university of groningen

Honours project report

Application of NFC technology: comparing and contrasting NFC and QR technology for information retrieval

Author(s): Athul Raj Nambiar (s4126351) and Suhaib Basir (s3964973) ()
Supervisor(s): Dr. George Azzopardi

August 21, 2022

1 Introduction and motivation

Data transfer is at the heart of communication in the modern world. Every day, we are required to share and exchange some forms of digital data in our daily routine, especially using our mobile phones, and there are many tools that can allow us to accomplish these tasks. There are many methods third parties can intercept and steal sensitive information being transferred using traditional methods such as phishing techniques, spoofed area codes, computer viruses etc. This becomes a bigger problem when a user is transferring something sensitive such as bank details, money, or other sensitive documents. Therefore, choosing a medium of communication that ensures ones data is transferred effectively, safely, and efficiently is something that people must strongly consider before transferring any form of sensitive data.

NFC is very highly regarded in the field of computing and information science as its ease of use, ease of implementation, and enhanced security makes it a popular mode of sharing information with people around you. Currently, some of the more popular applications of NFC technology are credit/debit card payments via your phone, smart ticketing, and key-less access in hotels. NFC technology has a few security concerns as well, the main one being 'eavesdropping' which occurs when a third party intercepts a signal sent between two devices. The idea we will explore in this project is creating a platform that allows people to share custom information via NFC technology, particularly a platform that allows a teacher/professor to record attendance using NFC technology. We think that this could be a useful application of NFC technology as it allows people to share more customized data and could help people in their daily activities by sharing information quicker with other use cases as well; for example, filling out forms at the hospital, sending important files to colleagues, inventory management etc. To make this research more complete, we are also going to assess the efficiency, security, and overall compatibility of NFC technology compared to QR technology. QR technology, or quick response technology, is a type of matrix bar-code that holds information horizontally and vertically. This code can be scanned and easily transformed into human-readable information. Comparing the two forms of technology will help us assess if the current standard of using QR technology can be effectively replaced by using NFC.

The initial motivation behind this investigation was inspired by the Covid-19 pandemic, and the idea was to create a more effective methodology to help local restaurant and bar owners to track their customers in case they were notified of a Covid-19 outbreak within their establishment. However, since pandemic is no longer a prevalent issue within the Netherlands and restaurant and bar establishments no longer require such a method of tracking, we could still look into the notion of creating an NFC enabled platform which allows users to transfer custom data, which is the main purpose of this investigation. In order to support the main objective, we will also be conducting a user survey in order to determine if everyday users (specifically a target group of students) would prefer to use NFC or QR based methodologies for everyday information retrieval.

2 Methodology

In this section, we will discuss the outcomes of our research and the general steps we will take to design our applications.

2.1 Research Outcomes

The approach we took in this project was to initially create two different types of applications (NFC based and QR code based) which allows the user to transfer customized information to another device or some cloud database through the use of the smartphone. As outlined earlier, the use case we are employing is for With this in mind, we have two research outcomes that we would like to achieve:

1. Get a better understanding of the current scope of QR and NFC technology designed to be used by the average user
2. How to develop an NFC enabled platform that is able to transfer custom data
3. Gain insight into what mode of technology students prefer to use; either NFC or QR based.

The specific technology we wanted to compare was NFC technology and QR technology as a means of information retrieval and transfer. After conducting and implementing our research, we decided to create 3 applications, two of which are NFC based and the other one that is QR based.

2.2 NFC

During our investigation, we wanted to implement a smartphone application that allows for custom text to be transferred via NFC technology by only using a standard Android smartphone.

2.2.1 NFC Tags

The first option we will explore is to create a mobile app which allows the user to send customized data via NFC tags. The app should work as a reader and a writer; it should be able to input information into the tag, store it in a database, and if an activated tag is placed near the phone, it will read the information that is in the tag. Figure 1a represents the process that takes place when using our mobile application with an NFC tag. In the figure, the same mobile application will act as both the NFC tag writer and the NFC tag reader, just at different times. The same processes take place as shown illustrated in our Figure.

2.2.2 NFC Card Emulation

An alternative to NFC Tags is the process of making a smartphone into a custom emulated smart card. This process requires two mobile applications on two different smartphones, one phone with an application that will emulate a smart card and one phone with an application that will act as

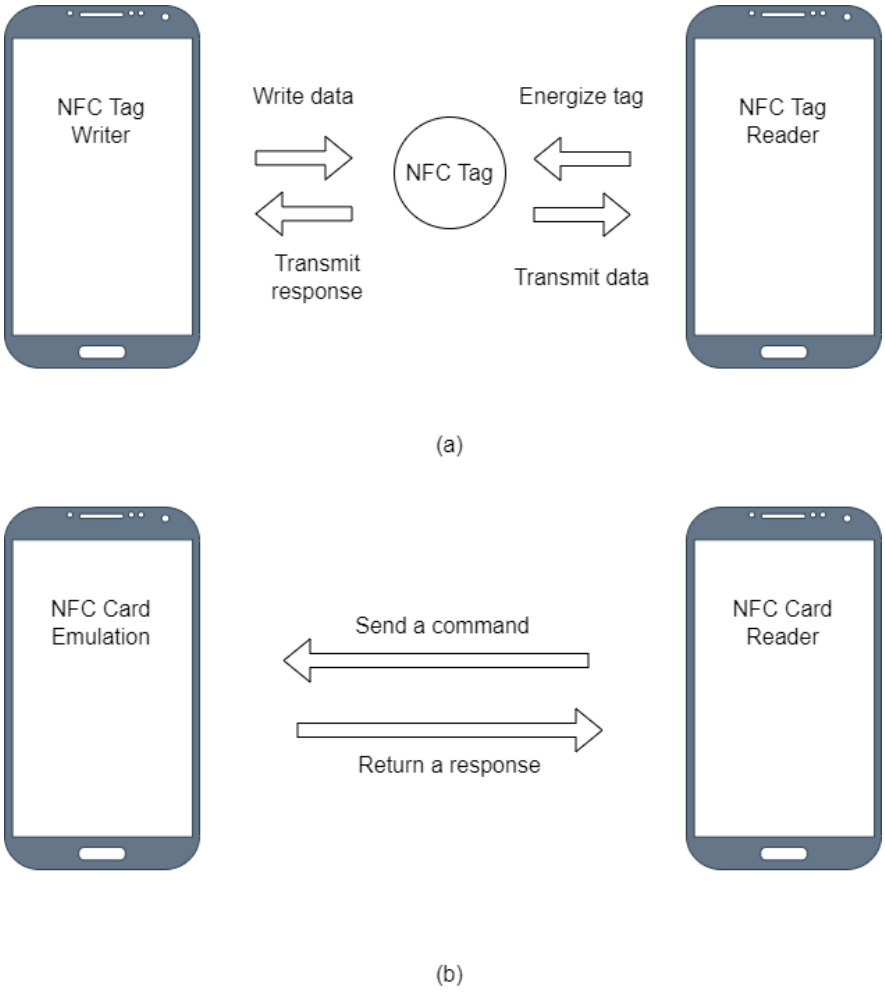


Figure 1: This Figure shows the pipeline process for both our NFC based Android applications. (a) represents the app pipeline using NFC tags. (b) represents the app pipeline using NFC card emulation.

an electronic card reader. The card emulator application will emulate a pre-programmed electronic card chip with custom information, in our example about the information of a student. The card reader application will be programmed to use NFC technology on the smartphone to detect an NFC chip, transmit a command and output the response from the NFC chip. This entire process is outlined in Figure 1b.

