Department of Computer Science and Engineering
The Chinese University of Hong Kong

Mobile Application Using NFC

LYU1301 Final Year Project Report

Wu Han Ning Henry 1155017408
Chan Sin Yee 1155006665

Supervised by Michael R. Lyu

NFC gives opportunities to connect people between virtual world and physical world. In this project, we do research on NFC and made use of this powerful technology to build an attractive Android app. With the idea of social network, our app, Cucom, is finally implemented.

Contents

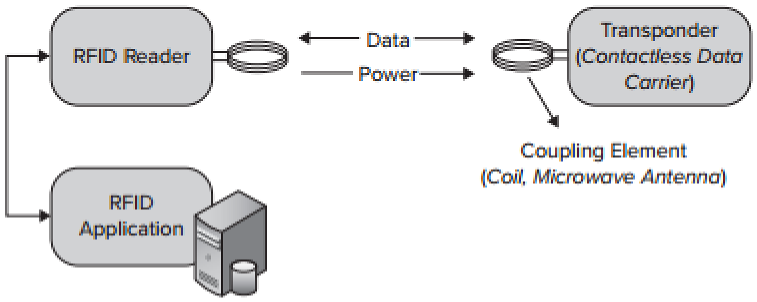
1 Introduction

1.1 Background

Near Field Communication (NFC) is one of the enabler for ubiquitous computing. This is a new technology begins to deploy on new smartphones. Users can use NFC to pass data wirelessly through devices or between device and a tag by touching them together. Since the design of NFC consists of security, the NFC enabled mobile devices are good replacement of credit cards, tickets, car keys and hotel room access card. It also simplifies the interactions of users with intelligent environment. The technology that NFC used is based on Radio Frequency Identification (RFID), but focus on consumer smartphone.

1.1.1 Radio Frequency Identification

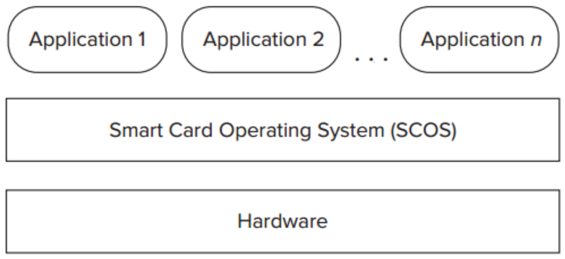
Radio Frequency Identification (RFID) is a wireless communication technology for identification and tracking. The data transmission results from electromagnetic waves, which can have different ranges (20 – 400 cm)



The RFID application is assigned specific information after transmission then the application can do some operation. A typical example is the Octopus system in MTR station. Passenger uses his Octopus card to validate e-money using RFID, the RFID application then receives the data inside the Octopus card. If the card contains enough money, then unlock the gate otherwise keep locking the gate. The RFID technology is being used all over the world for different applications. Including inventory system, Human implants, Animal Identification, Casino Chip-tracking, etc. The main defect of RFID is that the tag is so passive such that people who own a tag cannot read the data inside directly. Presumably, normal people do not own an RFID reader. For example, passengers cannot check their e-money inside their Octopus card at home; they have to go to the MTR station to check it using specific machine or device.

1.1.2 Smart Card Technology

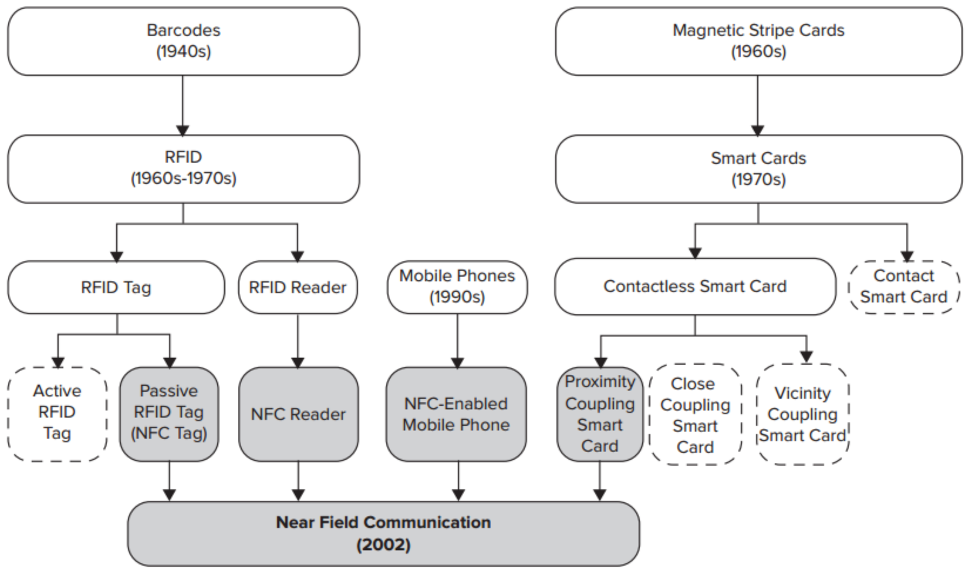
Smart card is similar to a tag but it contains an embedded microprocessor to enhance the secure ability. In order to generalize the usage of smart card, the microprocessor may have installed a Smart Card Operating System (SCOS). Upon the operating system, applications can then be installed.



However, the communication of smart card relies on two methods, physical contact through the conductive contact plate on the surface of the smart card and magnetic strip. Examples are EPS, HKID Card and CULink. Although it has high security, it requires user to insert or slide the card into a machine or device which may not be so convenient. Besides, likewise, user cannot read the data inside the tag unless suitable public machine is nearby.

1.1.3 Near Field Communication

NFC integrates the technology of RFID and Smart Card putting it into a mobile phone. It standardized for consumer smartphone. Since mobile phone has a monitor, user can read the data from a tag through the phone in anywhere. With an NFC-enabled mobile phone, user can even turn the phone to an active tag containing personal information, credit cards and set of keys. Moreover, because of the mobility of NFC device, devices can communicate using peer to peer mode, it, on the other hand, enhance the interactions of mobile phone usage.



1.2 Motivation

Living in this era, we cannot imagine what our lives will become if mobile phone disappears. Originally, mobile phone was used for voice communication between two people apart from each other. Yet, mobile phone evolves so quickly such that it becomes a mobile computer today. Now, people not only use phones to make phone call, they use their phone for searching, gaming, socializing and messaging. Since the mobile phones attach to them everywhere, the power of the phone becomes the power of them as long as the battery is charged. This special power makes them easy to work on anything; they can plan a party by just several clicks. The bonds between people should become increasingly stronger, but is this happening in reality?

We observe that many people get addicted on their phones. In a study of 1,600 managers and professionals, Professor Perlow found that 58% of smartphone users don't go 1 hour without checking their phones [1]. People stuck in the world of smartphone, communicating through the screen. In our experience, people use their phone during transportation, dining or even walking. They become so immersed in the virtual world that they end up not paying attention to the real world, people and conversation around them. This could harm the relationship between people.

To create a connection between the virtual world and the physical world we live in. Near Field Communication (NFC) might be the technology that makes this vision comes true. In today's smartphones, manufacturers are starting to immerse the NFC technology into new phones, e.g. BlackBerry Bold 9790(2011) Nexus 7(2013). Increasingly many people own a smartphone with an NFC service, which leaves a question: What can developers make use of this treasure? People are required to go to a specific location in order to use NFC; this is the key to connect people to the real world.

We believe NFC can saves people from the immersive virtual world by creating a unique culture. We believe that if we can create a popular app that encourages people to use NFC, people will pay more attention to the physical world.

1.3 Project Objective

The goal of our project is to write an attractive Android application using NFC service. We have set the following objectives for our app to achieve the goal:

The app should

- Be Interactive with the physical world using NFC. The beauty of NFC is the connection between virtual world and the reality; we have to focus on the function that NFC brought to the app.
- Be Interesting. The idea has to be interesting which makes user to share with their friends, and finally addicted to it.
- Contains user-generated content (UGC). UGC helps the app to grow by users themselves, a fruitful content attracts new users and keep the old users alive.
- Has well and fluent user-experience. User may feel inconvenient if it has bad user-experience, which may stop user to continue using the app.

2 STUDIES ON NFC

NFC is a set of standard for smartphones based on RFID technology created by Philip and Sony. Standards include Near Field Communication Interface and Protocol (NFCIP-1/2), Near Field Communication Wired Interface (NFC-WI), Front-End Configuration Command for NFC-WI, NFC Data Exchange Format (NDEF), etc. NFC enabled devices can establish data transmission by touching 2 NFC enabled devices together or by touching a tag to the device over a few centimeters.

Since NFC is inherited by RFID, the operating frequency is the subset of that of RFID which makes NFC-enabled mobile device supports some of the RFID tag. Besides, NFC is invented partially for replacing credit cards and access keys, security is a major issue. Although RFID supports operating range up to 4 meters, NFC restricts to only 0-4 centimeters which greatly enhance the security.

2.1 Comparison with Bluetooth

In the spectrum of wireless communication, we categorize them according to their operating range. There are wireless wide area network (WWAN), wireless local area network (WLAN) and wireless personal area network (WPAN). For the WPAN technologies, Bluetooth is one of them and is also available in mobile phones. Since NFC and Bluetooth are both short-range communication technologies and integrated into mobile phones. We can give a simple comparison between them.

Aspect	NFC	Bluetooth
RFID compatible	ISO 18000-3	active
Standardization body	ISO/IEC	Bluetooth SIG
Network Standard	ISO 13157 etc.	IEEE 802.15.1
Network Type	Point-to-point	WPAN
Cryptography	not with RFID	Available
Range	< 0.2 m	~100 m (class 1)
Frequency	13.56 MHz	2.4–2.5 GHz
Bit rate	424 kbit/s	2.1 Mbit/s
Set-up time	< 0.1 s	< 6 s
Power consumption	< 15mA (read)	varies with class

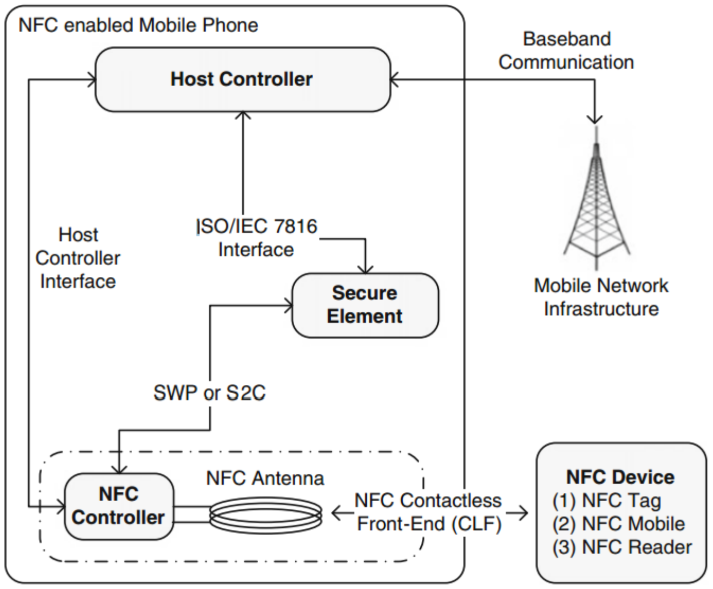
The operating range of NFC is so short, which reduces the likelihood of unwanted interception during any transmission. Yet, NFC sets up in less than 0.1 seconds, the short handshaking time allows devices the begin transmission just after the touch.

Compare with Bluetooth, first requires searching for targets, after selection then handshaking with authentication. In practice, it takes up more than 10 seconds in Set-up time.

NFC requires comparatively low power; hence it works with an unpowered NFC tag. In contrast to Bluetooth, more power is needed to illuminate through a long range of distance. Operating with passive device is difficult.

