School of Engineering Science
Simon Fraser University
Burnaby, BC V5A 1S6
wangyiz@sfu.ca

November 6, 2014

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 305W/440W Design Specification for the Smart Locker

Dear Dr. Rawicz:

Attached is a document from Tap Lock Inc. describing the Design specification for the Smart Locker. Our company is designing and implementing a smart locking system that can be controlled by smart phones via NFC (Near Field Communication). The whole system is consisted of a physical locker and a self-developed android-based application. It can assign access automatically and allow each user to share access just by tapping two smart phones together.

The functional specifications provides all requirements for the entire system. This document will be used by Tap Lock Inc.'s project manager and designers as a reference and design guide throughout the research and development process.

Tap Lock Inc. consists of five intelligent, creative and motivated third and fourth-year engineering students: Wangyi Zhu, Yangyang Li, Kaiqi Li, Haishuo Zhang and Zheng Gao. If you have any questions or concerns about our project, please feel free to contact me by phone at (604)-767-8090 or by email at wangyiz@sfu.ca.

Sincerely,

Wangyi Zhu
President and CEO
Tap Lock Inc.

Enclosure: Functional Specification for the Smart Locker

Design Specification for a

Smart Locker

Project Team:

Wangyi Zhu
Yangyang Li
Haishuo Zhang
Kaiqi Li
Zheng Gao

Contact Person:

Wangyi Zhu
wangyiz@sfu.ca

Submitted to:

Dr. Andrew Rawicz – ENSC 440W
Steve Whitmore – ENSC 305W
School of Engineering Science
Simon Fraser University

Issue Date:

November 6, 2014

Revision:

1.5

Executive Summary

At the beginning of each semester in some universities like SFU, students have to finish several steps to request a locker, by sending an email. However, the existing system is very complicated and time-consuming: after sending out an email, students have to wait for administrator's approve to set up and obtain the locker's password.

As technology developed, many advanced locking systems have been created to simplify people's life, such as RFID lockers. Although this innovation has introduced a new generation of electronic locking system, it only provides an easier way of opening and locking a locker. In other word, the existing locking system in the market is not intelligent enough to improve users' experience and maximize the power of technology. The Smart Locker from Tap Lock Inc. seeks to provide users with a multifunctional, easy-to-use and very secure electronic locking system via NFC. The user can access the locker by just tapping their smart phones to the locker. In addition to this easy-to-use function, users can also request and share access through smart phone application. Moreover, user information and locker identities are pre-recorded in database for security purpose.

There are two phases of our Smart Locker development. The first phase includes functions as shown below:

- Saving user information in database
- Saving locker identity in database
- Assigning access automatically
- Opening and locking the locker via NFC
- Sharing access via NFC by tapping two smart phones together
- Self- alarm system

All operations above are controlled through an easy-to-use Android application. This application can increase efficiency and security of user experience. This prototype phase is planned to be finished in a four-month development cycle and to be completed on December 6, 2014.

In the second phase of development, we aim to extend usage of this electronic system. For example, students can use our Android application to access school's printers. In addition, we suspect to develop contactless payment via NFC instead of using a lab printing card. Tap Lock Inc. is committed to create an intelligent system that not only allow user to easily open lockers but also simplify the school life in various ways.

Table of Contents

List of Tables

Table of Figures

Abbreviations

NFC	Near Field Communication
NDEF	NFC Data Exchange Format
HCE	Hose-based Card Emulation
RFID	Radio-frequency identification
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
IC	Integrated Circuit
I/O	Input/output
LED	Light-emitting diode
SPI	Serial Peripheral Interface
SQL	Structured Query Language

Glossary

4 byte positive integer	A specific number that assigned to each locker. Also saved in database for identification.
Trigger	Smartphone's build-in NFC chip are inducted and controlled through our application
NDEF	One kind of NFC forum data format
ID	The specific identification number assigned to each locker and stored in database
Near Field	Devices are very closed to each other's antenna, within a distance of 0.1m
HCE	Emulating an NFC Tag with an Android Device without a secure element

1. Introduction

The Smart Locker is an electronic locking system which allows the user to control the closure through smart phones via NFC. Once the user finished registration and login process, he/she can start to request a locker. Once authorized, the users are able to open the locker by tapping devices on the locker. Moreover, primary user has rights to share access to whomever they want by tapping two smart phones together. For security purpose, user's information and locker's identities are saved in database. The whole system consists of three main components: an electronic locker with build-in programmable NFC chip, a server and an Android-based mobile application. Detailed requirements for the Smart Locker, proposed by Tap Lock Inc., are described in this design specification.

1.1 Scope

This document describes the design specifications for our proof-of-concept model for the Smart Locker. Detailed explanations of design features will be provided in this document as well. This design specifications is expected for use in future development and improvement of the product.

1.2 Intended audience

The design specification is intended to use by all members of Tap Lock Inc. The project manager shall refer to the design specification as a general design guidelines to measure designing progress during development period. Design engineers shall refer to the requirements as overall design goals and foresee the potential risks involved in implementation. Test engineers shall use this document as guidance in troubleshooting and debugging during testing period.

2. First Phase – Prototype

As introduced, the Smart Locker is an electronic locking system which is based on NFC technology. Since smart phones are almost available to everyone nowadays [1], we choose to develop an application and provide different functionalities. The smart phones will act as a key so that the user shall no longer remember a password. Basically, our prototype is designed to simplify a traditional locking system and provides more functionalities such as sharing access and automatic registration. Moreover, the prototype will be single cell of the locker. It is no need to create multiple cells of locker because all the cells are stand alone and identical to each other. Although safety is the priority of Smart Locker, the prototype will be made by wood because we just need to test and show functional capabilities.

3. Second Phase - Final Product

The Smart Locker is our company's first prototype. In the future, we propose to design a customized system for universities which not only provides students with access to lockers but also assign access to lab entrance and other electronic equipment such as printers. Furthermore, we aim to develop our application to supply more functionalities such as contactless payment.

Through this, students can use smartphones to make a payment for using lab equipment. The material for locker will be some strong metal to make sure it is safe enough to store things. The size of locker can depend on user because user can put several cells into a large block of lockers. The final product must have some connection between cells to make sure it is enough strong and will not fall or course something bad.

4. General System Overview

Generally, the Smart Locker consists of three main components: a server, a smart phone application and an electronic locker. In the whole system, the locker component is the only physical component which including an NFC tag, an NFC Shield on Arduino and a metallic lock. The rest parts are all software based. There will be an admin to manage locker usage. Admin will have the highest control of the system. If anything bad happen, user should contact admin immediately.

4.1 System Block Diagram

Figure 1 below is a block diagram that summarizes our system. The user needs to login in the mobile application to access the system. For instance, if the user wants to request a locker, he/she has to login first and then choose the "request a locker" function in the application. Then, the application will verify the user identity with database and check for availability. Finally, the server will send a feedback to tell whether the request is permitted or denied.