2.3 QR

In order to extend this research and conduct a more complete analysis, we are going to create a QR enabled web-app that accomplishes the same task as the NFC enabled apps. When a user scans the QR code it should open a form that takes the name and the student number of the student and then stores that information into a google sheets document.

3 Research of technologies

3.1 HTML

HTML stands for hypertext markup language and it is just the standard markup language for creating a website. We use it in combination with the other technologies but it is basically used to outline and create the divided structure of the webpage for the QR code enabled webapp. Usually the layout of a Web App is to create HTML file that contains HTML code. However with the user of tooklits such as Bootstrap that is not necessary. In our case we used Bootstrap components that contains HTML and CSS code and creates the structure of the overall Web App.

3.2 CSS

CSS stands for Cascade Style Sheets and the purpose of it is to make changes to the visual aspects of the Web App (colour, size, styles, etc.) by styling HTML documents. This can also be done with HTML alone, but with CSS it makes it easier for a user to make stylistic modifications too many elements of the Web App by assigning the elements to certain classes and making stylistic changes without too much redundant code in the HTML. Therefore CSS is a way to describe how HTML elements should be displayed when I load the Web App [1].

3.3 Bootstrap

Bootstrap is a framework used with HTML, CSS and JavaScript to create well designed and responsive front-end designs and can be really useful for many interface [1].

3.4 Java

Java is a high-level Object Oriented Programming language that allows programmers to write the code once and run it anywhere. Android studio offers developers to write their programs in both Java and Kotlin (discussed in the next section). The reason we chose this language is because we have ample experience coding in this language, having taken two courses in our degree program and by taking part it various personal projects using Java, we felt comfortable using this language. However, since we have never developed mobile applications before, there was still a lot of research and learning we had to conduct. We had to learn about new class types and objects which we could use help make our NFC enabled mobile applications [2]. Firstly, we had to learn about the concept of 'Intents' which is something we never learned. Intents are objects which can request actions from other application components and essential allow for communication between components. We also learned about the concepts of activities, services, and broadcasts. An Activity is essentially a single screen in an app, and intents are used to start new activities. A service is a component that conducts operations in the background that does not need a user interface; intents essentially describe what service needs to be carried out. And finally, a broadcast is a message that an app can receive [2]. Using these concepts we were able to build our app using Java.

3.5 Kotlin

Kotlin is an open-source statically typed programming language that also has type interface [3]. Kotlin targets the JVM, Android, JavaScript, and Native. Kotlin has both object-oriented and functional constructs, and it can therefore use both styles individually but also mix them. The goal of our research is to develop Android apps, because Android provides the functionality we need as discussed in section 3.6. Furthermore, at Google I/O 2019, they announced that Android development will be increasingly Kotlin-first [4]. The reason for this is that Kotlin is more expressive, concise, has language features to help avoid common programming mistakes producing safer code, 100% interoperable with the Java programming language, and has structured concurrency which simplifies background task management. In our research, we use both Java and Kotlin for our research; different applications are programmed with Java and Kotlin. We program in Java to practice and refresh our understanding of OOP concepts, and then program in Kotlin for its further advantages and gain experience. The advantage of Kotlin compared to Java is that Kotlin is more concise. Rough estimates indicate approximately a 40% cut in the number of lines of code [3]. It's also more type-safe, e.g. support for non-nullable types makes apps less prone to Null-Pointer Exceptions. Additionally, using Kotlin is beneficial because of its structured concurrency, allowing us to easily manage our background services for our card emulation program.

3.6 Android

Android OS is a mobile operating system that is based on the Linux kernel and other open source software that is designed for touch screen devices and therefore primarily runs on smartphones and tablets [5]. We chose to work specifically with Android devices for our research because the developer documentation for creating Android apps allows us to work with NFC and Host-Based Card Emulation (HCE). We chose not to attempt to create an IOS app because currently IOS does not allow for the possibility of developing apps with HCE which was a required to provide valuable insight into our research.

3.6.1 NFC tags

As outlined in our methodology, we plan to make two distinct apps using NFC technology. One which utilises NFC tags and another which emulates a card. For the first app, we had to buy NFC tags and then learn how to be able to program these tags within Android Studio. In order to program the tags we had to look into the NFC documentation that Android studio provided. We learned that there are three modes of operation for NFC capabilities within Android [6]:

- Reader/Writer mode
- P2P mode
- Card emulation

For the purposes of the NFC tag enabled Attendance application, this portion of the research only required us to look into the Reader/Writer mode of the NFC capabilities. This worked perfectly with the type of tags we bought - tags that offer write and read semantics, allowing us to write and then re-write information to a tag multiple times. The tags are shown below.

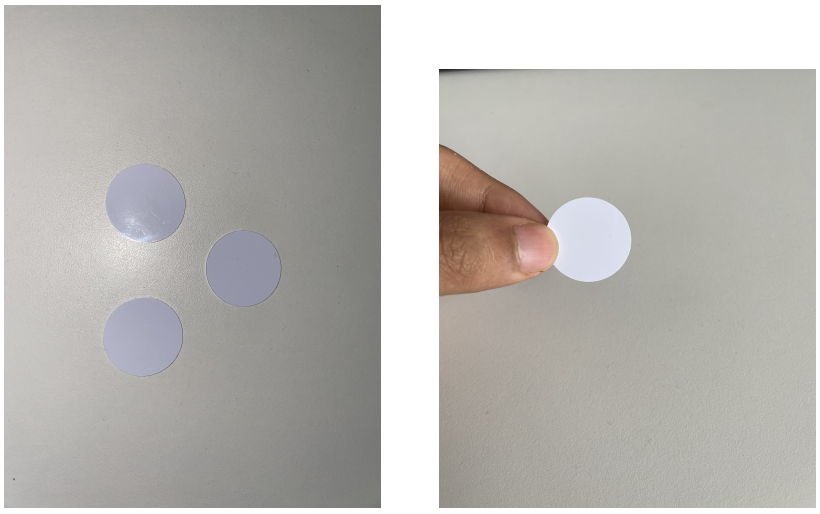


Figure 2: Images of the NFC tags

Using these NFC tags, we had to read and write data using the "NDEF" format [6]. NDEF, or NFC data exchange format, is a data format that is used to exchange information between NFC compatible devices and NFC cards/tags. According to the documentation, the data is stored in objects known as NDEFMessages which contain multiple NDEFRecords [7]. An NDEF record contains the data itself, it could be an URI or a "custom application payload" (customised data of a specific data type). The structure of the NDEFRecord contains extra information such as the data type of the data, fields to let the device's parser know how the data should be parsed, and the length of the data; these fields are the 3-bit TNF, length-type, and the length payload. Once the record has been properly parsed and the data has been obtained the system creates an Intent (explained in the Java subsection) that requests for further action; displaying the read information from the tag, or storing the information into a database.