2.2 NFC Mobile Architecture

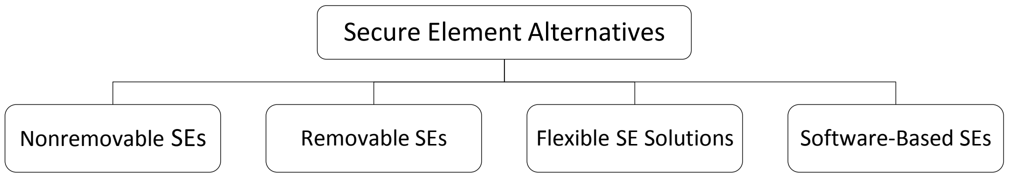
NFC-enabled mobile devices are typically composed of various circuits such as host controller, secure element(s) (SE) and NFC contactless front-end (NFC CLF). The NFC CLF includes an NFC controller and an NFC Antenna. NFC controller is the driver to convert host controller interface (HCI) commands to analog signal for NFC antenna. Host controller is the main controller of the mobile phone, which is connected to the NFC controller using HCI. The host controller sets the operating modes of the NFC controller then processes the data that is sent and received. On the other hand, SE provides a secure storage environment for private data such as credit card information. It is possible for a mobile phone to contain more than one secure element.



2.2.1 SE

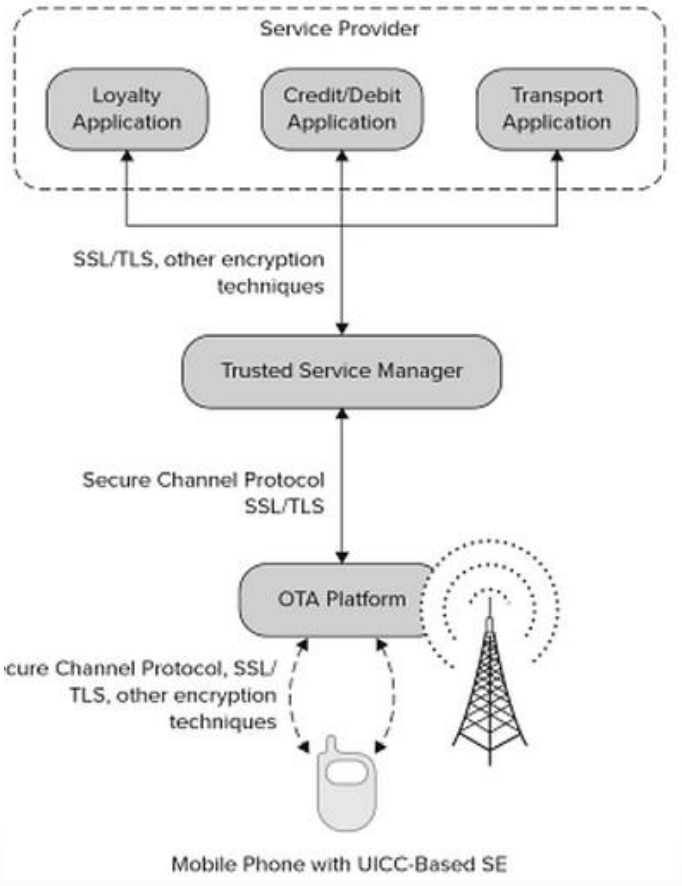
NFC enabled devices must ensure customer the credit card information and transaction takes place in a secure environment. The protection method is very similar to that of smart card technology. An SE, itself, is a separate microcontroller that has its own operating system, such as MULTOS and JavaCard OS. If such sensitive data is needed to transmit E.g. Payment, the secure element will directly access the NFC controller.

There are various SE alternatives available in the market. We can divide them into 4 parts: Nonremovable SEs, Removable SEs, Flexible SE Solutions and Software-Based SEs. Nonremovable SE is an embedded hardware which gives a relatively high security. The chip embedded in the mobile phone is inserted during the manufacturing phase; hence it has the same security level as smart card. Removable SE such as Secure Memory Card (SMC) and Universal Integrated Circuit Card (UICC), these allow user to change after purchasing the mobile phone, thus manufacturers only need to produce one generic phone for public. Furthermore, because the NFC enabled mobile phones are not very common todays. Thus, for SIM-based SEs and SMC-based SEs, SE manufacturers need to produce a generic SIM card that can support both NFC-enabled mobile phone and common mobile phone. Therefore, we have a new class of SE, Flexible SE Solutions.



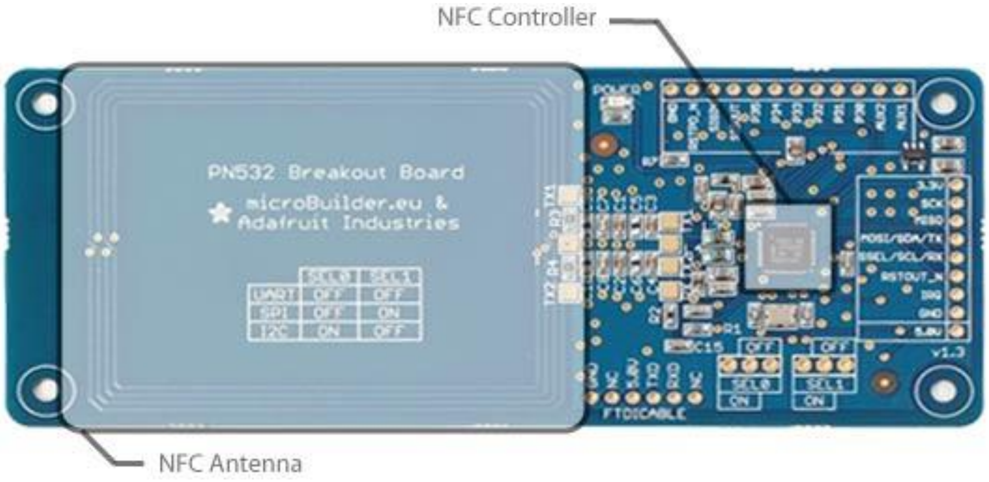
SE Management

Average user does not have the ability to alter the SE, but management is still needed. The authority is given to the Over-The-Air technology (OTA). Upon approval of user, They can install and load new NFC applications on SEs, remotely through UICCs. Besides, they can activate / deactivate SEs, remote service managing, PIN reset of NFC applications on SEs.


Service Provider
Loyalty Application
Credit/Debit Application
Transport Application
SSL/TLS, other encryption techniques
Trusted Service Manager
Secure Channel Protocol SSL/TLS
OTA Platform
cure Channel Protocol, SSL/ TLS, other encryption techniques
Mobile Phone with UICC-Based SE

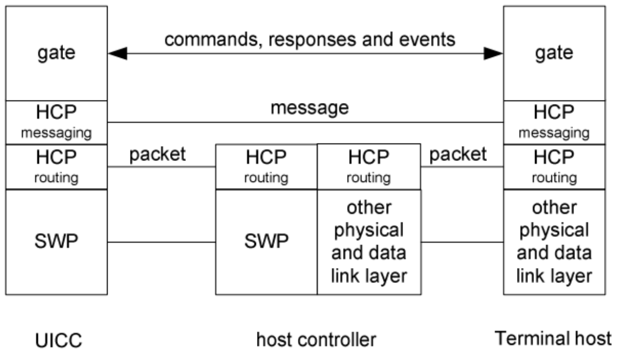
2.2.2 NFC Interface

The NFC Interface is composed of an NFC controller, an Antenna and NFC Contactless Front-End (NFC CLF). The NFC interface is able to communicate with host controller, SEs and other NFC interface through RF signal. It is the core component of NFC. The NFC controller works as a modulator and demodulator between the analog RF signal and NFC antenna. Besides, the NFC controller receives operations the host controller via HCI. The NFC CLF is the analog front end of the NFC controller. It is the logical interafce defines the protocol on top of the data link layer.



HCI

HCI is a logical interface allowing NFC interface to communicate directly with the host controller. Host controller sends command and data as a package to the NFC controller under Host Controller Protocol (HCP). Examples of HCP are SPI, I2C and HSU (High Speed UART) serial links.



At the beginning, the Host controller has to initialize the NFC controller by setting up configurations. After configuration, the NFC controller will wait for RF signal from other NFC devices. If RF signal is detected, the host controller is allowed to receive ID of the NFC tag / NFC device. Then host controller can use its ID to communicate with the NFC tag / NFC device using its protocol. For different type of target, the NFC controller will use different communication interface standards.

Parameters	ISO/IEC 14443	ISO/IEC 15693	ISO/IEC 18092
Operating Mode	Reader to card	Reader to card	Peer-to-Peer
Communication Mode	Passive	Passive	Active and Passive
Range	Proximity	Vicinity	Proximity
Data Rate	106 Kbps	Up to 26 Kbps	106, 212, 424 Kbps

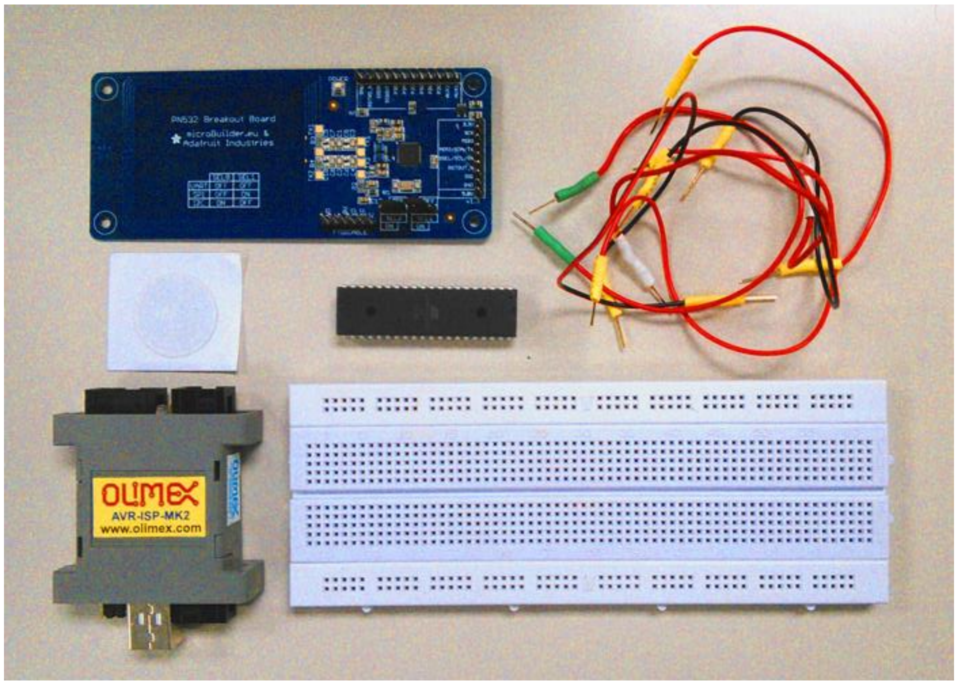
Serial Peripheral Interface Bus (SPI)

SPI is the transport layer between microcontrollers. Devices communicate in Master/Slave mode where the master initiates the data frame. A common clock is required to be established between master and slave. The Host controller, in this case, act as the master and the NFC controller act as a slave. All the commands in HCP will be converted into packets then be transmitted to the NFC controller.

2.2.3 Experiment

In order to investigate the operations of NFC in details, we have made a simple prototype to read and write NFC tags. In the prototype, it contains a simple host controller, NFC controller and an NFC antenna. The host controller tells the NFC controller to write an URL into an NFC tag. Then we will test it using an ordinary NFC-enabled mobile phone.