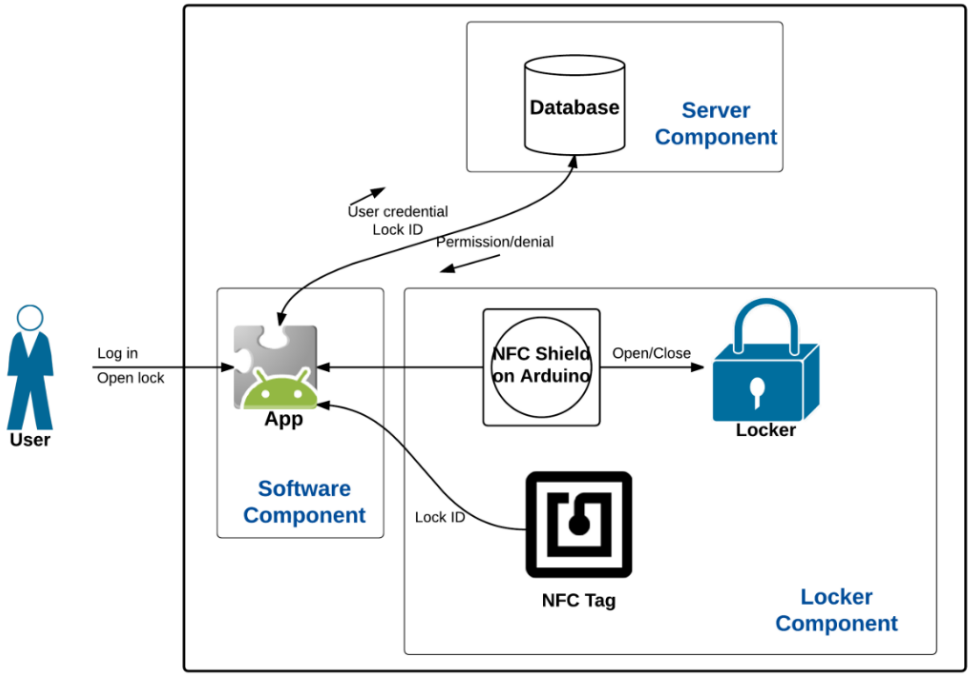

Figure 1: Smart Locker Functional Block Diagram

4.2 Prototype Model

Physically, the Smart Locker is built with an NFC Tag, an NFC reader, an Arduino microcontroller and a metallic lock. The rest part are all software-based, including a database and a smartphone application. The data is transferred through a Near-Field between lockers and smartphones. The Near-Field Communication is established automatically once the NFC tags are close enough to each other, usually under 4 centimeters [2]. The smartphones and database is connected through Wi-Fi. Figure 2 shows how data is exchanged among NFC tags, NFC readers, smartphones and database.

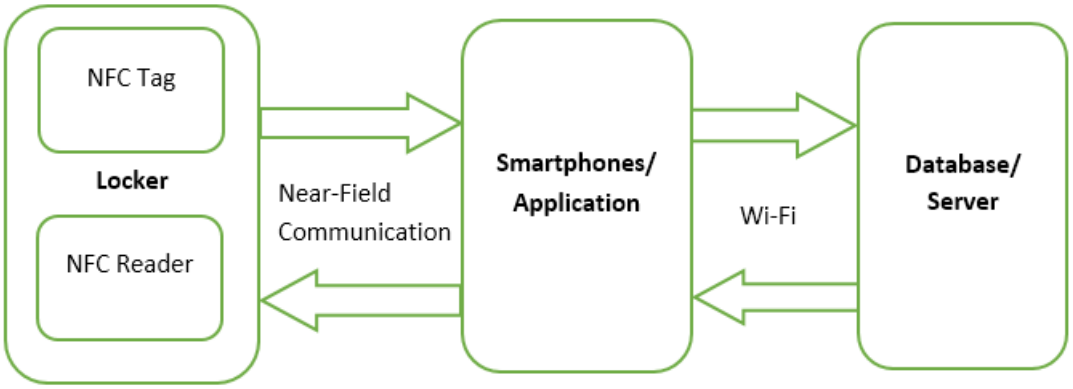

Figure 2: Communication between Locker, Smartphones and Database

4.3 System Flowchart

The previous section is a briefly introduction how the Smart Locker System is built. The following flowchart in Figure 3, Figure 4 and Figure 5 will give a detailed explanation of how the Smart Locker system is working. Our system is designed with three main functions for user. One of the main functions is to rent a locker automatically through the smartphone application.

First of all, the user needs to login if already has an account, otherwise registration is required. This is designed to protect user privacy and increase security of the application. Once the user has successfully login, he/she can do various tasks through the interface. Figure 3 shows the procedure that how users login into our system.

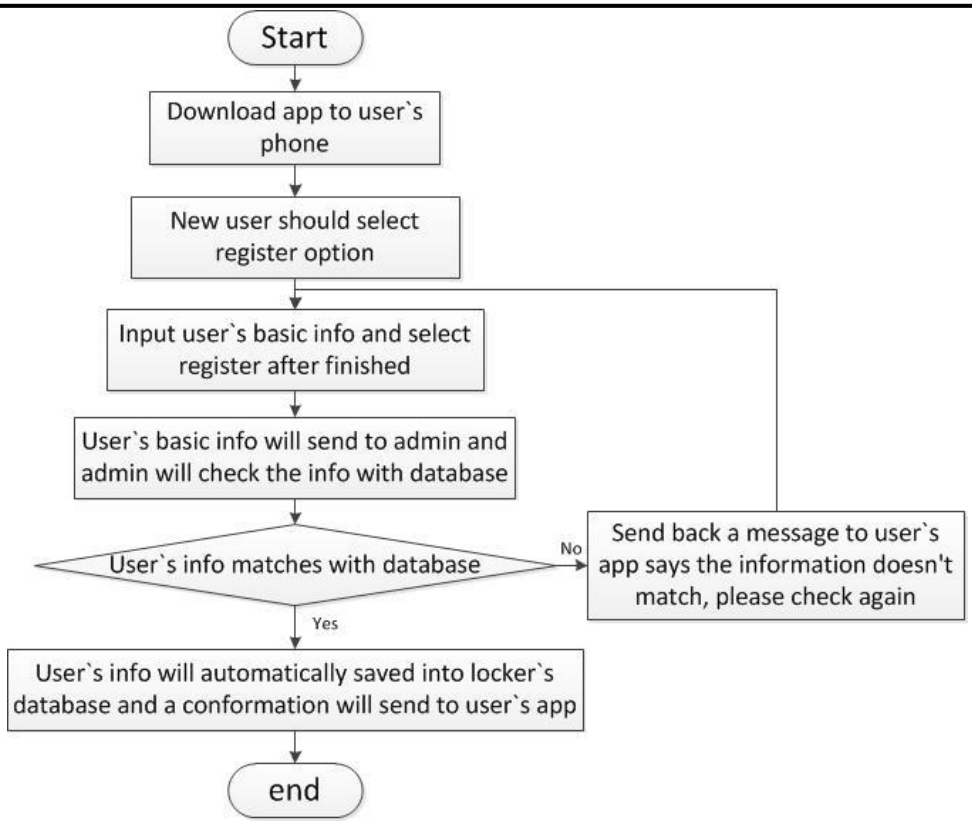

Figure 3: Flowchart of User Registration

Once the users are registered in our system. He/she can start to do various tasks through our application. The Smart Locker also allows user to choose their preference. For example, in universities, a student can decide whether to rent a locker in gym or library. User's preference will be checked in database and then a feedback will be sent back to the user. If there is no locker meets user's requirement, the system will randomly assign a locker to the user. In Figure 4, it shows a step by step procedure of renting a locker through our smart phone application.