3.6.2 Host-Based Card Emulation

For our research, we require Android devices that offer NFC functionality. The purpose of this research is to develop applications that can be used by the average user and nowadays, an average smartphone has NFC functionality and these devices also support NFC card emulation. In most cases, the card is emulated by a separate chip in the device, called a secure element [8]. Many SIM cards provided by wireless carriers also contain a secure element, however in modern devices developers would not have access to its functionality as they are used mainly by wireless carriers. However, Android 4.4 and higher provide an additional method of card emulation that doesn't involve a secure element, called host-based card emulation (HCE) [8]. HCE allows the possibility

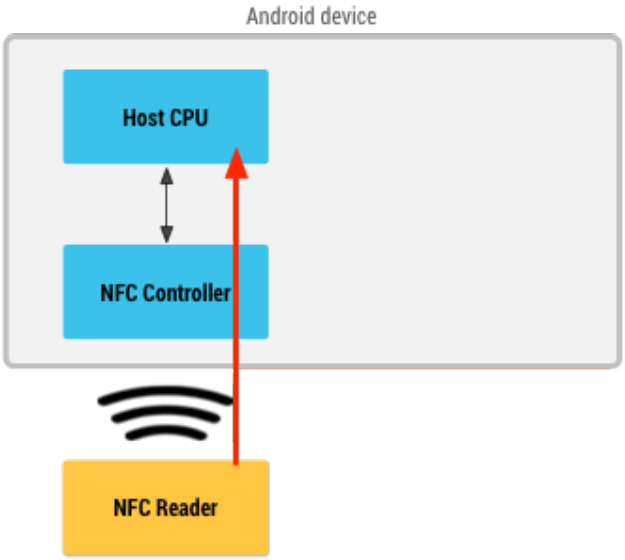


Figure 3: This Figure shows NFC card emulation without a secure element that occurs when we use HCE in an android device [8].

for an Android application to emulate a card and talk directly to the NFC reader. In Figure 3 we see that when an NFC card is emulated using HCE, the data is routed directly to the host CPU instead and there is no secure element involved in this process.

With HCE we need to consider the supported NFC cards and protocols. Android 4.4 and higher versions support several protocols that are common in today's market. Contactless payment cards and several existing contactless cards are based on these protocols. NFC readers in the current market also support these protocols, including Android NFC devices functioning as readers themselves. This allowed us to construct and deploy an end-to-end NFC solution using Android-powered devices based on HCE. We explain our implementation of this process in section 4.2.

4 NFC MOBILE APP

In this section, we will discuss the two NFC technology based apps that we developed. We first introduce the development process and finally the end result for both our NFC Tag based Apps, and our NFC Card emulator based Apps.

4.1 NFC DATA TRANSFER USING TAGS

The idea behind this application is that the student gets a tag/card and using a writer they are able to customize their NFC tag/card to hold their personal and unique information. The student can then use this one tag multiple times across all of their classes as they simply have to scan the tag/card once they enter the lecture hall. The reader which scans the tag reads the information that was written onto it and saves that information to an internal database, which can be viewed at the

click of a button. This is one method to ease the attendance taking process in a school/university setting. This concept can also be extended to various other settings such as checking in at the gym, tracking guest information at restaurants/bars, or to confirm identity for proof of ownership.

4.1.1 Development of the App

The main technologies used for creating and supporting this app are: Java to program the functionality, XML to design the layout of the app, SQLite database to store and query the information, and NFC tags to which the app writes and reads information.
We started by designing the layout of the app, using XML. At the time of development we only had access to one android device with NFC capabilities that is why we decided combine the writing and reading functionality within one app. Therefore, this app has two text fields which take in user input for the student name and the student number. In order to submit the data and write it to the tag, if a tag is present nearby, a 'write' button is present under the edit fields. Under the 'write' button there is a 'View' button which queries all of the entries inserted into the database. Finally, the last component of the app is a text view which prints the information stored in the database, if it was written to a tag.

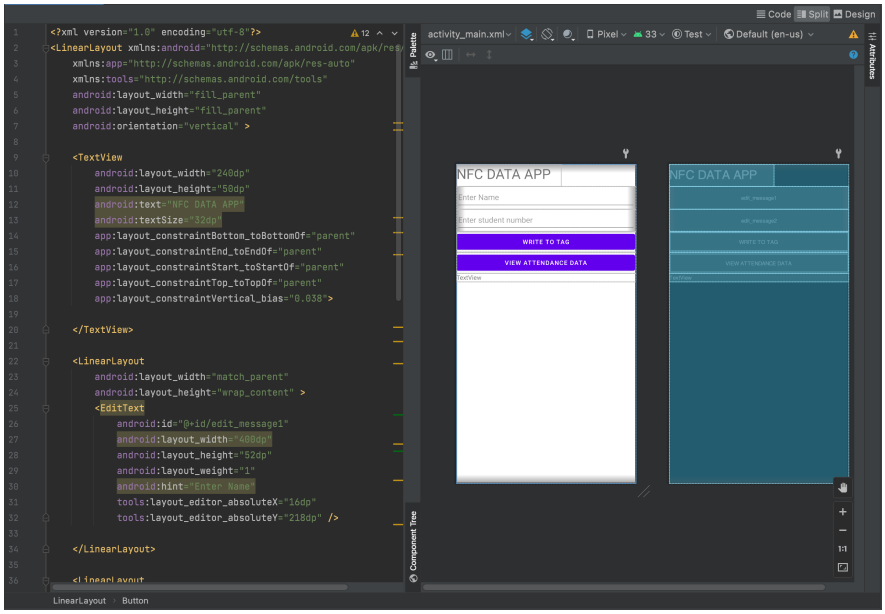


Figure 4: Designing the layout of the app

Once the layout was designed, we began programming the buttons and the NFC connection with the tags:
Firstly, we will explain how we write information into the tags. This process is done by taking the inputs from the two text views and combining it into one string. Then we send this string to the *writeToTag*() method. This method creates a new NdefRecord using the *createNDEFRecord*() method and then stores the new method inside an NdefMessage, as explained in section 3.4.1. The

createNDEFRecord() creates the NdefRecord by serialising the string into bytes and then saving it to a new NdefRecord object. This new object is saved inside an NdefMessage and then it establishes a connection to the nfc tag. We then use the in-built *writeNdefMessage*() function to write the information to the tag which saves the data as an NdefMessage object. It is important to note that the write button will only undergo the aforementioned process if there is a tag detected. If no tag is detected it will state "No nfc tag detected" and will do nothing.
Furthermore, we will outline how we read information from the tags. Writing to the tag involves serialising the information into bytes, therefore, reading from the tag involves de-serialising the data from bytes into a string. When a tag is placed near the device and the app can register the tag, the *readFromTag*() function is called. If a tag is near the device but the device does not have NFC capabilities an error message appears and the app closes. The *readFromTag*() creates a new NdefMessage object and it gets assigned to the one stored in the nfc tag. Once the NdefMessage object from the tag is obtained, it gets sent to the *convertToText*() function which takes an NdefMessage as input and returns a string by extracting the payload which is contained within the NdefRecord inside the NdefMessage object. The returned string is then saved to the text view which the user can see.
Finally, storing and viewing the information to and from the database is quite a simplistic process. When the app is launched an sqlite database is created with a schema that has two fields: name (TEXT) and snumber (INT). There are two supporting methods for the database functionality *insertData*() and *getData*(), which insert the student name and number into the database and query all of the stored data from the database, respectively. When data is written and read to and from the tag, the same data is stored in the database using the *insertData*() function. The view button then calls the *getData*() function and shows all the data stored in the database to the user.