Components



- Breadboard
- Connectors
- ATMEGA324A microcontroller (Host Controller)
- PN532 NFC/RFID controller breakout board (NFC Interface)
- Mifare Ultralight (NFC Tag)

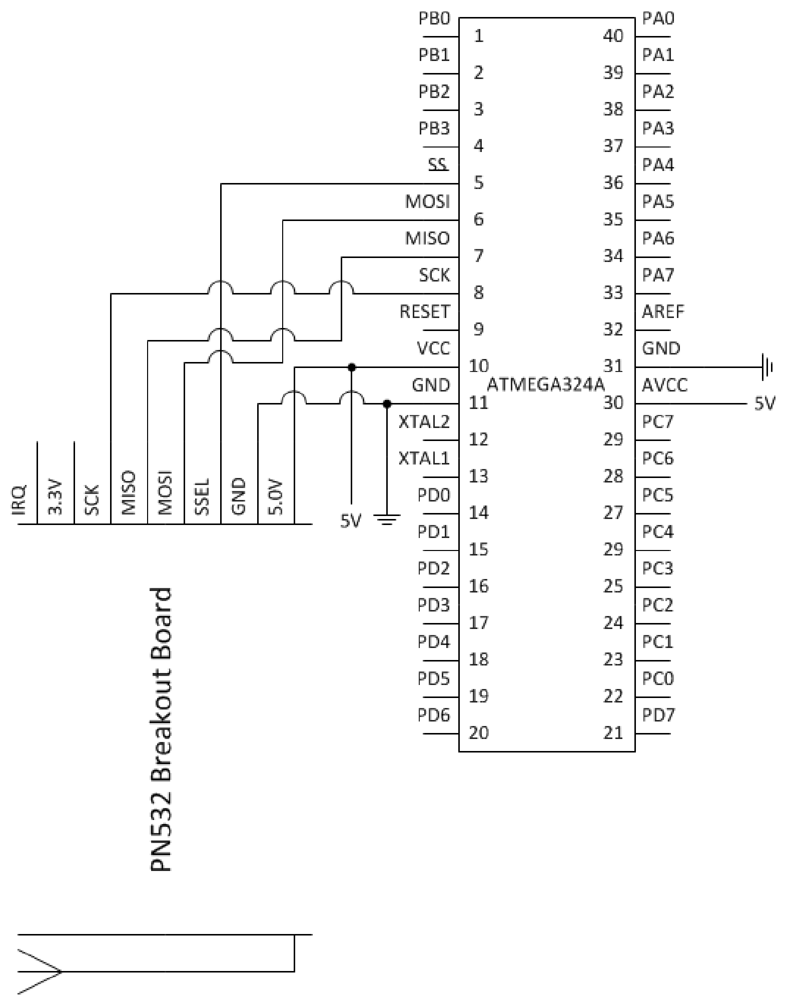
Equipment

- Computer
- Olimex AVR-ISP-MK2 Programmer
- Atmel Studio

Design

After the host controller initializes the NFC controller, the host-controller will keep monitor the incoming RF signal. If an anonymous NFC tag is moved in within a range, the host controller will write a hardcoded data into the tag. We have chosen to use SPI bus as the HCI, since the microcontroller use SPI as the default IO interface.

We connected the circuit as follows:

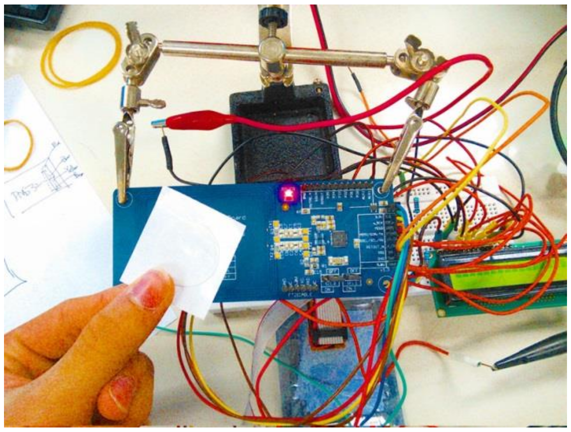


The controller writes an URL (http://www.cuhk.edu.hk) into a Mifare Ultralight Tag using ISO14443 standard whenever an NFC Tag is detected. This code is written as follows:

```
uint8_t data[20]={  0x03, 0x10, 0xd1, 0x01, 0x0c, 'U', 0x01,
                    'c', 'u', 'h', 'k', '.', 'e', 'd', 'u', '.', 'h', 'k', 0, 0};

// Initialize NFC Device
setupPins();
begin();
SAMConfig();

// Main loop
while(1)
{
	while(!readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uid_length));

	// When a tag is found
	for(i = 0; i < 5; i++)
	{
		// Authenticate block
		if(authenticateBlock(1, uid, uid_length, 4 + i, KEY_A, key))

		// Write data into the block
		writeMemoryBlock(1, 4 + i, data + i*4);
	}
}
```

Testing

We put a tag in front of the NFC antenna for a while, then put back to an NFC enabled mobile device. We can successfully initiates the CUHK website.



2.3 NFC Operating Mode

There are three classes of NFC devices: NFC enabled mobile device, NFC tag and NFC Reader. By pairing up these devices, we have different interaction styles. Each NFC operation has an initiator as well as a target. Initiator is an active device that initiates the communication with the target. Target, on the other hand, can be either active or passive. Hence, an NFC tag cannot be an initiator. In summary, the NFC operating modes are the reader/writer, peer-to-peer and card emulation modes with different interaction styles.

Operating Mode	Initiator	Target
Reader / Writer	NFC Mobile	NFC Tag
Peer-to-Peer	NFC Mobile	NFC Mobile
Card emulation	NFC Reader	NFC Mobile

Reader/Writer Mode enables NFC mobile device to exchange data with an NFC tag. Peer-to-Peer mode enables two NFC mobile devices to exchange data with each other. Card emulation mode allows Mobile device use its SEs to emulate a smart card such as credit card and ID card.

2.3.1 Reader/Writer Operating Mode

In reader/writer mode, the mobile device initiates the NFC tag by providing power through EM energy. Upon authentication, the mobile device is free to choose reader mode or writer mode. In reader mode, the mobile device can request a read operation on specific block of data in the tag. In writer mode, the mobile device overwrites any data on the block, modification is not supported. The only possible data rate in this mode is 106 Kbps.

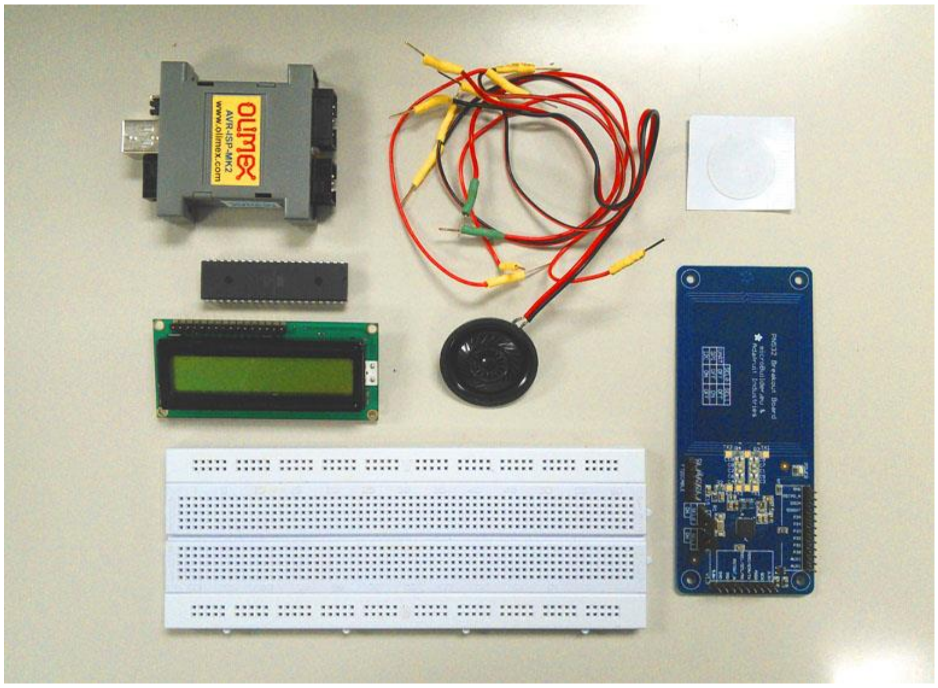
NFC enabled phone
Tag

Experiment 1

We have implemented a simple Octopus System. The NFC tag acts as an octopus card that stores the amount of money. On the other side, the NFC reader acts as the payment system in the minibus. Initially, the tag contains 300 dollars. When the tag is touched on the payment device, the money will be decremented by 3.

The process can be repeated, illustrating the read / write mode in NFC communication.

Components



- Breadboard
- Connectors
- ATMEGA324A microcontroller (Host Controller)
- PN532 NFC/RFID controller breakout board (NFC Interface)
- TS1620-1 LCD Monitor

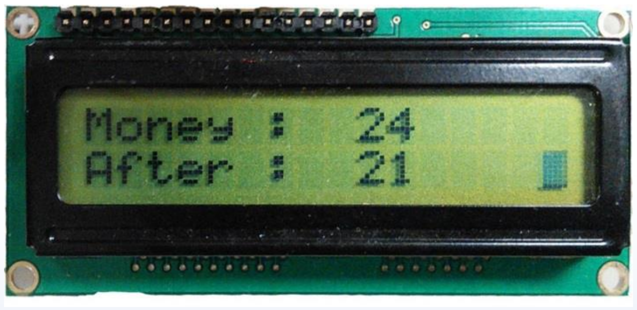
- Speaker
- Mifare Ultralight (NFC Tag)

Equipment

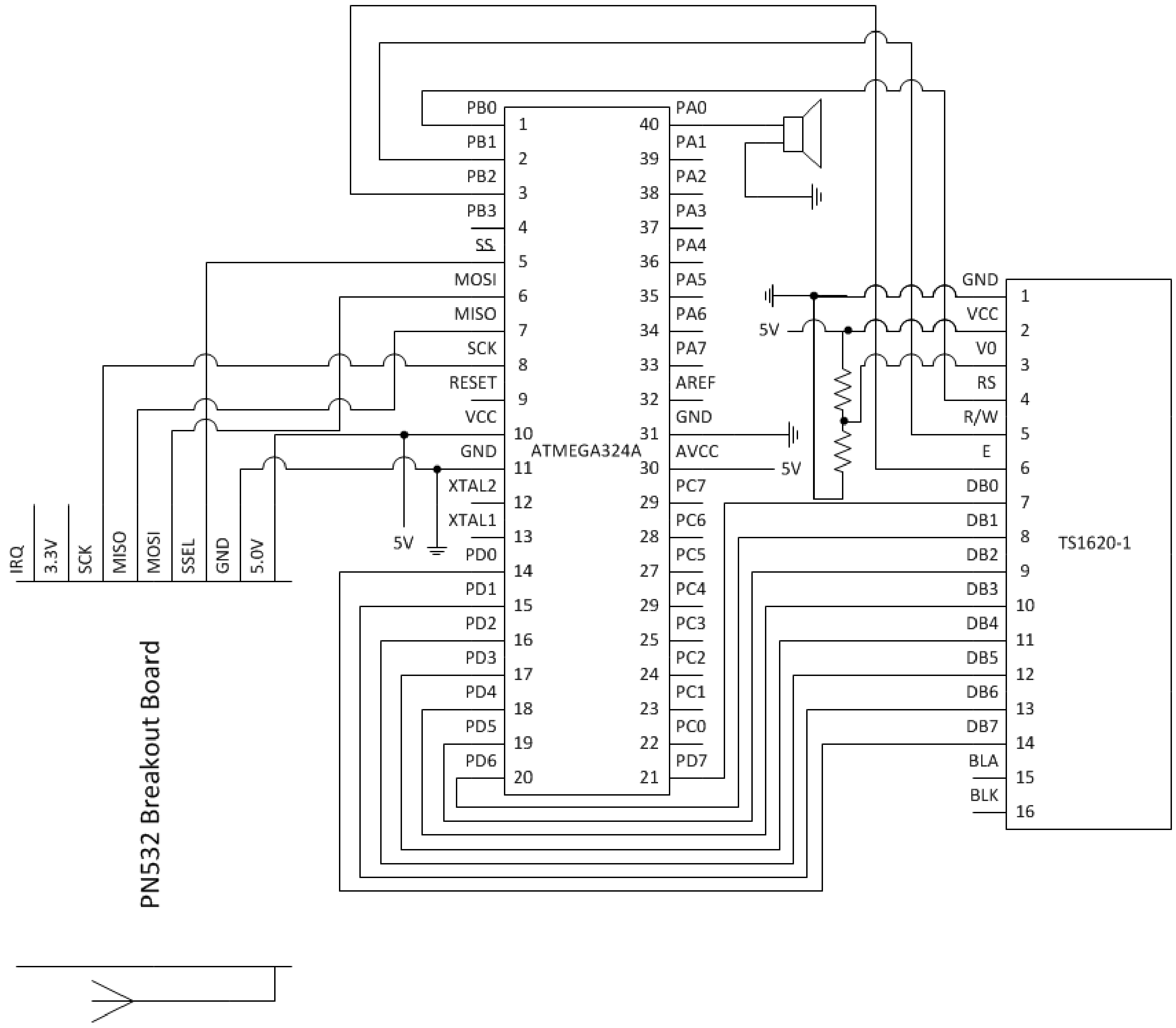
- Computer
- Olimex AVR-ISP-MK2 Programmer
- Atmel Studio