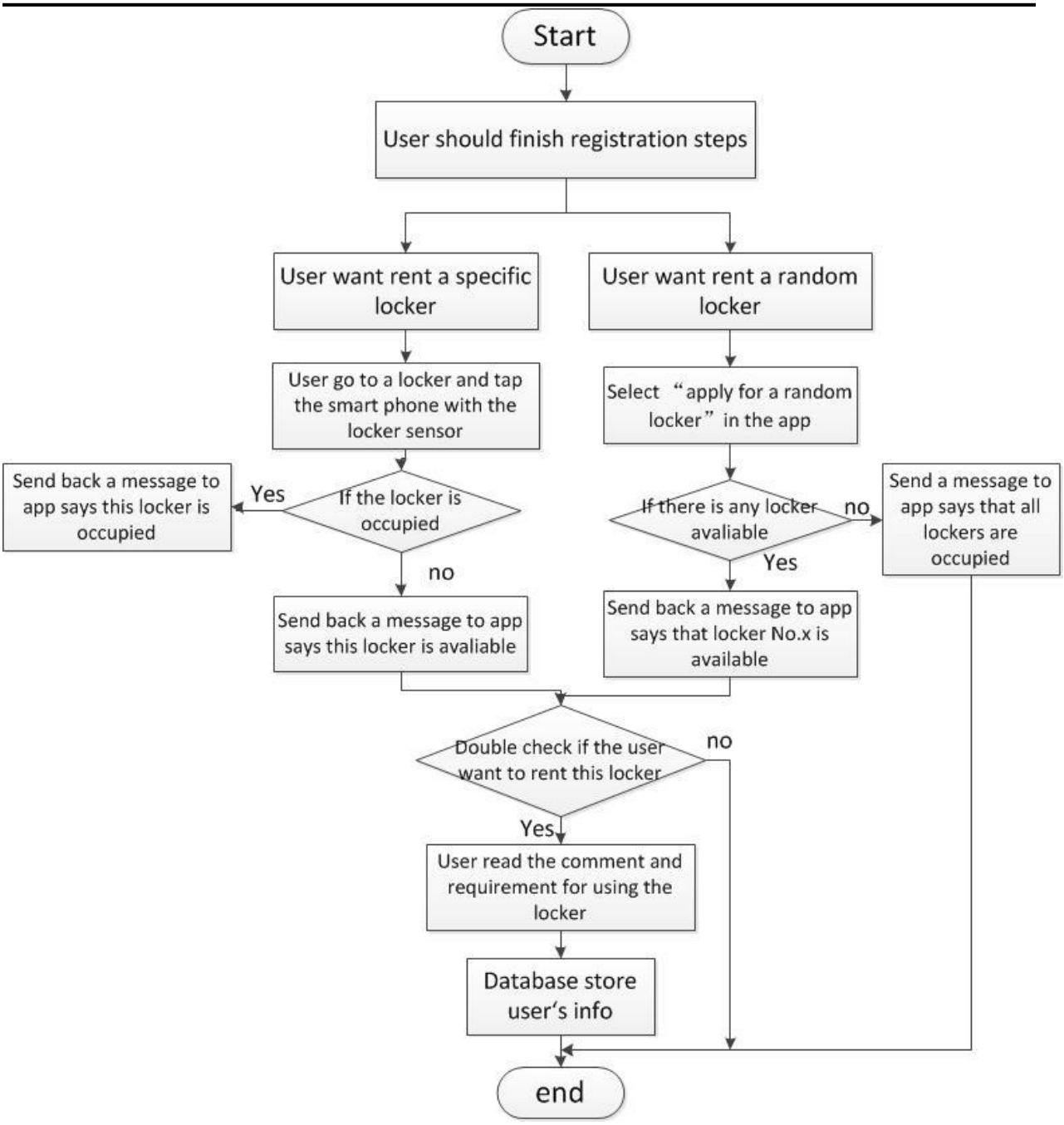

Figure 4: Flow chart of requesting a locker

Once the user finished renting a locker, the system will start assigning access. Figure 5 shows how to open a locker. Through the action of tapping, the NFC tag inside smartphone and locker will establish a near-field and start data exchange. When the application recognize the specific ID of the locker, the value is sent back to the database. If the user is confirmed as the owner of the locker, the server will sent a feedback, telling the application to send out a signal to open the locker.

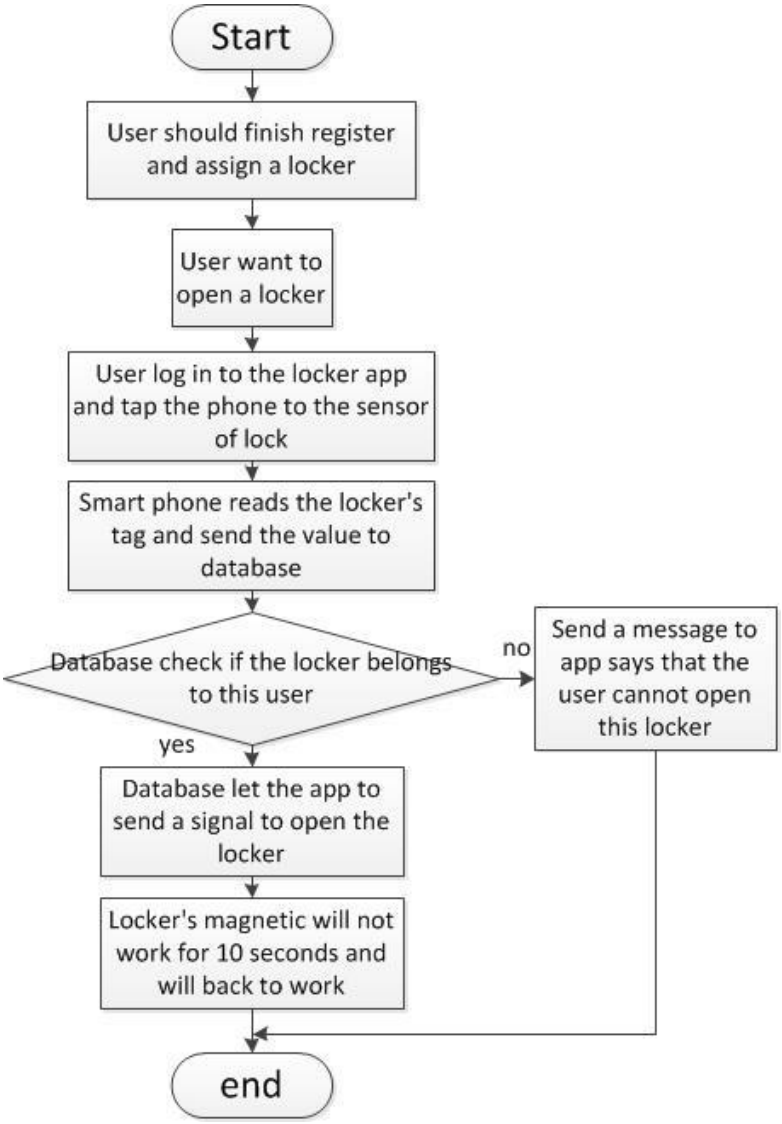

Figure 5: flowchart of opening a locker

5. Server

In the Smart Locker system, the server is the database. It saves all authentication information of each user and locker. Stored information is used to check the status of each locker, such as availability and keep in track with authorized permissions. Thus, once the user send out a request, the system can automatically check for availability, identify locker and assign access. In conclusion, the server communicates with the Android application to verify credentials, add new users, provide digital locker keys and etc. Figure 6 shows the components in sever.