4.1.2 Results: NFC Tag enabled attendance app

The following figures will illustrate the design and the functionality of the app.

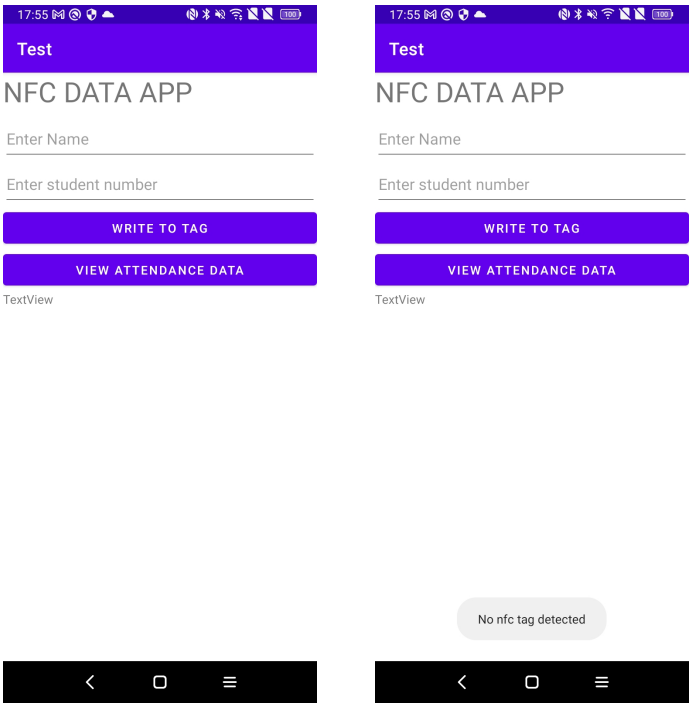


Figure 5: Layout of the app and error message when no tag is nearby

The figure 5 shows the basic layout of the app, with two input fields, two button and a text view. When no tag is near the device and the user tries to write information

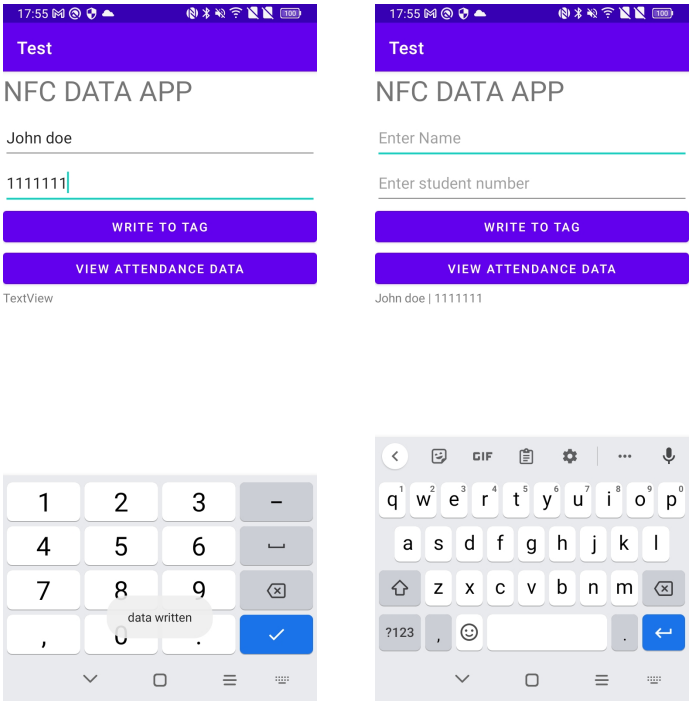


Figure 6: Writing and reading information to and from the nfc tag

The figure 6 on the left demonstrates that once the input fields are filled and the write button is pressed, the data is written to a tag, if present, and a success message pops up. The figure on the right 6 shows how once the app reads the nfc tag it processes the data and writes the same data back onto the screen.

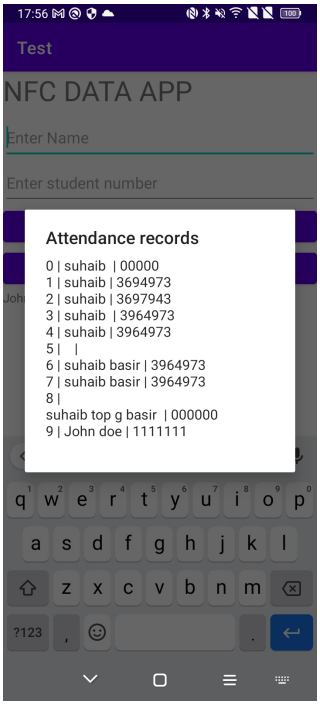


Figure 7: Output of the view button

The figure 7 shows what happens when the view button is pressed: it queries and outputs all of the information present in the sqlite database.

4.2 NFC data transfer using card emulation

The original aim of this application was for a student to enter their student information manually through the app, and then create a smart card that contains the student's detail. However, due to difficulties explained in section 4.2.1, we were not able to implement the manual input, however were able to create a contactless card that is preprogrammed with a student's information in order to register their attendance. We develop a secondary application that is able to read the emulated smart card and retrieve the preprogrammed student information on the chip of our emulated smart card. To avoid the confusion, HCE is different to NFC cards because in HCE the term "card" actually refers to a contactless smart card (typically with a protocol of ISO/IEC 14443-4) [8]. Whereas NFC cards are memory tags (although that some may have other additional processing capabilities) that store (freely readable) data in NDEF format [9].

4.2.1 Development of the App

The implementation of our HCE app required the development of two apps; one for the smart card emulator and one for the smart card reader. We did not use the same app as that developed in section 4.1.1 as we require we use a specific protocol to signal to our smart card.Furthermore, we required to android devices to test our apps and the NFC transmission process. The main technologies used for creating and supporting these apps are: Kotlin to program the functionality for the apps and XML to design the layout of the app, just the NFC tag based app.

Smart Card Emulator:
We designed the layout of the smart card emulator using XML. We initially attempted to create an emulator app that is able to input custom information into the chip after starting the app during run-time and modifying the chip. Therefore, we created a layout that is seen on Figure 9a. The app has two input text boxes and a "Submit Data" that were intended to allow user inputs to modify the data on the emulated chip during runtime. Then, by clicking "Transmit Data", the user modified data on the chip would be transmitted when a reader was held close to the emulation device. However, due to complication in the implementation and time restrictions, we designed an application that cannot modify the preprogrammed HCE card during the runtime. Instead, in the course of our research, we developed an app to load a customized emulated card that cannot be modified by the user in the app, and can only be customized by the app developer.