Design

When a valid NFC tag is detected, the host controller will read the data in the tag. Then it will decrement the value by 3 and write it back on the tag. At the same time, the speaker will gives a ‘beep’ sound indicating the transaction is done (It also simulates the sound effect on real octopus system). On the LCD monitor it shows the amount of money before and after.



In addition to the original circuit, we have added an LCD monitor and a speaker. The circuit is modified as follows:



The host controller reads the value from the tag, and then parses the value into integer. After parsing, the value will be incremented by 3 then parse it back. Since the operation is so short, the read and write processes can be done in only 1ms. The possibility of failure is decreased dramatically.

The main part of the code is shown in the following

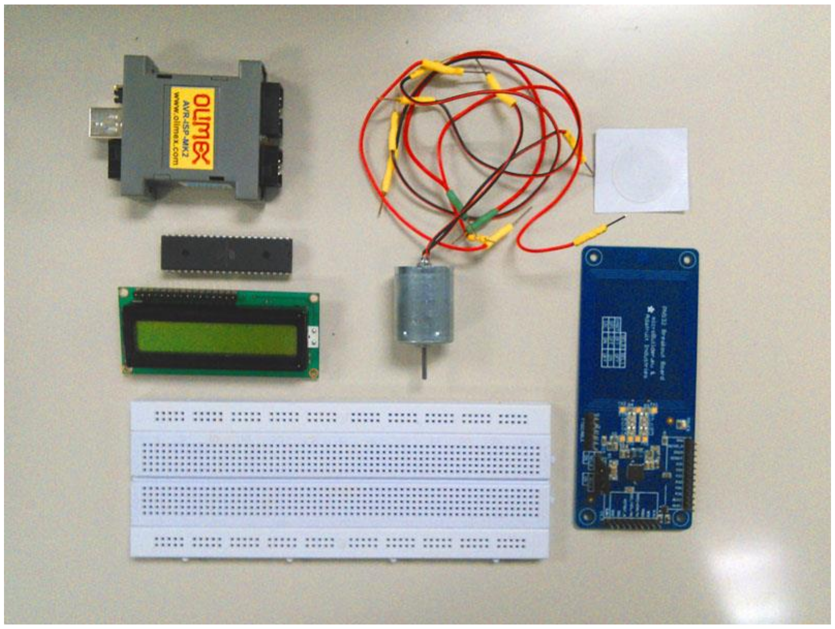
```
    // Initialize NFC Device
    setupPins();
    begin();
    SAMConfig();

    // Initialize 1602
    LCMInit();

    // Main loop
    while(1)
    {
              // Wait for a tag
              DisplayListChar(0, 0, "Put a tag      ");
              while(!readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uid_length));
```

```
// Authenticate Block
if(authenticateBlock(1, uid, uid_length, 6, KEY_A, key))
{
	// Read memory block
	if(readMemoryBlock(1, 6, data))
	{
		// Update the value
		putValue(data, getValue(data) - 3);
		writeMemoryBlock(1, 6, data);

		// Display the amount of money for user
		sprintf(str, "Money : %3d    ", getValue(data)+3);
		sprintf(str, "After : %3d    ", getValue(data));
		DisplayListChar(0,1, str);

		// Play the "beep" sound
		for(i = 0; i < 800; i++)
		{
			PORTA = 0x01;
			_delay_us(625);
			PORTA = 0x00;
			_delay_us(625);
		}
		DisplayListChar(0,1,"                ");
	}
}
```

Experiment 2

In this experiment, we will demonstrate using NFC to control hardware devices such as door lock. The NFC tag simulates a door key, and a motor simulates the door lock. Because each NFC tag has its own unique tag ID which cannot be modified, we use this key to verify the identity of the tag.

Components



- Breadboard
- Connectors
- ATMEGA324A microcontroller (Host Controller)

- PN532 NFC/RFID controller breakout board (NFC Interface)
- TS1620-1 LCD Monitor
- DC motor
- Mifare Ultralight (NFC Tag)

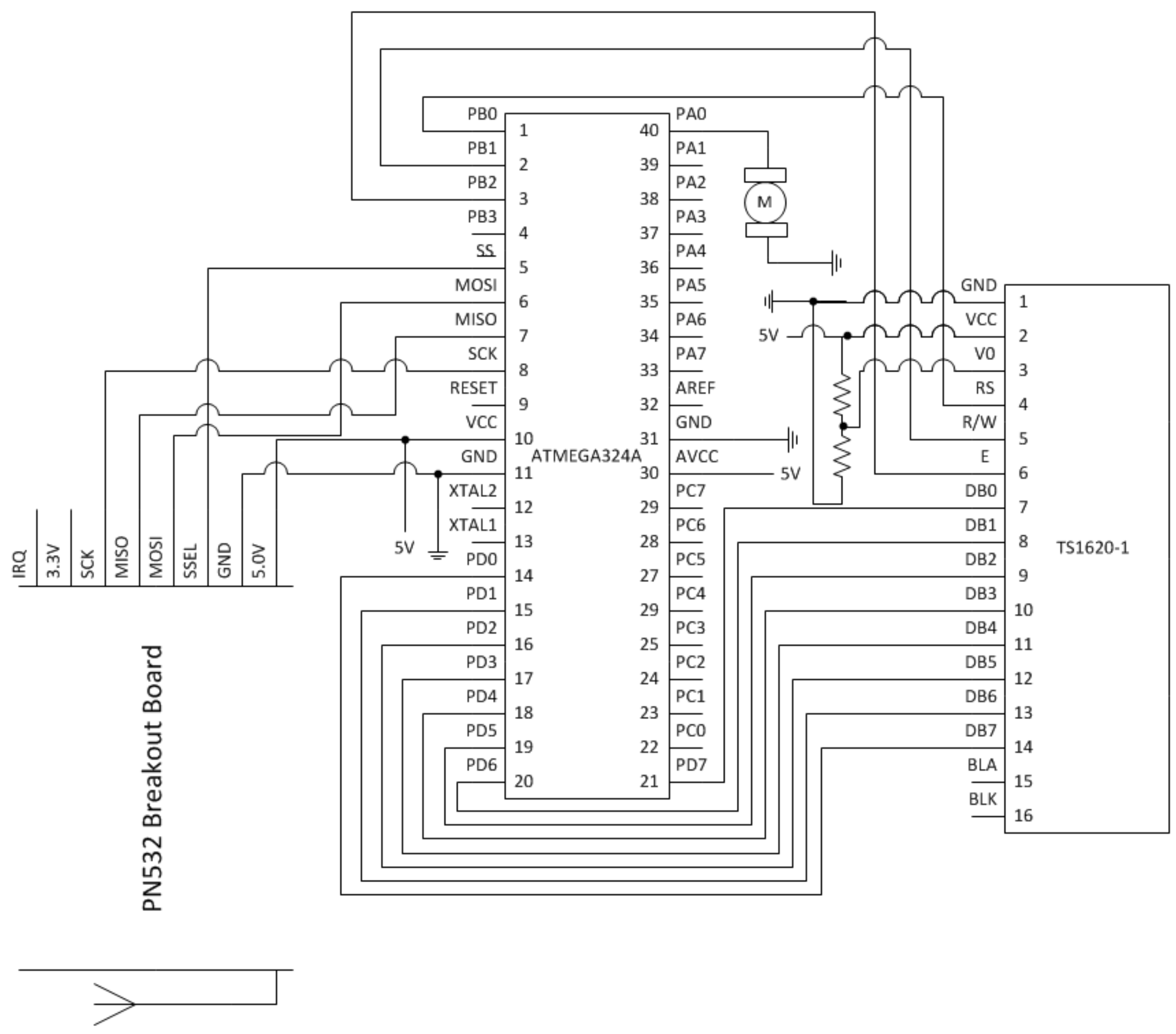
Equipment

- Computer
- Olimex AVR-ISP-MK2 Programmer
- Atmel Studio

Design

Initially, in the host controller, we have hardcoded a key as the unlocking key. When an NFC tag is detected, the NFC controller pass the tag ID to the host controller, then the host controller will compare the tag ID with the hardcoded key. If they are the same, the motor will spin indicating that the door unlocks.

The circuit is similar to that of the previous experiment except that the speaker is replaced by a DC motor. The circuit is connected as follows.



The code snippet is shown in the following, indicating the structure in the single loop processor.

```
	// Initialize NFC Device
	setupPins();
	begin();
	SAMConfig();

	// Initialize 1602
	LCMInit();

	// Main Loop
	while(1)
	{
		// Get card UID
		id = readPassiveTargetID(PN532_MIFARE_ISO14443A);
		while(id == 0)
			id = readPassiveTargetID(PN532_MIFARE_ISO14443A);

		// If card UID is the key
		if((((id >> 16) & 0xffff)==0x7a43) && ((id & 0xffff)==0x2f80))
		{
			// Show unlock message
			DisplayListChar(0,1,"Unlocked!    ");

			// Run motor
			PORTA = 1;
			_delay_ms(2000);
			PORTA = 0;
		}
		// If the card UID is not the key
		else
		{
			DisplayListChar(0,1,"Not this one");
			_delay_ms(2000);
		}

	}
```

Application

In reader/writer mode, there are various possibilities of applications. The data format can be URL, text, multimedia or even a self-defined format. After transmission, the app can use the data to do any kinds of operations. Hence, a wide range of applications in education, remote services, medical and entertainment potentially may be generated using the reader/writer mode.

Smart Poster

The most famous use of reader/writer mode is smart poster. The idea of smart poster is to make the poster alive. An NFC tag is prepared with an URI link, when an NFC enabled mobile device touches the tag, a corresponding website will pop up. This tag will then be stuck on a poster.

Business Card

Business card is another application using the NFC tag. People save his personal information into an NFC tag, whoever wants to get the contact information; he can simply use its mobile phone to touch the tag. Then the contact information will be automatically inserted into his phone.

NFC Shopping

Recently, an electronics retailer opens an NFC shopping wall in Russian subway. On the wall, it listed the photos of products. Customers are required to use its NFC mobile phone to touch the tag on the wall. Then the corresponding website pops up with the product in the online shopping cart. By using the mobile phone to touch the tags, customer will finally get a shopping cart with a list of items. After that, the customer can submit the shopping cart and pay for it online.