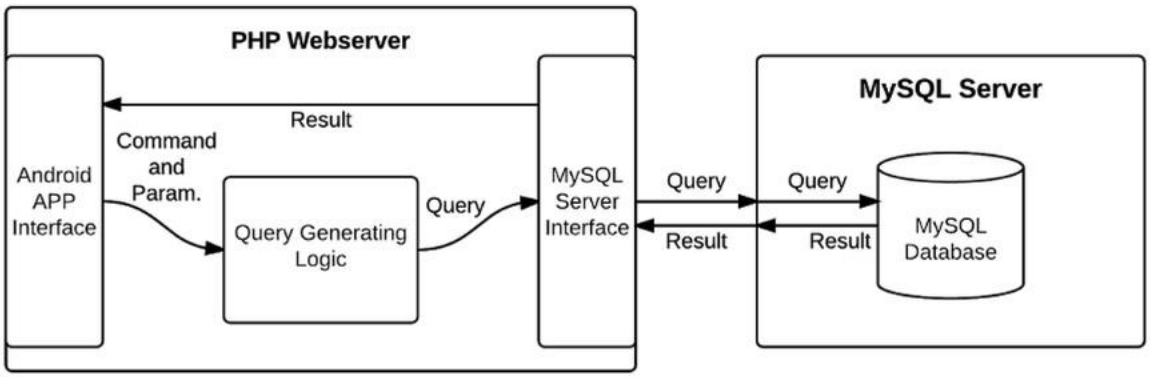

Figure 6: Server Component

5.1 Server/Database Specifications

5.1.1 Server Components

The server consists of two parts, a PHP based web server and a MySQL server, containing a MySQL database. The MySQL database have all the credential and locker rental information stored within and the PHP-based webserver is responsible to bridge the Android Application and the MySQL server. For example when a user logs in, the APP would send a request to the PHP server to verify the credentials provided, the PHP server would then send a query to the MySQL server to retrieve the credentials of the corresponding user name, the MySQL server would then search for the credentials in its credential table, and notify the PHP server whether the credential was found or not, and finally the PHP server would send a message to the Android Application to inform the success/failure of the login. The only component the Server interacts with is the Android Application. The PHP webserver and the MySQL server can be setup on one or two separate Windows or Linux powered computers.

5.1.2 PHP Components

The PHP server consists three parts, the Android Application Interface, the Query Generating Logic and the MySQL Server Interface. The Android Application Interface receives commands from the Android Application and return the outcome of the commands to the Android Application via JSONs. The Query Generating Logic takes the command from the Android Application Interface and generate the corresponding SQL queries. And the MySQL Server Interface pass the query from Query Generating Logic to the MySQL server, retrieve the SQL result sets and pass them to the Android Application Interface.

5.1.3 MySQL Server and Table Design

The MySQL server take query from the PHP server, run the query to retrieve data from the MySQL database and pass them to PHP server. The MySQL database inside the server should contain a total of three tables: User, Rental, and Locker. In the User table each row is consist of a username up to 10 characters, a password up to 30 characters and an email address up to 20

characters, with the username as the primary key of the table. In the locker table, each row is consist of a positive integer lockerID ranging from 0 to 65535 in hexdecimal, a 8 byte key in hexdecimal, 10 characters naming the group of locker this locker belongs to, and a two character string of the position of this locker inside the group, with lockerID as the primary key. The Rental table is consist of three columns, the username, the lockerID and the status. The username and lockerID are as described above and them together is the hybrid key of this table. The status is two characters indicating the ownership status. The status is PO when the user represented by the username is the Primary Owner of this locker, the status is a number when the user is given a limited number of access by the primary owner of this locker. Shown below are the mockup tables.

Table 1: User Table

Username	Password	Email
kegao	1a2_b3cs	kegao@ssu.com
jlee	241uaasd	jlee@ssu.com
iahua	sasdfdasg	iahua@ssu.com

Table 2 Lock Table

LockerID	Key	GroupName	Position
0001	0AE3BFCA	GymFstFlr	A1
00AF	F125EDA5	GymSecFlr	B3
FFFF	DA5FCA99	OffForFlr	D6

Table 3: Rental Table

Username	LockerID	Status
kegao	00AF	PO
jlee	0AC1	03
iahua	0C34	10

5.2 Server Design Requirements

A SSL encrypted connection should be established between the Android Application and the PHP webserver. When a user is logging in, the user-provided credentials will be send to the PHP server and the PHP server will query the MySQL server to count for the number of entries in the

database with the exact username and password, if no match is found, the MySQL server will return 0, if a match is found, the result will be 1 instead, either way the credential information stored in the MySQL server never leaves it, which eliminates the potential of credential leaking from the PHP server.

For future developments we may consider to use a Java-based webserver instead of the PHP-based webserver for better performance.

6. Electronic Locker

Table 4 is a list of hardware components that we used to build our electronic locker. Detailed design specifications are discussed in the following sections.

Table 4: Physical Components

Name:	Quantity	Usage
NTAG203 Type 2	1	Save the specific ID of each locker
NFC Shield	1	Read data from smart phones (ex. command to open/lock)
Arduino Uno R3	1	Read and analyze data from NFC Reader
LEDs	2	One green LED to show the accepting state and one red LED to show the denying state
Electromagnetic lock	1	Lock the locker

6.1 NFC Tag and NFC Reader Overview

We use NFC tags and NFC readers to build our electronic lockers. The NFC tag of each locker is pre-programmed by Arduino microcontroller and assigned a 2-byte positive integer ranging from 0 to 65535, which is used to uniquely identify all lockers. The locker identification numbers are stored in the Server with the corresponding locker's physical location. When user tapping their smartphones to the electronic locker, the built-in NFC chip of smart phones will be triggered. Then, our application starts to read the locker identification number from the tag and check the user information with database through wireless internet connections. If the information is matched with what saved in database, the server will send a 4-byte digital key to the application. Then, through Host-based Card Emulation provided by Android 4.4, the application makes the phone emulated NFC tag with a NDEF message containing the digital key as its content, the

NFC reader will read the digital key from smart phones and transfers data to Arduino. If the digital key matches the one stored in the Arduino. If the credentials are not matched, the server will send another message to smartphones to notice users their request has been denied.