The HCE architecture on Android is based around the *Service* components on Android. A *Service* is an application component that can perform a longer-running operation while not interacting with the user. One of their key advantages is that it can run in the background without any user interface. In our particular case, we run the HCE service in the background at the initialization of the Android activity for our particular app. This is also the particular reason why our app cannot currently modify the data on the emulated card chip via user input in the app during runtime. We start our service when the app is initially activated, and therefore the chip can only contain the preprogrammed information. During the runtime, the specific HCE service class cannot be modified because Android does not allow for the *AndroidManifest.xml* file to be modified during runtime. This is the file that loads our service when opening our App. Due to lack of time, we could not finish the implementation of an alternative method so that the user can input custom data into the emulated card. However, we discuss our attempt in section 8.1.

In our current implementation, when a user taps our emulated card to an NFC reader, the Android system needs to know which HCE service the NFC reader wants to communicate with. Therefore, we need a communication protocol, which in smart cards are done through the ISO/IEC 7816-4.This specification defines a way to select applications, centered around an Application ID (AID). An AID can consist of up to 16 bytes. Based on the first 4 bits, AIDs are divided into different groups. AIDs starting with 'F' are proprietary AIDs (meaning they have no registration). Hence, for our apps,

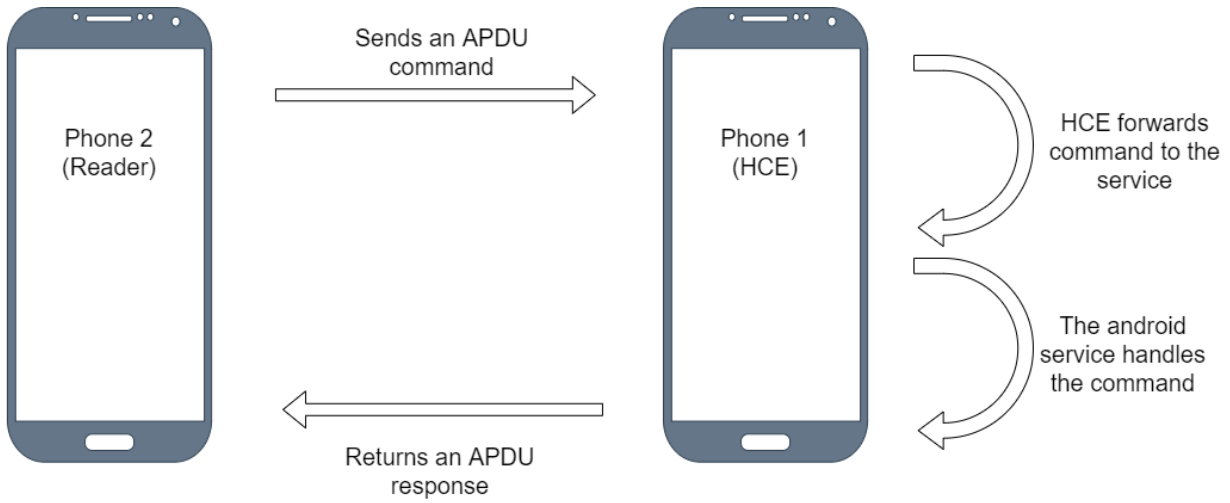


Figure 8: The data interaction process between the card emulator app and the card reader app.

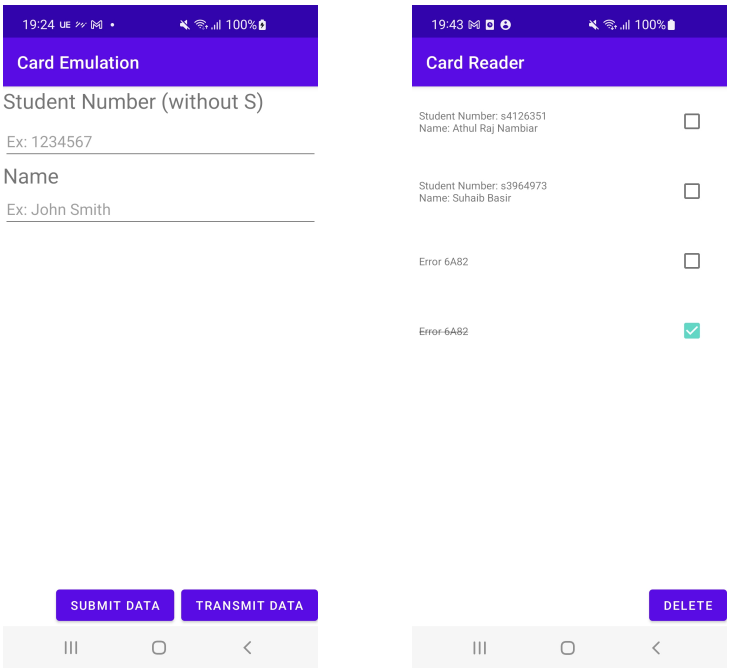
we use the AID "F0000001234567" which is arbitrary for research purposes and is an AID that is unregistered.

For our app to communicate with a card reader, a reader has to send an APDU command (Application Protocol Data Unit Command) to the card, which will respond with an APDU response[10]. Within the APDU command, the reader specifies the AID of the smart card it would like a response from. We run our HCE service in the background of our app by extending the *HostApduService* class and assigning it to our *Service* in the *AndroidManifest.xml* file at the launch of the app. Furthermore, we define the AID for the emulated card in the *apdu_utilities.xml* which will be checked once the HCE app receives and APDU command. When the device receives a command, we check the contents of the command and send the appropriate response using the overridden method *processCommandApdu()*. Figure 8 shows the entire process that occurs during the interaction between the smart card emulator (Phone 1) and smart card reader (Phone 2).

Smart Card Reader:

For our smart card reader, we designed the layout to be in a check-box format, as shown in Figure 9b. So that every time a user holds a card against the reader app, the APDU response from the HCE emulated card would be presented in a rectangular text view. There is a check-box next to the text view, with a delete button at the bottom of the screen. The delete button is implemented so that any selected box will have a line through all the text in the text view, and pressing the delete button will remove all the rectangular text views of the selected boxes.

Our reader has two main classes, the *MainActivity* class and *NFCResponseViewUtilities* class. We extend the *MainActivity* class with the *NfcAdapter.ReaderCallback* class. The purpose of this subclass is so that a callback is invoked when our Android activity finds a tag while the foreground activity is operating in reader mode. We specifically use the *onTagDiscovered()* method to send the



(a) (b)

Figure 9: In this graph, we see the end results of the two apps developed for the smart card emulation process on Android. (a) is the smart card emulator app, that contains information input text fields to store text information for a student; the input layout was not completed and therefore will be used for future work. (b) is the card reader app that contains a check-list of all the responses read when hovered over an NFC smart card.