2.3.2 Peer-to-Peer Operating Mode (P2P)

P2P mode is the operating mode which allows two mobile devices to communicate with each other. Since, this mode is new from RFID, NFC has its own standards; i.e. NFC Interface and Protocol-1 (NFCIP-1) and Logical Link Control Protocol (LLCP). Besides, both devices are active and one of them has to be the initiator, they communicate in bidirectional half-duplex mode. However, the data rate is only up to 424 Kbps; transmission of large data is not practical. The high security nature of NFC allows mobile phones to exchange sensitive information. Since NFC has short handshaking time, it facilitates the transmission process. This feature also gives another set of applications. NFC can help other connection to finish the handshaking process. For example, user can use NFC to pair up

Bluetooth devices. With the help of NFC, it improves the security of the pair up process and shortens the authentication time.

Application

The transmission data format of Peer-to-peer mode is very similar to that of Reader/Writer mode. Typical data format are text, image, file and business card. Since both are active devices, they have higher flexibility with respect to Reader/Writer mode. Device can change the content of transmission as they wish. Developer is allowed to develop their own protocol on top of P2P mode depends on the application. On the other hand, hardware can use NFC P2P mode to pair up with the mobile phone such as Bluetooth earphone.

S-Beam

One of the largest limitations in NFC is its low transfer rate. It is not capable to transfer large file such as video. However, Samsung invented a new usage of NFC which enhances the transmission to another level. S-Beam allows users to pair up their phones using NFC, then it connects the phones using Wi-Fi-Direct and transfer the file/data through Wi-Fi. Since Wi-Fi has much higher data rate, the transmission speed is improved.

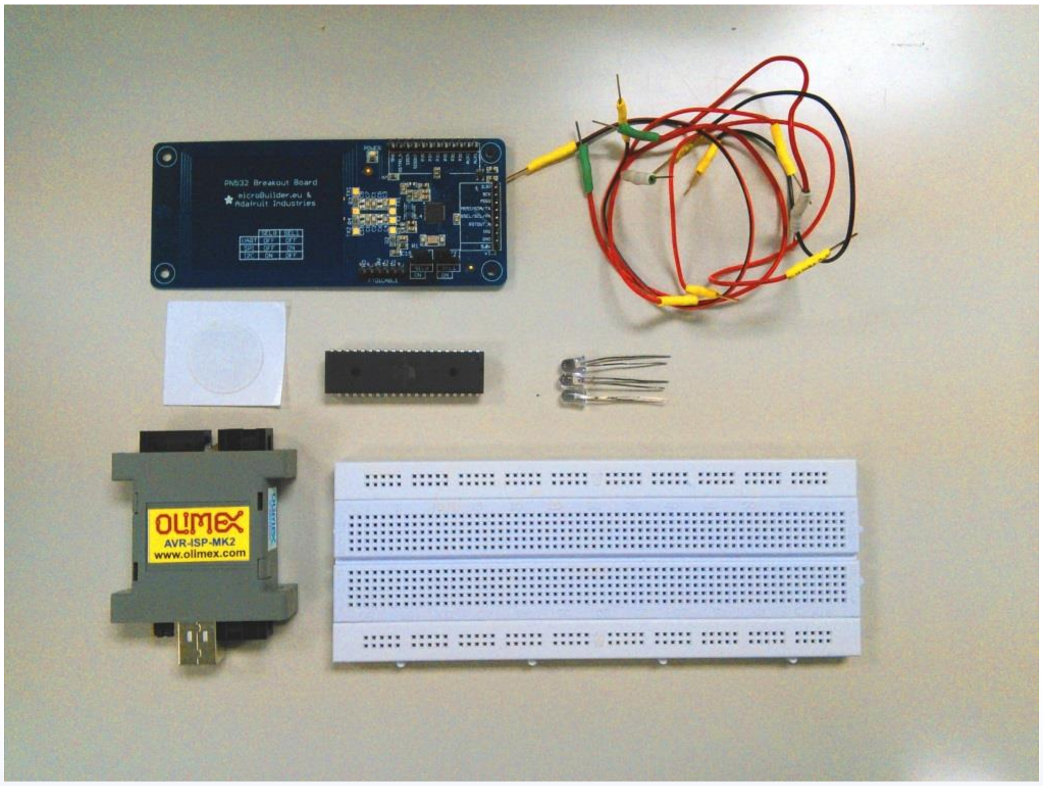
2.3.3 Card Emulation Operating Mode

Card emulation mode allows NFC-enabled mobile devices to function as contactless smart cards such as credit cards and loyalty cards. In this mode, the mobile device acts as a passive tag and the card reader act as an active initiator. The reader generates RF field to “power up” the NFC interface in mobile device (similar to the communication in RFID), then the mobile device sends RF signal using that field. This transmission mode enhances the security. Besides, with the SEs embedded in mobile devices, NFC reader directly communicates with the SEs, hence, the host controller is not allowed to eavesdrop the data in between.

Experiment

This experiment simulates the card emulation mode using the NFC reader and an NFC-enabled mobile device. The mobile device act as a light color controller and a circuit will simulates an NFC enabled lamp. Because individual SE can only owned by the manufacturer for security reason, we have to use an NFC tag to simulate the SE in the phone.

Components



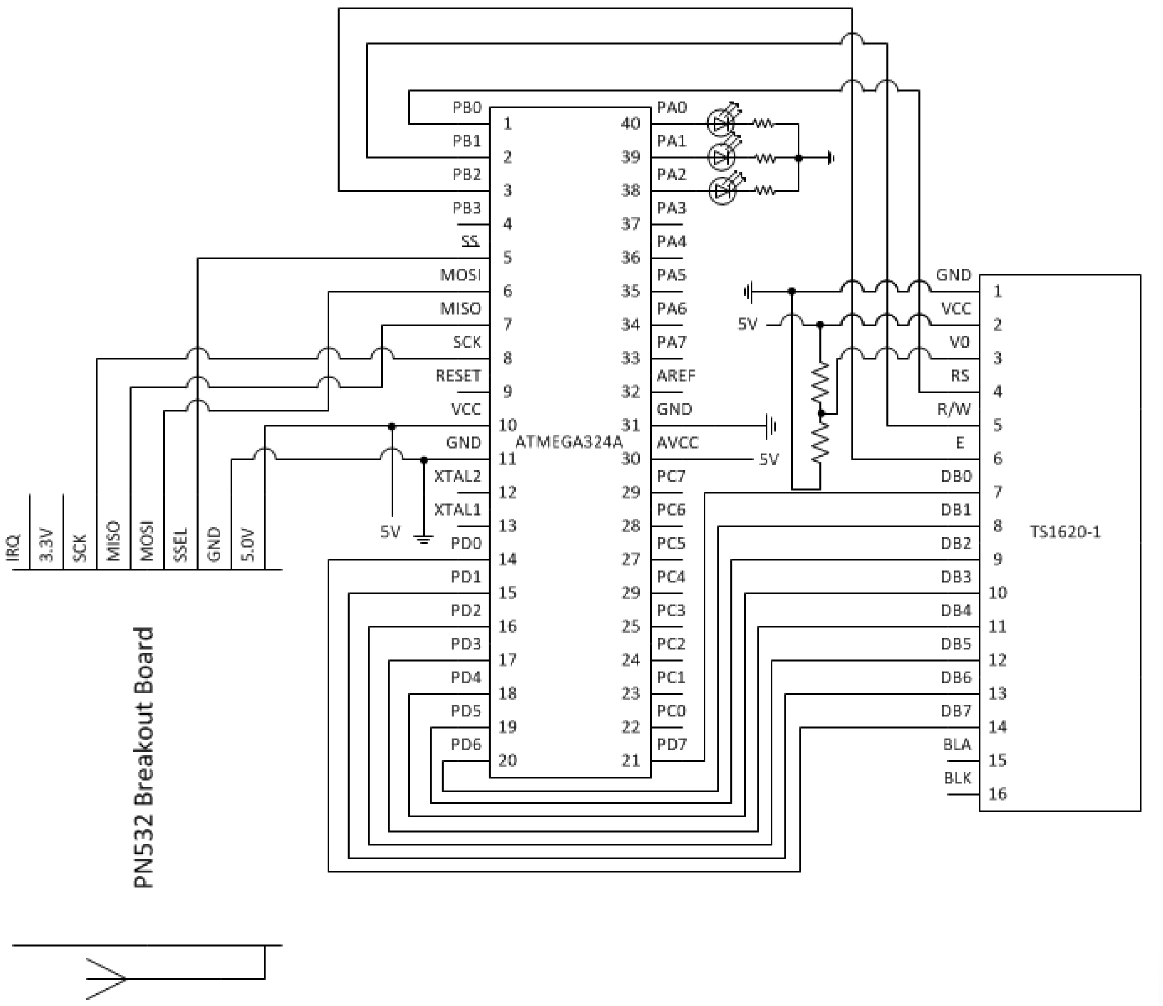
- Breadboard
- Connectors
- Samsung Mega (NFC-enabled mobile device)
- ATMEGA324A microcontroller (Host Controller)
- PN532 NFC/RFID controller breakout board (NFC Interface)
- TS1620-1 LCD Monitor
- Red LED
- Green LED
- Blue LED
- Mifare Ultralight (NFC Tag)

Equipment

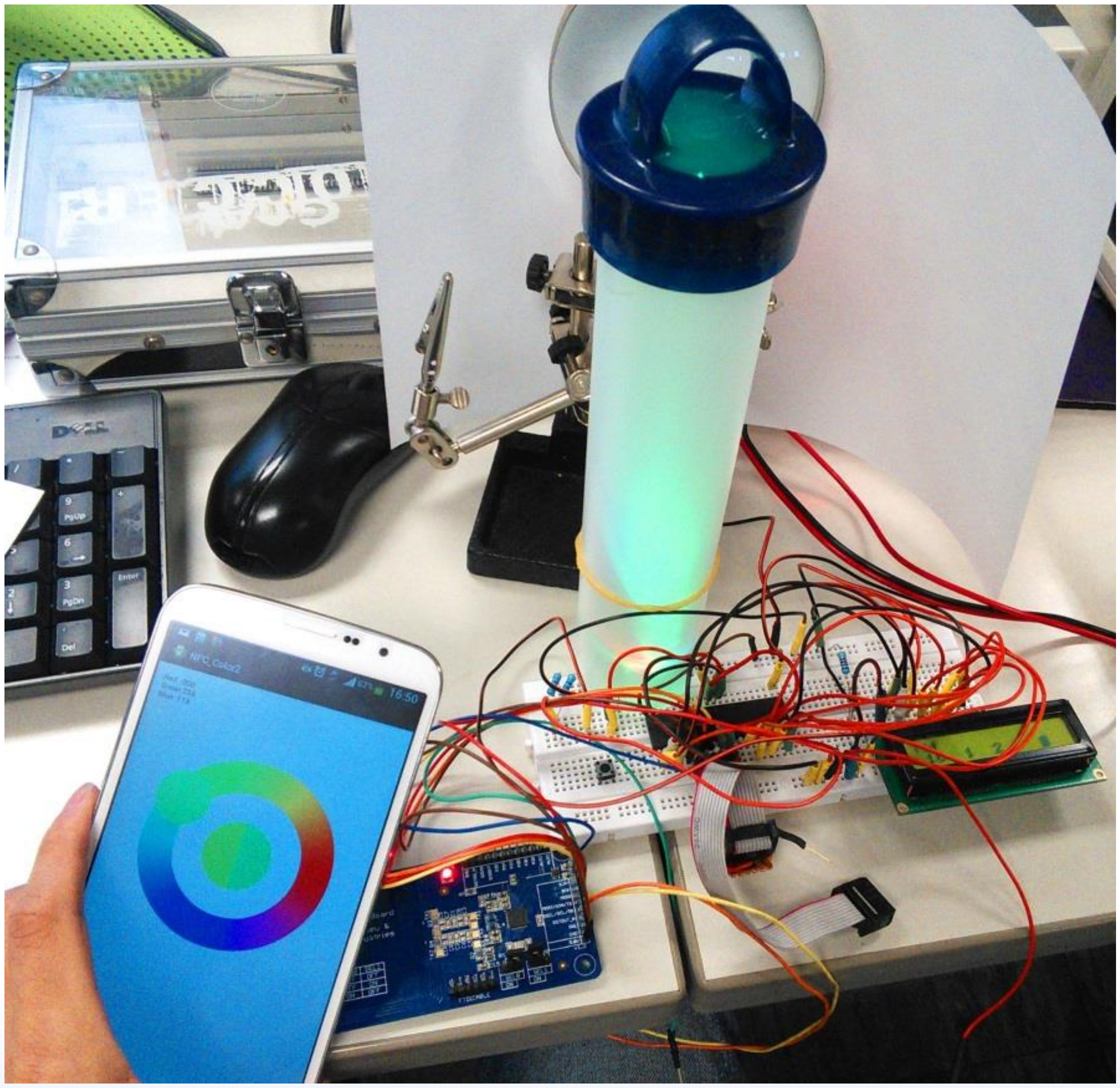
- Computer
- Olimex AVR-ISP-MK2 Programmer
- Atmel Studio

Design

The experiment composed of two devices and an NFC tag. The NFC reader controls the LEDs, each LED represents one color. By varying the intensity of the LEDs, different color can be represented.