6.2 Types of NFC Tag

There are 4 different types of NFC tags in the market: Type 1, Type 2, Type 3 and Type 4. Type 3 and Type 4 tags are read-only, data being entered by manufacturer or using a special tag writer. In order to easily reprogram our NFC tags for testing and reduce the costs, Type 3 and Type 4 tags are abandoned.

On the other hand, Type 1 and Type 2 tags are dual state, can be either read/write or read-only. Therefore both types are suitable for our project. In the Smart Locker system, NTAG 203 (Type 2) will be used since it is universally compatible and cost-effective with good memory capacity (144 Bytes) [3][4].

6.3 NFC Reader Design Specifications

The NFC Reader has two main parts, Arduino and NFC shield. It should also contain a switch which is made by a transistor and two LEDs, red and green. In terms of the NFC shield, it should read the message from smartphones and send the corresponding feedback to Arduino, accepting or denying. In terms of Arduino, it should analyze the feedback from NFC shield. If the feedback is accepting, Arduino will close the switch and turn on the green LED. Otherwise, Arduino will leave the switch at open state and turn on the red LED. NFC shield and Arduino will communicate through SPI (Serial Peripheral Interface). Figure 7 shows the NFC Reader that we are using in our project and where the NFC Reader located.

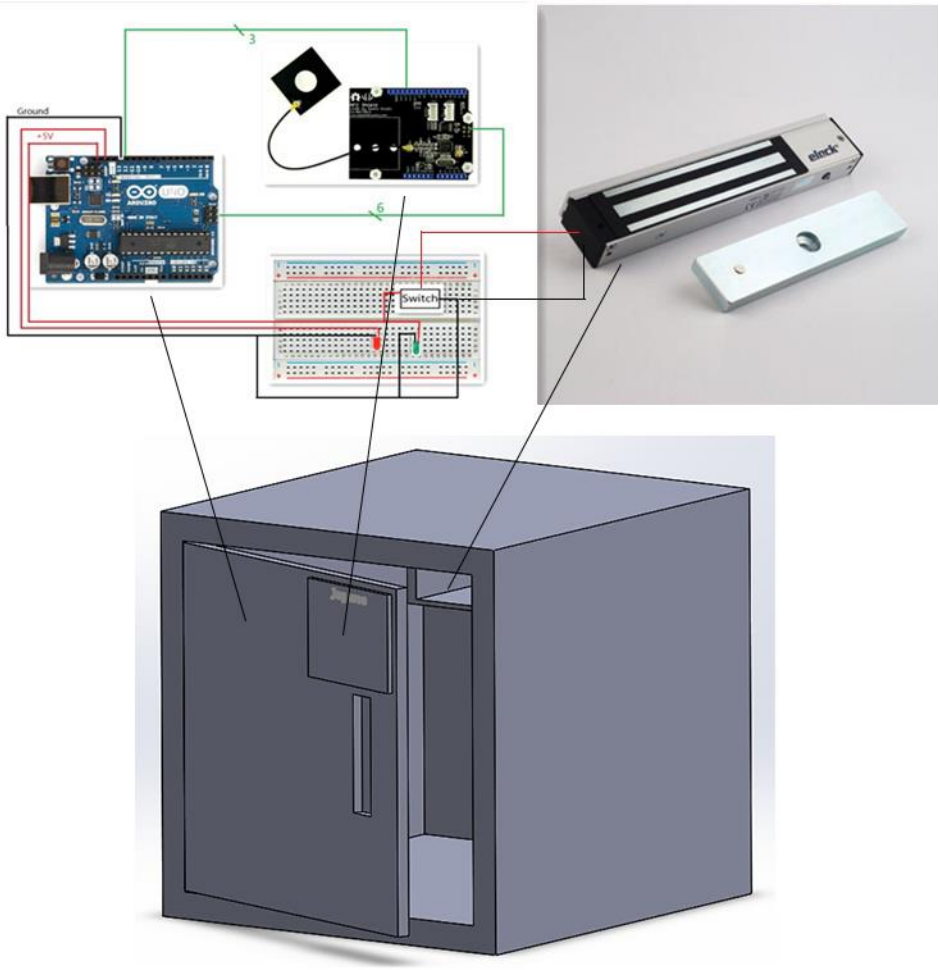

Figure 7: NFC Reader

For future development, one NFC shield should have multi-antennas which can receive the message from smartphone from different locker. Also, Arduino should be able to analyze all the messages from different antennas and send correct command to the locker simultaneously. Therefore, one Arduino and one NFC shield can control multi-lockers. It will significantly reduce the cost.

7. Arduino

For our project, we need a microcontroller to program the NFC tag in order to naming all the lockers. Also, the microcontroller reads and analyzes data from the NFC reader to decide whether to open the locker or not. For our system, we have initially considered two different options to programing the NFC tag: a FPGA board and Arduino.

7.1 Types of Microcontrollers

Arduino is an open source physical computing platform based on a simple input/output with a programmable integrated circuits (IC), ATmega328. In addition, Arduino has sensors such as

push buttons, touch pads and thermistors. Actuators such as LED lights and speakers are also available. Thus, Arduino is very useful for rapid prototyping [5], such as the Smart Locker. Comparing to FPGA, Arduino is much smaller but still powerful. FPGA contains different I/O ports and some IC modules such as audio/video codec, but these functions will not be used for our Smart Locker. In addition, Arduino has enough memory and lots of serial ports which is enough for our development. Moreover, Arduino is much cheaper than a FPGA board. In our project, we spent $60 on two Arduino microcontrollers. This might be the most important reason for choosing Arduino.

Although Arduino Uno is power enough for this project, all pins are occupied and it is really hard to add other functions. For future productions, we can use more powerful microcontroller boards such as Arduino Mega.

7.2 Arduino Design Requirements

Arduino is a very development-friendly hardware so it becomes a perfect tool for hobbyists and project developers. The Arduino board is very compact and it is not costly. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs for users and it will be friendly for add-ons. Specifically, Smart Locker is using an add-on called “NFC shield” which mounts on an Arduino Uno. Talking about programming, Arduino is using a specific language which called robotic-c, which is very similar to C and C++ language. It will be really easy for designers to program.

8. Android-based Application

Our Smart Locker system comes with an Android-based application. This interface establishes communication between lockers and database. Once the user tap the smartphone on the locker, the embedded system will start scanning the built-in NFC chip inside the locker to communicate with the server component. Once the request has been approved, the server will send a NDEF message to the locker component and give permission to open the closure.

8.1 Interface Overview

Multiple functions are designed to perform various tasks, such as sending closure requests and share access. The following are some basic activities that we need in our GUI design and more details and activities will be added during progressing.