APDU command, by calling the *Tag.transcieve()* that contains the AID we want a response from. *NFCResponseViewUtilities* class is to add the functionality of the check-box view and is structure using the layout from *info_blocks.xml*.

4.2.2 Results: HCE Attendance Card App and Reader App

Figure 9, we show the end product of both apps developed for the card emulation via NFC process. Figure 9a is the smart card emulator app. As discussed in section 4.2.1, the current layout with text inputs and buttons were developed to modify the data in the emulated card. However, since that functionality is not completed and left for user work, everything relevant for our emulated smart card with preprogrammed custom data occurs in the background in the software when this app opens. Therefore, once this screen open, our smart card with student attendance information is emulated. Figure 9b shows the smart card reader app that is able to communicate with our custom smart card AID. It is structured in the form of a check-box list. The first two items are when on Figure 9b are the responses, that we get when the smart card emulator app on a different Android device is tapped on the device with the reader app. Notice the two emulated cards that were tapped were preprogrammed with a different student's information. The last two "Error 6A82" responses shown are the responses that are displayed when the reader is near an ID card with an NFC chip.

This shows our reader displays errors for when incorrect AIDs are presented to the reader.

5 QR enabled web app

In this section we will outline how we made the QR-code enabled web-app for the student attendance use case. As outlined earlier, the main technologies we used to make the web-app and to get the QR code are: HTML, Google API, and the QR code generator from qr-code-generator.com. The web application itself is built on very basic functionality and accomplishes the simple task of registering attendance by taking in data from two input fields, one for student name and the other for student number. When the user clicks on the submit button, the information will be stored in a google sheets document, and a message will appear telling the user the information has been stored.

The QR code below opens the link to the website with the url: https://sbasir17.github.io/QR-CODE/

5.1 Development of the Web App

The main technology used to develop the Web App was HTML. The development process was quite simple as we only wanted the Web App to contain a form which takes two inputs, therefore, the HTML code itself is quite short and simple. We used the bootstrap front end toolkit for the design of the website as they have plenty of CSS templates to chose from in order to make the website look nicer. They also have HTML based design templates for forms which we used.

We used the "form" element within HTML which contains interactive elements to submit information. For example, we used two "input" elements one which took text based information for the name and another which took numerical information for the student number. We also added a button so the user could submit the information. In addition to the button, we had to add an even listener to the button which was connect to the Google Sheets API that would then write the information into the google sheets document.

The google API works by accessing the google sheet document when the HTML script is given a specific script URL. Once it accesses the document it searches for how many columns contain data and which row was the last data entry contained in. It then calculates which row and column the new data entries must be inserted.

Overall, the design and functionality is very simplistic and has been made to give an idea of how a typical QR code enabled online form looks like and how it stores information. The main purpose, as stated earlier, is to compare its usability to the NFC enabled mobile apps.

5.2 Result: QR enabled Web App

Figure 10: QR code which will lead you to a Web App

Once the User has scanned the QR code the following website will open and will contain a form for attendance.

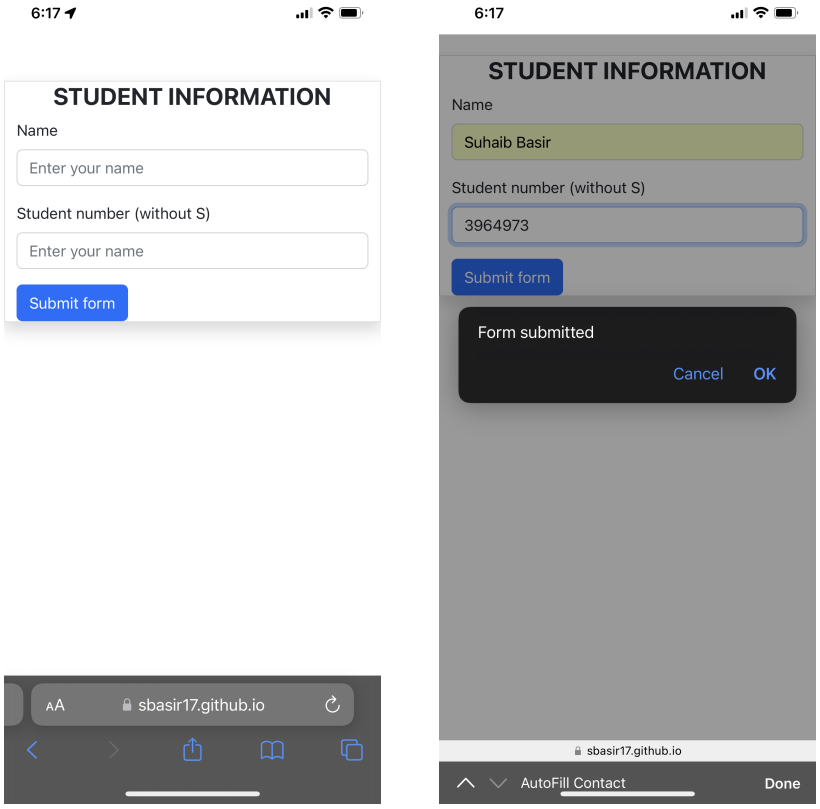


Figure 11: Web App which is opened by the QR code above

Once the form has been filled in and submitted the data is transferred and saved to a google sheets document in the following manner.

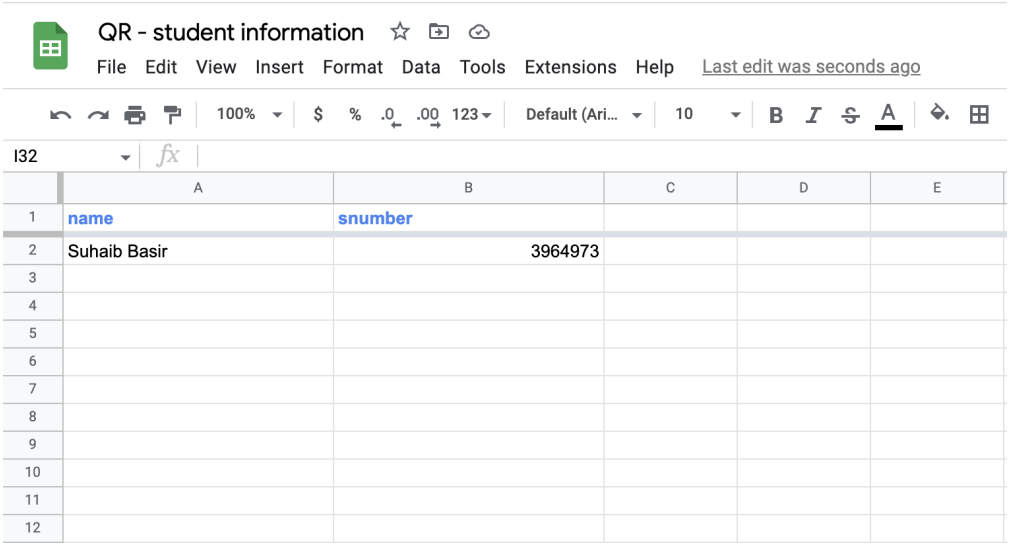


Figure 12: Google sheets document

6 Comparing usability section

In this section, we will be comparing the usability of all three apps to one another. We will list out their pros and cons from a usability perspective. We will also conduct a user survey in the next section and compare our analysis to the results we get from our survey in order to determine which mode of information transfer is most convenient for the attendance use case.