The NFC tag contains the information of present color. It sticks on the NFC reader, and the NFC reader continuously monitor the content in the tag then updates the color of the LEDs. In the NFC-enabled mobile device, we have implemented an app which shows a color picker and writes the data into the NFC tag.



Since, the microcontroller cannot output analog signal. Hence, we use frequency to control the intensity of LEDs. Also, single thread nature of the processor makes it difficult to monitor the tag. Therefore, in high level implementation like C, we have to schedule different operation in a while loop. The code is given as follows

```
    // Initialize NFC Device
    setupPins();
    begin();
    SAMConfig();

    // Initialize 1602
    LCMInit();

    // Open LEDs ports
    PORTA = 0x07;
    DDRA = 0x00;

    // Main loop
    while(1)
    {
    // When NFC tag is found
        if(readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uid_length)){
            // Authenticate Block
            if(authenticateBlock(1, uid, uid_length, 6, KEY_A, key)){
                if(readMemoryBlock(1, 6, data)){
                    // Discretize the intensity into 1 to 16
                    r = 16 - getValue(data+1) / 16;
                    g = 16 - getValue(data+4) / 16;
                    b = 16 - getValue(data+7) / 16;
                }
            }
            releaseTag();
        }
        // Blink the LEDs
        for(i = 0; i < DIV; i++){
            if(i%r == 0) DDRA |= 0x01;
            else                DDRA &= 0xfe;

            if(i%g == 0) DDRA |= 0x02;
            else                DDRA &= 0xfd;

            if(i%b == 0) DDRA |= 0x04;
            else                DDRA &= 0xfb;
        }
    }
```

Application

The application of card emulation is very strict forward; it simulates different kinds of cards. By making use of the SE in the mobile device, the mobile device can serve as a credit card, debit card, loyalty card, ticket, access control unit and identity card.

Door Key

Hotel commonly uses an RFID card to unlock the door, however, a passive RFID card is not flexible and guest can steal it easily. NFC technology is proposed to use in hotel's door lock system. When a guest books a room from hotel, room information and digital key are sent to the guest's mobile phone. Since, the mobile device is active; a timer can be set inside the mobile phone. We can also extend the application to house door lock. One can use mobile phone to unlock its door, or even transfer a temporary key to a friend through the internet.

Payment

One of the big focuses on NFC is the replacement of credit card. By adding applications into SEs, one mobile device can emulates a set of credit cards and debit cards. When an NFC reader triggers the mobile phone, only one corresponding application in SE is allowed to communicate.

2.4 NFC Tag

NFC tag is an RFID tag that has no integrated power source. It is a low-cost and low-capacity passive device. Since it does not contain any power source, communication needs to be triggered by an active device. The active device use its RF fields to power up the NFC tag, then the communication can be start.

2.4.1 NFC Forum Tag Types

According to the NFC forum, NFC tag is defined into four tag types. Each tag types has different capacity and format.

Parameter	Type 1	Type 2	Type 3	Type 4
Based On	ISO/IEC 14443 Type A	ISO/IEC 14443 Type A	FeliCa	ISO/IEC 14443 Type A, Type B
Chip Name	Topaz	MIFARE	FeliCa	DESFire, SmartMX-JCOP
Memory Size	Up to 1 KB	Up to 2 KB	Up to 1 MB	Up to 64 KB
Data Rate	106	106	212	106-424
Cost	Low	Low	High	Medium/High
Security	16-bit or 32-bit digital signature	Insecure	16-bit or 32-bit digital signature	Variable

Type 1

Type 1 utilizes a simple memory model with communication speed 106 Kbps. Since is the most simple type, this tag type is cost effective and can be used in most NFC applications. The tag belongs to this type is readable and writable. It can be reformatted as read-only tag. The memory size is only up to 1KB; hence it is only just enough to save URLs and text messages. For larger data such as contact information and images, type 1 tag is not capable to store it. The structure of the tag is rather simple; it has two memory model mappings, static and dynamic, depending on the size of the tag. The tag is divided into blocks; each block contains 8 bytes of data.

Type 2

Type 2 tag is very similar to Type 1 tag. This tag type is also readable and writable, but the only difference is that the memory size is expanded to 2KB and no digital signature embedded inside the tag.

Type 3

This tag type is defined base on Sony FeliCa contactless smart card interface, hence it is more suitable for complex applications. It is also writable but with block size 16 bytes. Hence, its maximum memory size is 2KB. Due to the technology backed by contactless smart card, the capacity can be expanded into 1MB. The data communication speed is doubled to 212 Kbps, it is the most expensive among four.

Type 4

NFC tag of type 4 can be preconfigured into read-only in the manufacturing phase. This can guarantee the security of the NFC tag, since no one including the buyer can modify the data inside the tag. The capacity can be up to 64KB which has a larger capacity than Type 1 and Type 2.

2.4.2 Major Proximity Contactless Smart Card Technologies

NFC tag is developed on contactless smart card, hence they have the same fundamental structure. Up to now, the most famous and competing contactless smart cards are Mifare, Calypso and FeliCa.

MIFARE

Mifare is a widely used contactless proximity smart card system owned by NXP Semiconductors. The Mifare family contains different types of cards, such as Mifare Classics, Mifare DESFire, Mifare Ultralight and Mifare Ultralight C. Mifare-based smart card is widely used in range of applications, access management, e-payment and loyalty applications such as student card (CULink) in The Chinese University of Hong Kong.

FeliCa

FeliCa is a high-speed proximity smart card system from Sony. It is designed for electronics money card. Primary application of FeliCa smart card is public transport ticketing, for example the Octopus card in Hong Kong Mass Transit Railway (MTR).

2.4.3 NFC Tag vs. QR Tag

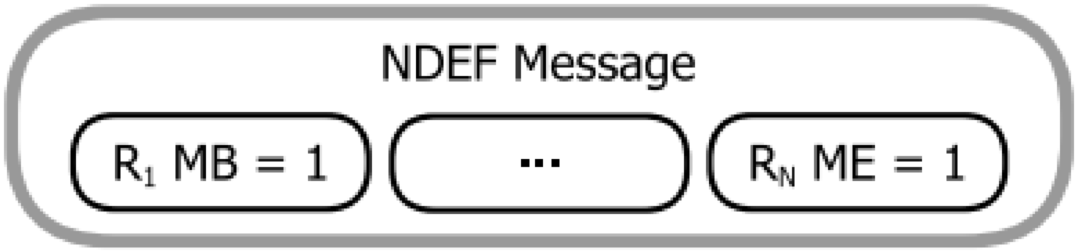
QR code is a 2-dimensional bar code that stores a short piece of information. The data type of QR code is usually URL website address. Common mobile phones contain cameras, hence mobile phone can use the camera to capture the QR code then decode it into URL message. The usage of QR code is similar to that of NFC tag, comparison between them can emphasize the characteristic of NFC tag.

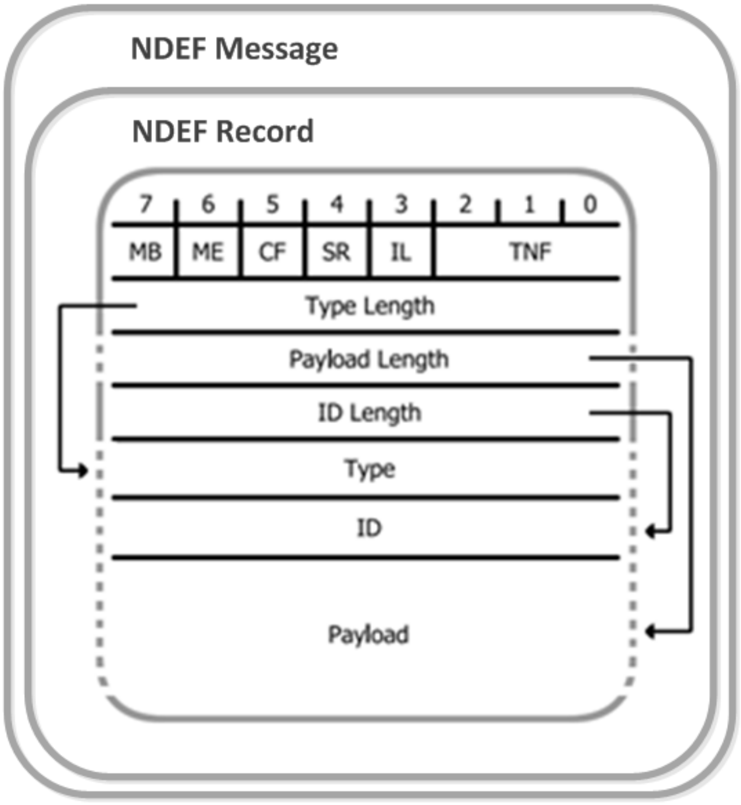
QR code requires user to use a camera app to focus on the barcode, the whole process (launch the app, focus on the QR code, wait until the app recognize the QR code) used approximately 10 seconds. But using NFC only needs 2 seconds, the corresponding app will be launched after the transmission. However, QR code is very easy to create. Only a black ink printer is needed to make a QR code. [2]But NFC tag requires a special circuit embedded inside the tag, which is not easily available. In the issue of security, NFC tag offer a unique manufacture ID number and specialist tags are also support encryption. For the QR code, it offers no security.

	NFC Tag	QR Code
User Experience	Immediate, no additional software	~10s, requires launching an app before scanning
Cost	HKD$ 1.00 – 10.00	~HKD$ 0.007
Size	10-30mm (diameter)	>20mm
Product Integration	Can be embedded inside a product	Must be printed visibly
Print and Customization	Full color custom print	Limited space without affecting the barcode
Availability in Mobile Phones	10% of smartphones	All smartphones
Programming	Mobile apps available on NFC-enabled smartphones	Wide range of websites with QR generation service
Security	UID, Encryption support	No

2.4.4 NFC Data Exchange Format (NDEF)

NDEF is the data format to exchange information between NFC devices. It is also the data structure inside an NFC tag. The NDEF message can be sent either in NFCIP-1 or LLCP. In one transmission, it can include more than one NDEF message. In each NDEF message, there are some NDEF records. Each record contains one piece of information in any defined format.