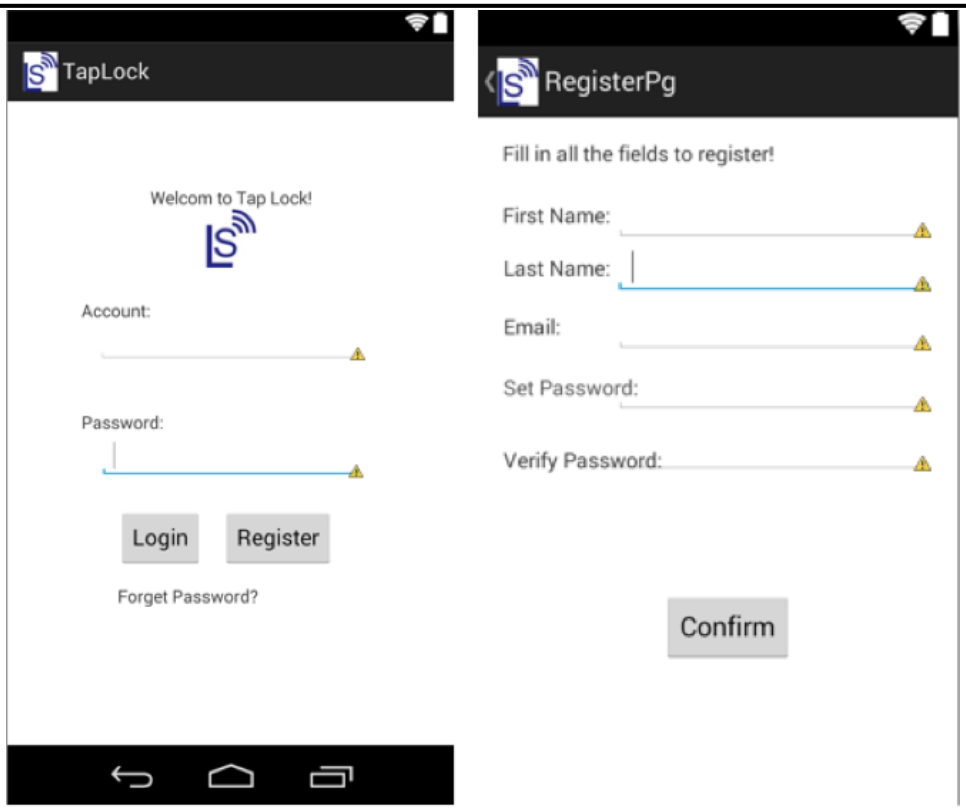

Figure 8: Login Page and Register Page

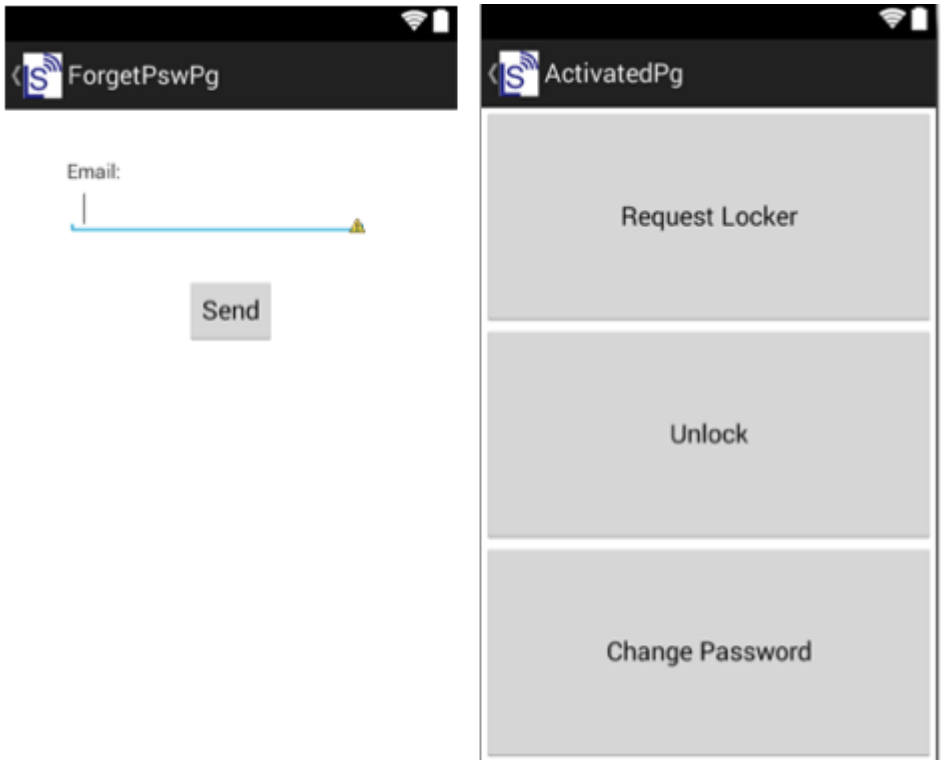

Figure 9: Forgot Password and Activity Page

Figure 8 Login Page (Left): this is the main page of our Android application. This page allows user to login into their account or register for a new account if they don't have one yet. Also, user can also retrieve their password by clicking 'forget password'.

Figure 8 Register Page (Right): User will be directed to this page if they click on 'Register' button on Login Page. This page allows user to fill in the required information to register for a new account.

Figure 9 Forget Password Page (Left): User will come to this page when they click on 'Forget Password?' text on Login Page. This page requires user's email to send out a reset password message.

Figure 9 Activity Selection Page (Right): Once user' information is verified on Login Page and is logged in, he/she will be directed to Activity Selection Page. This page contains several activity, such as Request Locker, Unlock Locker, and Change Password. User can choose to either one to proceed.

8.2 Design Requirements

8.2.1 Choice of Tools

We chose Android SDK to develop our Android application since it provides API libraries and developer tools necessary to build, test, and debug apps for Android.

Our GUI design is based on Android version 4.4 Kitkat Wear using Eclipse IDE. The Android project is separated into three main directories:

1) manifest file: describes the fundamental characteristics of the app and defines each of its components
2) main source files: includes activities that run when application is launched
3) app resources files: contains resources and definitions of GUI features, such as layout, drawable objects(i.e. bitmaps), and values(i.e. string and color definition)

XML and Java are the two software languages we have used in development. XML are used in things that are going to appear to user, such as layout, button, text field and color definition, while Java is mostly used when describing inner workings among different activities. After each design, we can build and run the application by setting up an emulator.

8.2.2 GUI Design

In the beginning of GUI design, all group members have met up to discuss the features needed by users. Then we outlined the activity pages and drew out a flowchart that describes our basic GUI structures. Our goal is to make a user-friendly GUI that is easy-to-use, self-explanatory, reliable and multi-functional.

8.2.3 Other Components

The Android Application communicates with both the Server and the Electronic Locker, the technical details of these communications can be found in previous section. The Android

Application consist of four major components beside the GUI, the Server Interface, the Tag Reader, the Host-based Card Emulator and Main Logic. The Server Interface issues commands to the Server, and receive results of the command executed. The Tag Reader reads the NDEF message containing the Locker ID from the NFC tag inside the locker. The Host-based Card Emulator emulates the Android device as a NFC tag containing a digital key to open the locker as a NDEF message. The Main Logic consist of the navigation of the GUI and the corresponding actions when the user interacts with the GUI.