6.1 NFC tag app vs QR enabled web app

The NFC tag app and the QR enabled web app fulfill the same function as in similar manners, as they both require the user to fill a form and then submit their personal information to be stored in a database. However, the manner in which the data is transferred and is stored greatly differs among these platforms, one using NFC capabilities and the other which opens a web form via a QR code. The QR enabled web form is a more familiar platform and does offer more accessibility as most modern phones are capable of reading QR codes, while not everyone has NFC tags available to them and if this were to be implemented in an actual classroom setting, the NFC tags would have to be provided to everyone. However, in terms of its usability the NFC tag application is more convenient as the information just has to be written to the tag once, and then every time the student has to record their attendance, they scan the same tag in each of their classes. While the student would have to repeatedly fill the QR enabled web form in every class they attend, making it a more cumbersome task. The NFC technology offers more safety and reliability when conducting attendance as the QR code can be scanned by anyone and anyone can fill random information into the form. One tag will be available to each student with their data pre-written into the tag therefore there is no concern about the authenticity of the data. However, it is possible to give someone else the tag, which could be a security concern.

6.2 NFC card emulator app vs NFC tag app

Although these two apps are built using NFC capabilities they operate in different ways. Both technologies leverage the ease, efficiency, and security provided by NFC technology, therefore from a technical perspective there is little difference between the two platforms, as they only differ in usage. The card emulator app allows an NFC enabled device to communicate with another NFC device or reader without any intermediary, such as a tag or a card. This would be the more optimal solution, as most modern phones have NFC capabilities and can run the card emulation code. Making this solution more cost-effective and easier to implement within a real university setting, as there is no need to purchase additional NFC tags. Another benefit over the NFC tag application is that there is no concern regarding the authenticity of the data as each student has to register their attendance using their own devices and the information which is set in the card emulator cannot be altered once it has been initialized. Finally, there is also an extra layer of security using card emulation over NFC tags, because to access the information on a smart card, the reader needs the specific AID which is not the case for NFC Tags.

6.3 NFC card emulator app vs QR enabled web app

The NFC card emulator and QR enabled web app achieve the same goal in the context of tracking attendance, however they do have some notable differences. The NFC card emulator is preprogrammed with information that the card is supposed to transmit when the app is developed. This means that the user only needs to download the app once, with their information already stored in the emulated card. Meanwhile, for the QR enabled web app requires the user to input and submit their information each time they scan the code. In terms of long term efficiency, the NFC card emulator would likely be more appreciated by users. Furthermore, an initial concern was that the NFC card emulator would have an additional cost of purchasing a separate NFC reader in order to read our smart card, unlike the QR enabled website which was the cheaper alternative. But we managed to remove that cost by developing a reader app as well (although this assumes that we have access to a spare Android device). One difference is that the QR enable web app is able to change the custom student data each time, whereas the smart card emulator cannot be changed as it is preprogrammed. Therefore, to modify the information of their smart card, the user would have to get in contact with the developer.

7 User Survey and Results

As part of our research, we created a survey and sent it out to around 50 students and managed to receive 22 responses. Since we could not send the developed apps to all the students, as they would not have the resources to test them out, we created a survey that explained the scenarios and asked them to order them by their preference in the context of registering attendance in their lecture. The survey that we sent our contained three scenarios:

1. **QR enabled web form:** You scan a QR code with your smartphone, it opens up a web page, and you type in your student number and name to submit your attendance information.
2. **NFC tag app:** You download an app. Each time, you have to type in your name and student ID, then you submit your attendance information onto an NFC tag (small circular chip acting as a middle man). The NFC tag is then scanned by the lecturer with a NFC reader (Teacher's computer).
3. **NFC card emulator app:** You download an app and input your information once. Then you open the app and hover your phone over an NFC reader to share your attendance information.

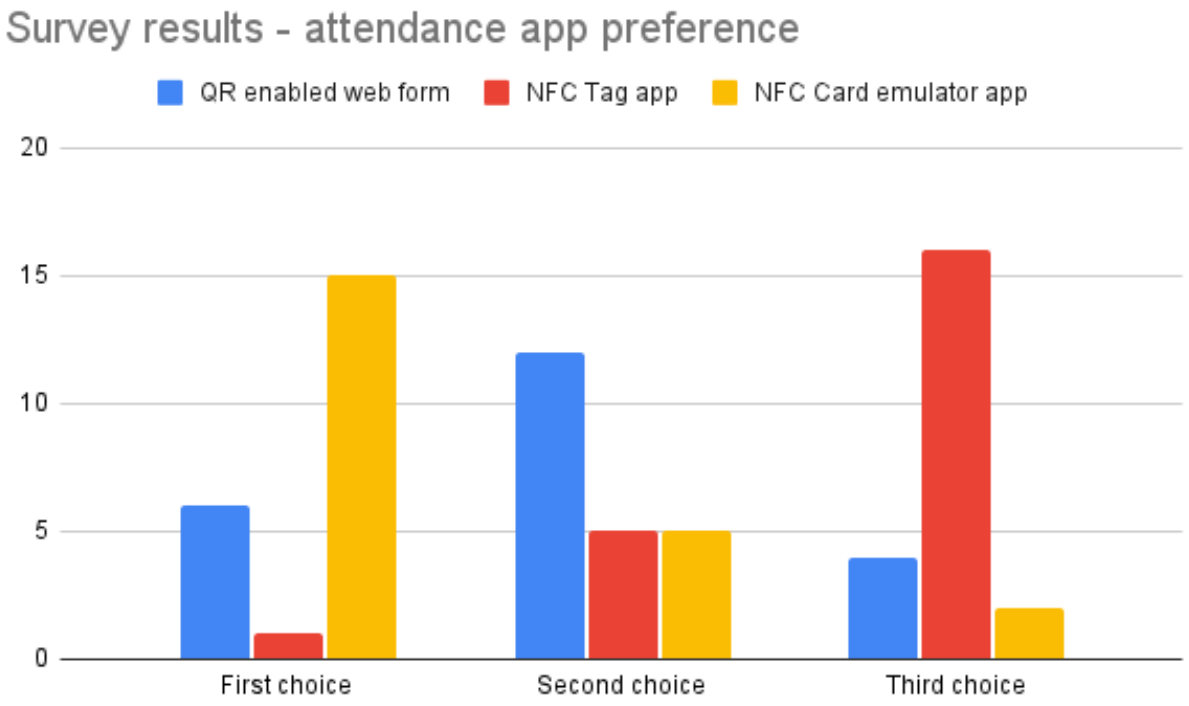


Figure 13: Results from the user survey

Figure 13 shows the results of our survey. We can see that in the context of registering attendance, the students preferred the use of the NFC card emulator app the most, the second choice was the QR enables web app and the majority last choice was the NFC tag app. After further inspections as to the reason, the students indicated that they prefer the NFC card emulator app the most because of its ease of use and only needing to submit their information and download the app once. While with the QR enabled web app and NFC tag app, they required to type in their information each time they want to register attendance. The reason the NFC tag app was last place seemed to be because the attendance process required an additional step of writing onto a NFC tag, which also increased the security risk or the possibility of tampering with the tag before it reaches the

system. Overall, the results from the survey were as we expected and gave a good indication of the preferences of the average user.