In an NDEF record, it has a definite data structure with varies fields. One of the fields, Type Name Format (TNF) defines the format of the data type. This field is a 3-bit field with values defined in the table

Type Name Format	Value
Empty	0x00
NFC Forum well-known type	0x01
Media-type	0x02
Absolute URI	0x03
NFC Forum external type	0x04
Unknown	0x05
Unchanged	0x06
Reserved	0x07

The NFC Forum well-known type defines many useful data types, such as smart poster, text, URI and signature. NFC Forum also allows developer to define his own data type in NFC Forum external type. The CTag used in Cucom is defined from this type.

The Payload is the real content of the data. Since payload length is a 32-bit unsigned integer, the payload consists up to 2^{32}-1 octets in size, which should be more than enough compare to the maximum data rate 424Kbps.

3 Studies On Android NFC Package

In Android SDK, NFC is already supported at API level 9. Since Cucom is only used Reader/Writer mode, we only discuss on the operations in this mode.

3.1 Android NFC Package

Android SDK provides two packages for different usages. android.nfc package is used in high level operation such as reading NDEF contents in known structure. android.nfc.tech package focuses on low level operation, it provides classes for different tag types, in each class it also provides methods to do low level tag operations.

3.2 NFC Dispatch System

There are two NFC dispatch systems in Android Reader/Writer mode. Intent dispatch system and Foreground dispatch system, each has its own characteristic in the application level. The tag intent dispatch system is used to launch applications when the predefined tags or NDEF data are identified in tags. After scanning an NFC tag, the corresponding registered application will be launched. For example, after installation of Cucom, the application is automatically registered for CTag. Hence, when the mobile phone touches to the CTag, Cucom will pop up and shows the detail of the CTag.

On the other hand, the foreground dispatch system is designed to handle tags when the application is running. After the foreground dispatch system is enabled, it will stand by and wait for any registered NFC tag. When the Android device touches the NFC tag, the applications registered as in Intent dispatch system will be ignored, since the priority of foreground dispatch system is greater than that of Intent dispatch system. For example, when Cucom is ready to write the content back to the CTag, it enables the foreground dispatch system and wait for a CTag.

In coding, tag intent dispatch system requires application registered the NFC tag types or NDEF data types in the manifest file using intent filter. While the foreground dispatch system registers inside an activity.

3.2.1 Tag Intent Dispatch System

When an NFC tag is scanned, the NFC controller notifies the Host controller of the mobile phone, then the OS will reads the content and UID of the NFC tag. Depends on the content, the OS will look for any registered application. If an activity of an application is registered to this content, OS will intent to that activity. If more than application is registered to the same content, an application chooser will pop up showing the registered applications; in this case user has to choose their preferred application. On the other hand, no application registered to this content, and then the NFC action will be ignored.

In registration, application can register to a specific NDEF data type such as URI and text. However, due to different situations, the content inside the NFC tag may not be concise enough. For example, the scanned tag may contain a valid NDEF structure and a known NDEF payload data type. A tag can also contain a valid NDEF structure but an unknown NDEF payload data type. Even worse, the scanned tag does not contain a valid NDEF structure but it is a known tag technology. Due to the above options, three different intents are developed: ACTION_NDEF_DISCOVERED, ACTION_TECH_DISCOVERED and ACTION_TAG_DISCOVERED.

ACTION_NDEF_DISCOVERED

This intent has the highest priority among three. When a tag contains a valid NDEF data type, the OS will try to run the registered application. For instance, Cucom registers this intent for CTag. When an NFC tag is touched and it contains a valid NDEF format with CTag data type, OS will launch the corresponding activity in Cucom.

ACTION_TECH_DISCOVERED

This intent has the second highest priority. If a valid NDEF format is detect from the scanned tag, but no activity has registered to the data type, the OS will call the ACTION_TECH_DISCOVERED intent. In this intent, application has to register to a tag type, such as Mifare Classic and Mifare Ultralight. In Cucom, when the like/dislike votes is ready to write it back to CTag, it enables the Foreground dispatch system to monitor the ACTION_TECH_DISCOVERED intent. Whenever, the supported tag type is scanned, Cucom will validate the data format in the tag itself. When the format is invalid or destroyed, Cucom can repair the tag by itself which enhance the robustness of the application.

ACTION_TAG_DISCOVERED

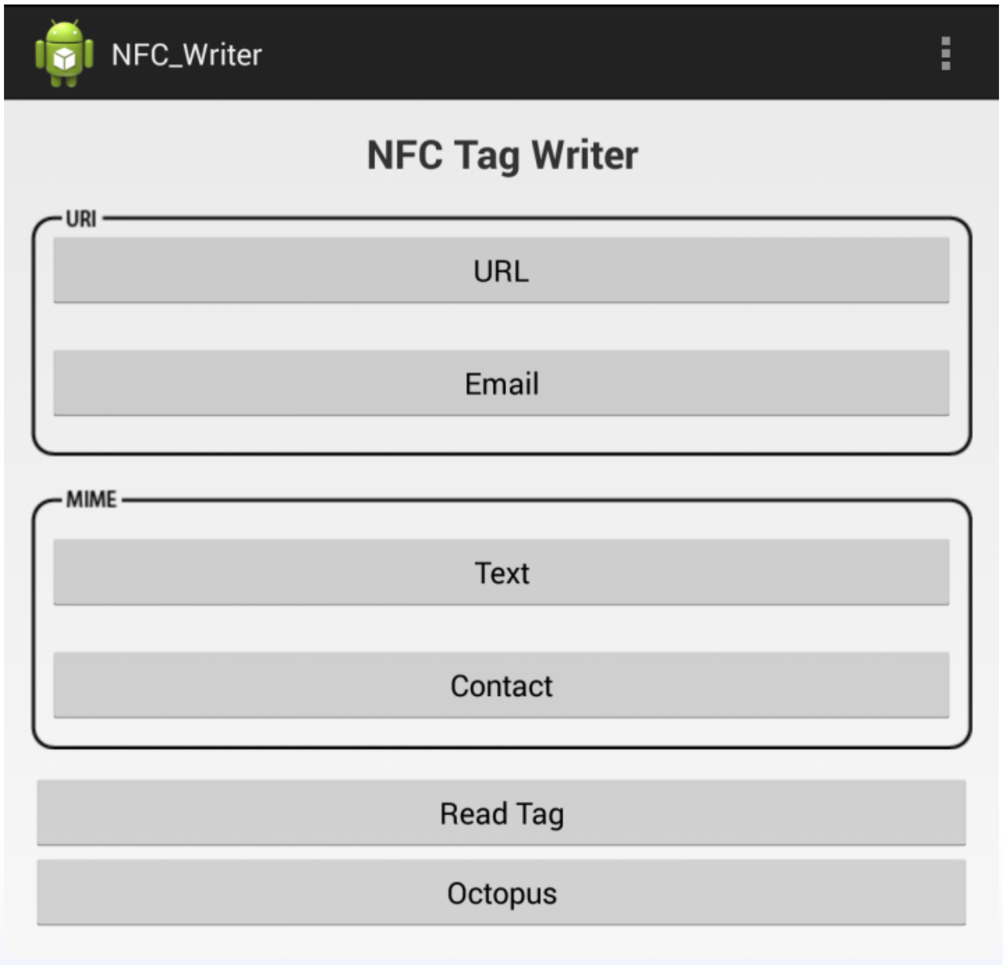
This intent has the lowest priority. If an NFC tag is detected but the NDEF data type is not recognized, or even the tag type is not recognized, then ACTION_TAG_DISCOVERED intent will be called. In this case, the programmer has to write the communication protocol by himself. This intent is rarely used.

3.2.2 Foreground Dispatch System

Foreground Dispatch system handles the NFC tag when the application is in active states. The priority of Foreground dispatch system is greater than Tag intent dispatch system. Hence, the active application has the full control of the incoming NFC tag. This is useful in writing operation.

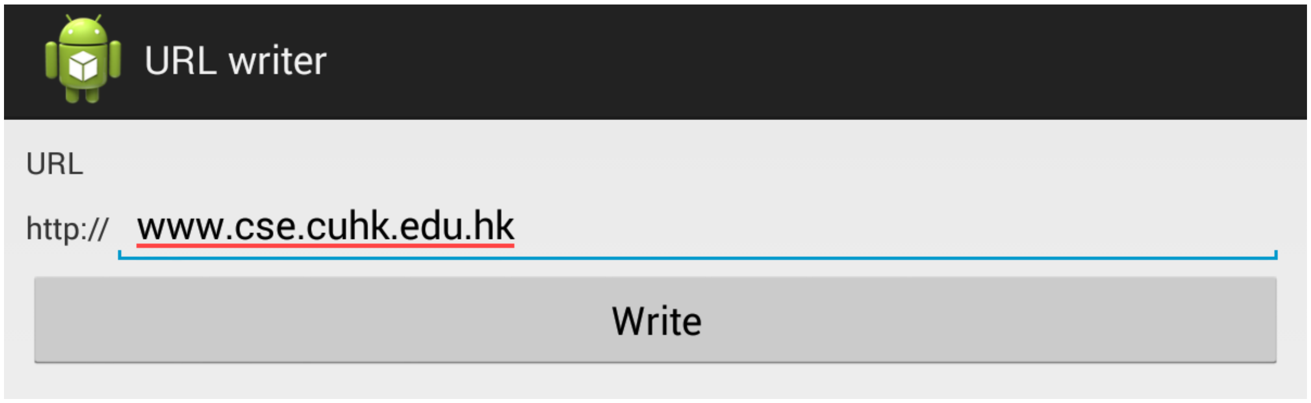
3.3 Experiment

Before we begin to develop our Android application using NFC (Cucom), we have written an NFC tag writer. This application can writes variety data types into a tag. We also learned UI interface through developing this app. This application not only demonstrates the usage of NFC Foreground dispatch system, it also demonstrates the intents between activities as well as the 9-patch image.



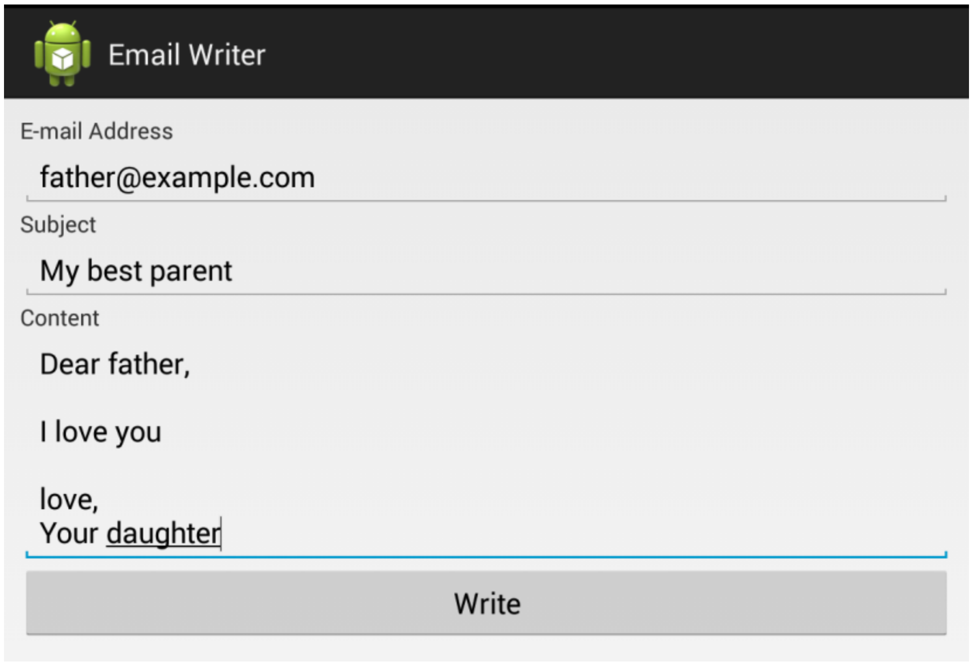
Specifically, the application can writes two URI formats and two MIME (Multipurpose Internet Mail Extensions) formats. It can also read the NDEF record in the NFC tag. It uses the Foreground Dispatch system to do all the read/write operations.