8.2.4 Future development

In future development we hope to implement a ring-shaped GUI to make the Application more one-hand friendly. Also we hope to add locker maps to aid the user in locating free lockers to their preference of height and location. And finally we want to implement various time options for the locker rental.

9. Test Plan

The test plan will be applied to our prototype for troubleshooting and debugging. The tests will be done to make sure that the device meets both functional and design specifications. We divide the test plan into three specific groups based on our three main components of the Smart Locker system. They are Server Testing, Physical Component Testing and Software Application Testing. Finally, once all parts are combined and integrated together, we will do the sampling and destructive testing to ensure that our product can work properly. All below test plans are summarized in Table13.1 Test Plan Table in Appendix.

9.1 Server Testing

Testing PHP Server

9.1.1 Android Application Interface

- Input any username and a password pair, click login, the interface should receive the username-password pair with a command to check the credentials.
- Perform an account registration, locker request, password change request and unlock request, and verify the interface receives the corresponding data and command.

9.1.2 Query Generating Logic

- Input a non- existent command and invalid parameter, and the Logic should output an error “Command not found”.
- Input a valid command and invalid parameter, including wrong type, missing parameter, extra parameter and out-of-range parameter, the Logic should output an error “Invalid parameter”.
- Input every valid command with valid parameter in turn, the Logic should yield the corresponding SQL queries for each command.

9.1.3 MySQL Server Interface

- Input a query and the MySQL server should receive the query.

9.1.4 Testing MySQL Server

Create tables containing test data including credential table, user information table and locker information table.

- Query the MySQL server to find the number of matching username-password pair in the User table, should return 1 for existing username-password pair and 0 for non-existing ones.
- Query the MySQL server to create new row in the User table to store a new username that does not already exist with its password and email address, the new row should be created with the new information.
- Repeat the above test with an existing username, an error should be returned.
- Query the MySQL server to create new row in the Rental table to store a new user-locker rental that does not exist yet, the new row should be created with the new information.
- Repeat the above test with an existing username-lockerID pair, an error should be returned.
- Query the MySQL server to remove a row identified by an existing username-lockerID pair, the row should be gone from the table.
- Repeat the above test with a non-existing username-lockerID pair, an error should be returned.
- Query the MySQL server to retrieve a status on a row defined by an existing username-lockerID pair and the status should be returned.
- Repeat the above test with a non-existing username-lockerID pair, an error should be returned.
- Repeat the tests for the Rental table on the Locker table.

9.2 Physical Component Testing

There are two physical components for the locker: the physical locker and the electromagnetic lock. Therefore, testing will mainly focus on their physical performances.

9.2.1 Testing physical locker

- Applying 50N forces on both inside and outside of locker body to check if the locker is strong enough to protect the things store inside.
- Open locker's door without installing electromagnetic lock to see if it can close by itself properly. If the door is stuck in the middle that means there must be some unexpected fraction on the edge. Double check each side of the door to see if there is some pop-up spot on one edge, remove this by using tools.

9.2.2 Testing for electromagnetic lock

- When the electromagnetic lock is inactivated, try to open the locker by using 60 lb force. In this case, the locker door should not be opened. If the locker's door has been opened, check in details which part is falling. Correct the failing part and repeat this test.
- Set the default time limit for closing lock to 10 seconds. Test it by opening the lock immediately after sending an activation message. If the door can be open, check if the

connection between lock and Arduino is correct or not. Then check if these component is powered. Then check if there is any logical problem in the controlling code. The last thing is check if the lock is still able to use.

- Set the default time limit for closing lock to 10 seconds. Test it by opening the lock 10 seconds later after sending an activation message. If the locker can be open, double check this function by closing the door for another 5 seconds. Try opening it again this time. If the door can still be open, check the connection between lock and Arduino is correct or not. Then check if these component is powered. Then check if there is any logical problem in the controlling code. The last thing is check if the lock is still able to use.

9.3 NFC Reader Testing

In this test, both accepting and denying cases should be tested. Also NFC reader should be able to read message from different types of smartphones, such as Nexus 5 and Samsung Galaxy.

9.3.1 Accepting and Denying Cases Test

- Using any smartphones to send correct opening message to the NFC Reader and check if the switch is closed and the green LED is on.
- Using any smartphones to send incorrect opening message to the NFC Reader and check if the switch remains open and the red LED is on.
- Using any smartphones to send 20 random opening message to the NFC Reader and check if the NFC Reader's reaction is correct.

9.3.2 Different Smartphones Test

- Using Nexus 5, Samsung Galaxy to repeat the Cases test. The NFC Reader should work perfectly with different phones.

If any test fails, the testing engineer should check whether the hardware fails or the code fails. In order to do this, the test person should change the hardware, using the same code and repeat all test cases. If all test cases pass, it means that the previous hardware may be broken. Otherwise, the test person should check the code for failure.

9.4 Software Application Testing

Software application is intended to become an easy-to-use and self-explanatory tool that allows users to control features of locker without technical knowledge. Therefore, a detailed feature testing is needed to test this software application.

9.4.1 GUI testing:

- Each features such as buttons, text fields and text areas are functioning as intended and directing user to the correct activity.
- Each features should be able to adapt various unexpected actions by users as many as possible. For example, when users have typed in unintended symbols in text field, or when users have clicked the button more than once, the system would not crash, otherwise, an error message should display to notify user about the situation.

- Time to connect different activities should not exceed 5s, otherwise, a timeout error message is needed it to notify users that something has gone wrong.
- Application is stable so that it does not crash and quit unexpectedly.
- Designs such as colors, font, and styles should be consistent throughout the entire application.

9.4.2 Functionality testing:

- Software application should correctly establish connections between the Android Application and server.
- Clicking buttons (i.e. login and register) the Main Logic should retrieve user input, if any, from GUI and send them as parameter of a command to the PHP server via Server Interface.
- User information should be correctly verified (i.e. login) or stored (i.e. register) and be sent back from server to GUI to allow user to proceed the next activity.

9.4.3 Tag Reader Testing

- Tap a NFC Tag containing a text type NDEF message, the Tag reader should be able to output the text.

9.4.4 Host-based Card Emulator Testing

- Give a key to the Host-based Card Emulator, tap the Android device to the NFC reader, the reader should be able to receive a NDEF containing the key.

10. Conclusion

This document clearly states the design specification of the Smart Locker. The Smart Locker system should contains three main components: server, locker and smart phone application. Our locking system allows users to automatically request a locker and open the locker by tapping smart phones to the lock. Development of the product will take place in two phases: proof-of-concept and prototype. All possible design challenges are mentioned in this document. This design specifications can be used as a guide throughout the design process. Further improvements and expansions of functionalities will be considered. The expected completion date for our prototype is December 6th, 2014.

