledEventConfigurator.java
- SchemaEvent.java,
- SchemaHandler.java,
- Visitor.java
- Window.java

TRITA-ICT-EX-2015:47