8 Conclusion

Through this investigation, we have attempted to ascertain various methods to make the process of attendance taking in a university or school setting easier. The primary focus of our research was driven by a motivation to learn more about NFC technology, and how we can utilize its ease of use, simple integration, and enhanced security and efficiency to create a new solution for this particular use case, with possibilities of extending our research in the future to other related use cases. Our investigation led us to create two distinct NFC enabled mobile applications which aim at easing the attendance taking process, and to further assess the effectiveness and appeal of these solutions we created a QR enabled web app which also records attendance via an online form. After creating these applications, comparing their usability, and conducting a user survey to ascertain which tool university students would use, we have determined that the most effective platform which can ease the attendance taking process is the NFC card emulator app.

Overall, the card emulator app leverages the benefits of NFC technology and has the capability of mitigating any of the associated risks and detriments. Since the majority of modern phones do have NFC capabilities, this solution would be easy to implement, as the only requirement would be to download the app and program the details of each student into the NFC card emulator within the app. This makes it easier to implement over the NFC tag application, as there is no need to purchase and provide tags for the students. Furthermore, by configuring each student's phone as an NFC card, it mitigates any concern regarding the authenticity of the data, however, with the tags it is possible for students to give someone else their tag to record attendance for them. Additionally, this solution is easier to use compared to the QR enabled web app, as the student only has to fill in their information once and then keep scanning the phone rather than filling the same form repeatedly; the QR solution also does not guarantee the authenticity of the data as the student can write whatever they want in the form. Furthermore, the URL retrieved from the QR code could be shared to students who are not attending the class. Conclusively, the NFC card emulator app performs well for our use case and can be extended to a variety of different scenarios with varying use cases.

8.1 Future Work

After researching NFC and QR technologies, we also consider possibilities for future research and improvements on our current implementations. In regard to our QR enabled web app implementation in the context of attendance, the biggest flaw is that once a person scans the QR code, they can forward the URL to other users that did not scan the QR code. To solve this issue in the future, we could design a web app that requires a user login. We would implement this using Fire-

base. Firebase was created by google and is a backend development software that has a lot of tools that can be used to create very useful Web Apps. For example, it has tools and services such as analytics, authentication, cloud messaging, a real time database, performance and others [11]. We can use firebase for its authentication and the database. With firebase, we can allow authentication functionality to the Web App by allowing the user to register an account and set a password. This allows the user to log in to an account and store the information on the real time database on firebase. It also allows the possibility to reset the user's password if necessary. As mentioned, the firebase database will be useful to keep specific information about each person's attendance. Once the user logs in, we can add a QR code reader SDK to the web app, so that once the QR code is scanned from the logged in account, the user's attendance will be registered. This implementation could solve the issue of being able to incorrectly register attendance.

With the NFC related app, the changes are less prevalent and are not major redesigns. With the NFC tag app, we only have particular design changes that we would like to make. We would like to add the functionality to edit and delete information from the tags. With the NFC card emulation app, we want to continue with the design changes that we were not able to implement as discussed in section 4.2.1. In the future, we would like to investigate the implementation of modifying and customizing the emulated card information during the runtime of the app, instead of only allowing for a preprogrammed information. We already attempted to investigate this issue, and it can be seen in our commented out code in our repository. How we plan to explore this in the future is to store the information input by the user in a separate file. Once that file is saved, we allow the user to start our subclass that extends the *HostApduService* class during runtime after the activity has started. We do this instead of at the initialization with the current version of our app. For this future implementation, we make sure our service opens the previously saved file with custom information saved by the user, and reads that information when creating the emulated card. This method is experimental and with further research could help solve our issue of creating custom NFC emulated cards during run time.

8.2 Individual Contributions

In this section, we will discuss the group and individual contributions. In terms of the workload, we divided our roles 50/50. We did this by splitting the workload each step of the way, had regular meetings and discussions, and set deadlines for ourselves. We are confident in saying that we managed to contribute equally towards this research project and the report. From now on, we will refer to ourselves in the third person for clarity.

Athul's primary focus was on the design and implementation of the NFC card emulation app and the creation of the survey to obtain external results. Suhaib's primary focus was the design and implementation of the NFC tag app and the QR enabled website. Athul and Suhaib decided that this split was fair as the during the research and development process, as the implementation of HCE

was more challenging and time-consuming than expected. Both Athul and Suhaib were involved in each other's individual tasks during the development process. Specifically discussing implementation challenges, coming up with suggestions and alternative solutions, and looking through each other's code to understand each other's work. Furthermore, Athul and Suhaib made a conscious effort to understand each other's implementation because one of their goals of this project was to improve their current skills and gain new programming skills. For the report, Athul and Suhaib wrote about their respective research focuses and for all the other sections, which made up the majority of the report, they wrote together.

References

[1] D. A. Yu, *The complete 2022 web development bootcamp*. [Online]. Available: https://www.udemy.com/course/the-complete-web-development-bootcamp/ (visited on 08/20/2022).

[2] *Intents and intent filters*. [Online]. Available: https://developer.android.com/guide/components/intents-filters (visited on 08/20/2022).

[3] *Faq: Kotlin*. [Online]. Available: https://kotlinlang.org/docs/faq.html (visited on 08/18/2022).

[4] *Android's kotlin-first approach*. [Online]. Available: https://developer.android.com/kotlin/first#:~:text=Kotlin%5C%20is%5C%20an%5C%20expressive%5C%20and,best%5C%2Din%5C%2Dclass%5C%20features. (visited on 08/18/2022).

[5] E. Mixon, *What is android os?* Apr. 2020. [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/Android-OS (visited on 08/18/2022).

[6] *Nfc basics*. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/nfc#java (visited on 08/20/2022).

[7] *Near field communication overview*. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc (visited on 08/19/2022).

[8] *Host-based card emulation overview*. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/hce (visited on 08/19/2022).

[9] *Nfc - ndef formatting*, Oct. 2019. [Online]. Available: https://help.gototags.com/article/nfc-ndef-formatting/ (visited on 08/20/2022).

[10] M. Hamdaoui, *How to build a simple smart card emulator amp; reader for android*, Jan. 2018. [Online]. Available: https://medium.com/the-almanac/how-to-build-a-simple-smart-card-emulator-reader-for-android-7975fae4040f (visited on 08/20/2022).

[11] L. Rosencrance, *What is google firebase?* Apr. 2019. [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase#:~:text=Google%5C%20Firebase%5C%20is%5C%20a%5C%20Google,creating%5C%20marketing%5C%20and%5C%20product%5C%20experiment (visited on 08/21/2022).