11. References

[1] A. Lella, “Smartphone Subscriber Market Share”, 2012. [Online]. Available: https://www.comscore.com/Insights/Market-Rankings/comScore-Reports-June-2014-US-Smartphone-Subscriber-Market-Share [Accessed 29 October, 2014]

[2] L. Francis, “Radio frequency identification: security and privacy Issues”, 2010. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-16822-2_4 [Accessed 3 November, 2014]

[3] I. Poole, "NFC Tags and Tag Types". [Online]. Available: http://www.radio-electronics.com/info/wireless/nfc/near-field-communications-tags-types.php [Accessed 4 November, 2014]

[4] NXP Semiconductors, [online]. Available: http://www.nxp.com/products/identification_and_security/smart_label_and_tag_ics/ntag/series/NTAG203.html [Accessed 4 November, 2014]

[5] A. Gibb, "New Media Art, Design and the Arduino Microcontroller: A Malleable Tool", February 2010 [online]. Available: http://aliciagibb.com/wp-content/uploads/2013/01/New-Media-Art-Design-and-the-Arduino-Microcontroller-2.pdf [Accessed 4 November, 2014]

[6] Erel, "Connect Android to MySQL Database Tutorial", 2011, March 21[online]. Available: http://www.basic4ppc.com/android/forum/threads/connect-android-to-mysql-database-tutorial.8339/ [Accessed 3 November, 2014]

[7] Android, "NFC Basics", [online].Available: https://developer.android.com/guide/topics/connectivity/nfc/nfc.html [Accessed 29 Sept, 2014]

[8] Android, "Host-based Card Emulation", [online]. Available:https://developer.android.com/guide/topics/connectivity/nfc/hce.html [Accessed 29 Sept, 2014]

12. Appendix

Table 5: Test Plan Table

Test Suite	Test Group	Test Case
9.1 PHP Server and MySQL Server Testing	9.1.1 Android Application Interface	9.1.1.1 Input any username and a password pair, click login, the interface should receive the username-password pair with a command to check the credentials.
		9.1.1.2 Perform an account registration, locker request, password change request and unlock request, and verify the interface receives the corresponding data and command.
	9.1.2 Query Generating Logic	9.1.2.1 Input a non- existent command and invalid parameter, and the Logic should output an error "Command not found".

		9.1.2.2 Input a valid command and invalid parameter, including wrong type, missing parameter, extra parameter and out-of-range parameter, the Logic should output an error “Invalid parameter”.
		9.1.2.3 Input every valid command with valid parameter in turn, the Logic should yield the corresponding SQL queries for each command.
	9.1.3 MySQL Server Interface	9.1.3.1 Input a query and the MySQL server should receive the query.
	9.1.4 MySQL Server Testing	9.1.4.1 Query the MySQL server to find the number of matching username-password pair in the User table, should return 1 for existing username-password pair and 0 for non-existing ones.
		9.1.4.2 Query the MySQL server to create new row in the User table to store a new username that does not already exist with its password and email address, the new row should be created with the new information.
		9.1.4.3 Repeat the above test with an existing username, an error should be returned.
		9.1.4.4 Query the MySQL server to create new row in the Rental table to store a new user-locker rental that does not exist yet, the new row should be created with the new information.
		9.1.4.5 Repeat the above test with an existing username-lockerID pair, an error should be returned.
		9.1.4.6 Query the MySQL server to remove a row identified by an existing username-lockerID pair, the row should be gone from the table.

		9.1.4.7 Repeat the above test with a non-existing username-lockerID pair, an error should be returned.
		9.1.4.8 Query the MySQL server to retrieve a status on a row defined by an existing username-lockerID pair and the status should be returned.
		9.1.4.9 Repeat the above test with a non-existing username-lockerID pair, an error should be returned.
		9.1.4.10 Query the MySQL server to retrieve a status on a row defined by an existing username-lockerID pair and the status should be returned.
		9.1.4.11 Repeat the above test with a non-existing username-lockerID pair, an error should be returned.
		9.1.4.12 Repeat the tests for the Rental table on the Locker table.
9.2 Physical Component Testing	9.2.1 Testing Physical Locker	9.2.1.1 Applying 50N forces on both inside and outside of locker body to check if the locker is strong enough to protect the things store inside.
		9.2.1.2 Open locker's door without installing electromagnetic lock to see if it can close by itself properly. If the door is stuck in the middle that means there must be some unexpected fraction on the edge. Double check each side of the door to see if there is some pop-up spot on one edge, remove this by using tools.
	9.2.2 Testing for Electromagnetic lock	9.2.2.1When the electromagnetic lock is inactivated, try to open the locker by using 60 lb force. In this case, the locker door should not be opened. If the locker's door has been opened, check in details which part is falling. Correct the failing part and repeat this test.

		9.2.2.2 Set the default time limit for closing lock to 10 seconds. Test it by opening the lock immediately after sending an activation message. If the door can be open, check if the connection between lock and Arduino is correct or not. Then check if these component is powered. Then check if there is any logical problem in the controlling code. The last thing is check if the lock is still able to use.
		9.2.2.3 Set the default time limit for closing lock to 10 seconds. Test it by opening the lock 10 seconds later after sending an activation message. If the locker can be open, double check this function by closing the door for another 5 seconds. Try opening it again this time. If the door can still be open, check the connection between lock and arduino is correct or not. Then check if these component is powered. Then check if there is any logical problem in the controlling code. The last thing is check if the lock is still able to use.
9.3 NFC Reader Testing	9.3.1 Accepting and denying cases test	9.3.1.1 Using any smartphones to send correct opening message to the NFC Reader and check if the switch is closed and the green LED is on. In Process
		9.3.1.2 Using any smartphones to send incorrect opening message to the NFC Reader and check if the switch remains open and the red LED is on. In Process
		9.3.1.3 Using any smartphones to send 20 random opening message to the NFC Reader and check if the NFC Reader's reaction is correct. Pending
	9.3.2 Different smartphones test	9.3.2.1 Using Nexus 5, Samsung Galaxy to repeat the Cases test. The NFC Reader should work perfectly with different phones. Pending
9.4 Software Application Testing	9.4.1 GUI Testing	9.4.1.1 Each features such as buttons, text fields and text areas are functioning as intended and directing user to the correct activity.

		9.4.1.2 Each features should be able to adapt various unexpected actions by users as many as possible. For example, when users have typed in unintended symbols in text field, or when users have clicked the button more than once, the system would not crash, otherwise, an error message should display to notify user about the situation.
		9.4.1.3 Time to connect different activities should not exceed 5s, otherwise, a timeout error message is needed it to notify users that something has gone wrong.
		9.4.1.4 Application is stable so that it does not crash and quit unexpectedly.
		9.4.1.5 Designs such as colors, font, and styles should be consistent throughout the entire application.
	9.4.2 Functionality Testing	9.4.2.1 Software application should correctly establish connections between GUI and server.
		9.4.2.2 Clicking button(i.e. login and register) should correctly send out user information typed in text field from GUI to server
		9.4.2.3 User information should be correctly verified(i.e. login) or stored(i.e. register) and be sent back from server to GUI to allow user to proceed the next activity.
	9.4.3 Tag Reader Testing	9.4.3.1 Component should be able to read the NDEF message in a NFC tag when the Andrino device is tapping the tag.
	9.4.4 Host-based Card Emulator	9.4.4.1 Component should be able to output incomming key in a NDEF message and the NFC reader have to be able to read the NDEF.