3.3.1 URI Writer



The URI writer allows user to write an HTTP link into the tag. User inputs the URL such as http://www.cse.cuhk.edu.hk/ then press write. The NDEF message of the link will be constructed and will be sent to the activity of Foreground dispatch system. When an NFC enabled android device touches the NFC tag, the browser will pop up with the link stored in the tag.

3.3.2 Email Writer



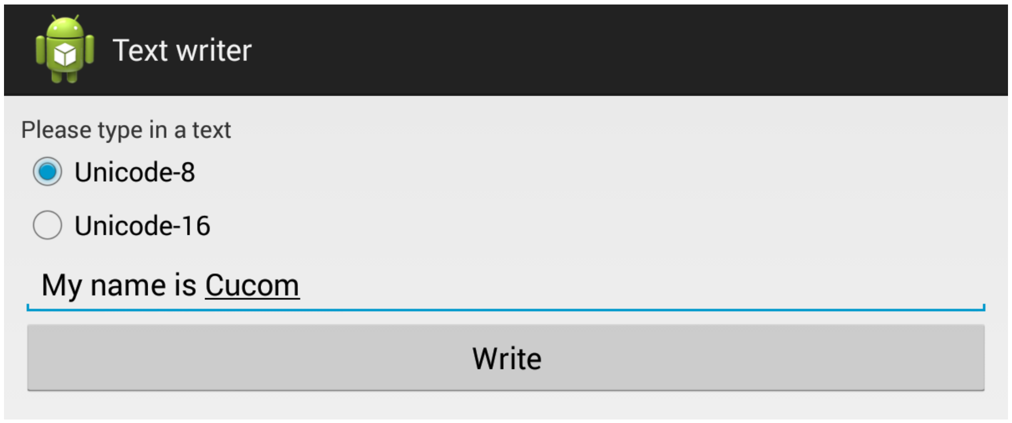
E-mail Address
father@example.com
Subject
My best parent
Content
Dear father,

I love you

love,
Your daughter

Email writer allows user to input the target email address, subject and content. The NFC tag stores all the details, when the phone touches the tag, the default email application will pop up. In the application it automatically creates a new email filled with the information stored in the tag. It is useful to create a framework for emails.

3.3.3 Text Writer


Text writer

Please type in a text

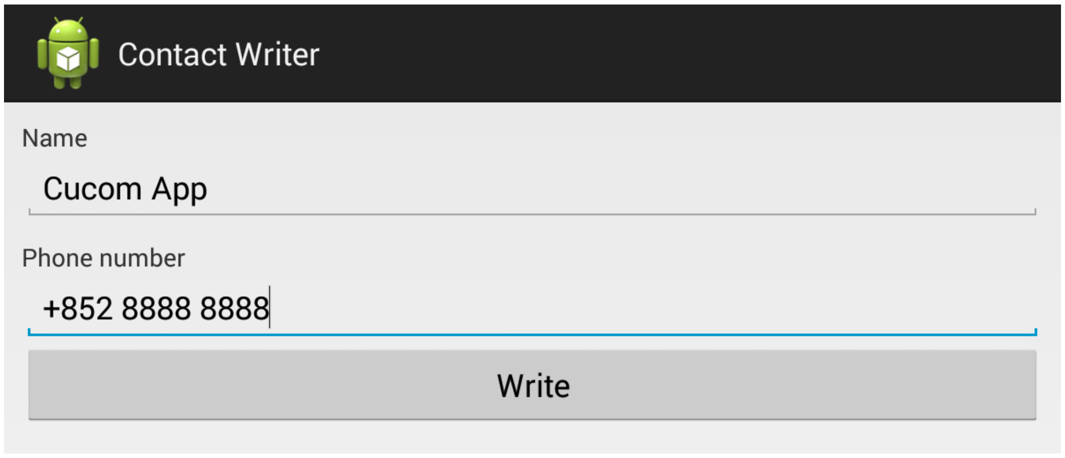
[✓ Unicode-8]

Unicode-16

My name is Cucom

Text writer simply writes a text into an NFC tag. User is free to choose between Unicode-8 and Unicode-16 input format. There is no default application that handles text message. Hence, user needs to install a specific application for text message.

3.3.4 Contact Writer


Contact Writer

Name

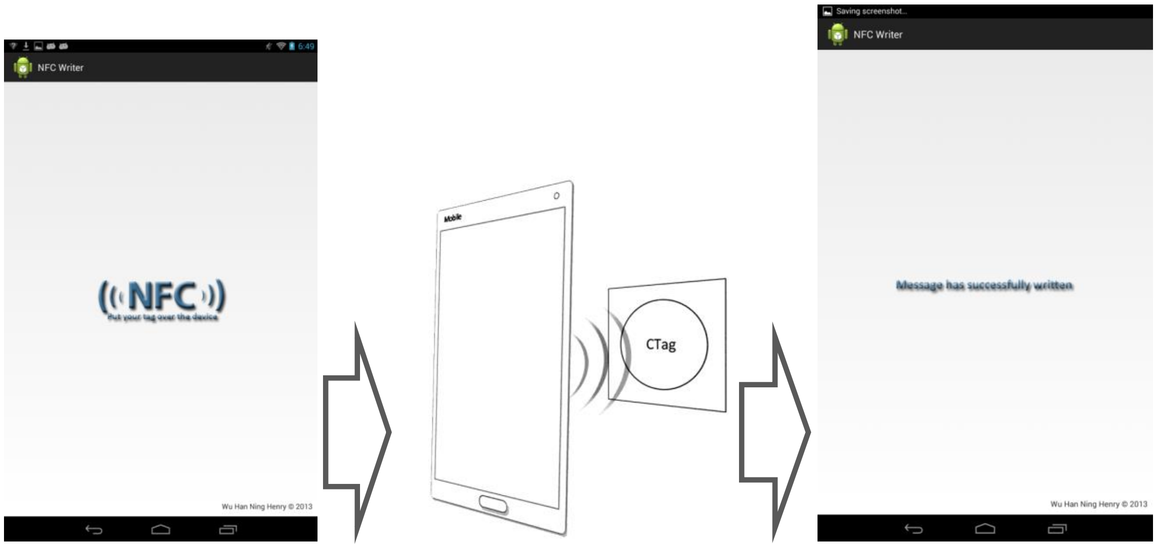
Cucom App

Phone number

+852 8888 8888

This writer enables user to store simple contact information into a tag. The tag immediately turns into a business card. When a mobile phone touches the tag, it automatically adds the contact into the phone.

3.3.5 Foreground Dispatch System Activity

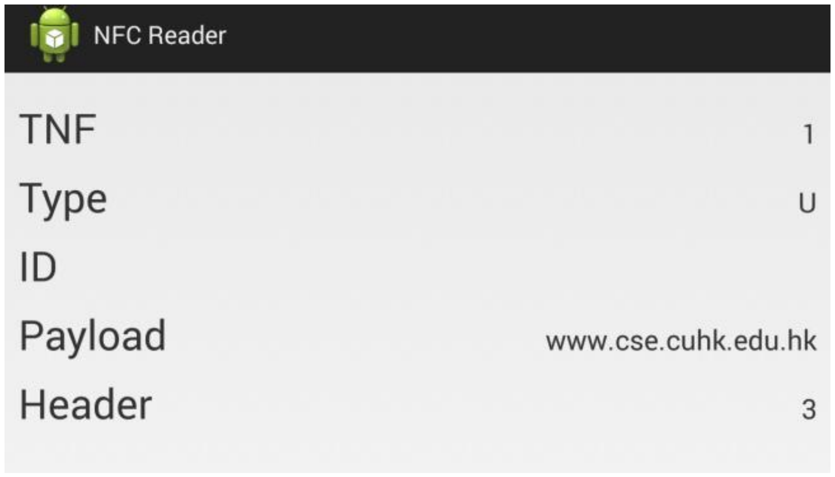


Put your tag over the device

Message has successfully written

When the user press write from the above activities, it will intent to here. In this activity, it enables a foreground dispatch system, user needs to put the phone on the NFC tag and wait for writing operation. When the operation is done, an acknowledgement will be displayed on the screen.

3.3.6 NFC Tag Reader


NFC Reader

TNF	1
Type	U
ID	
Payload	www.cse.cuhk.edu.hk
Header	3

This activity also equips a Foreground dispatch system. When a tag with a valid NDEF is detected, it reads the content in the tag, and then shows it on the screen.

4 User Experience

In the definition of ISO 9241-210, user experience is defined as ‘a person's perceptions and responses that result from the use or anticipated use of a product, system or service’. The user experience is an important role for a system since user experience just like the information architecture. The bad information architecture will lead to ineffective, inefficient and unsatisfying experience of using the system for the users. Note that using mobile device is different from using desktop computer. With a smaller display and different styles of user interaction, the design for mobile applications becomes strong influence in application development. The paradigm of mobile user interface is based on widgets, touch, motion and keyboards, which is different to the common WIMP (Windows, Icons, Menus, Pointer) interface style of Apple’s iOS and Microsoft Windows. With the limited screen size, UI gesture becomes an important role on app development.

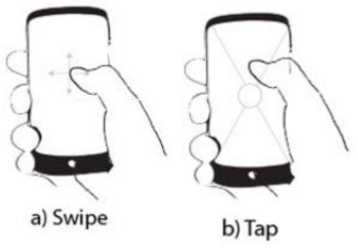
4.1 UI Gesture

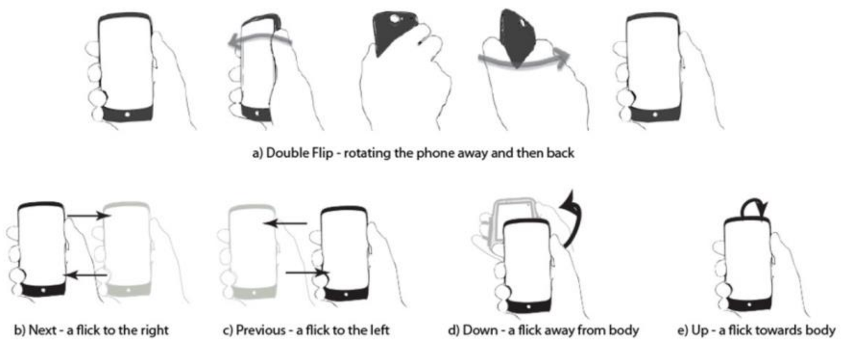
Most of the people are distracted by another task when using smartphone. For example, walking or taking MTR. Thus, there are two principles of the user interfaces for apps when users are distracted:

1. The interface should limit the need for visual attention during interaction.

2. The interface should provide streamlined commands for the most common tasks

These principles can be take benefit by applying suitable gesture. There are three alternative input forms supported by smartphone: tap, swipe and physical move.

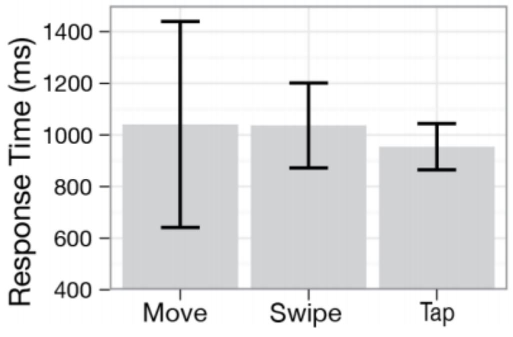



a) Double Flip - rotating the phone away and then back
b) Next - a flick to the right
c) Previous - a flick to the left
d) Down - a flick away from body
e) Up - a flick towards body

Each of them has their advantages and disadvantages. Since our project aims to encourage users to be interactive with the physical world, we mainly focus on the performance of moving scenario. The following are the different statistic measured when the participants are walking. The statistics are response time, success rate, screen gaze, walking speed and the times lost.

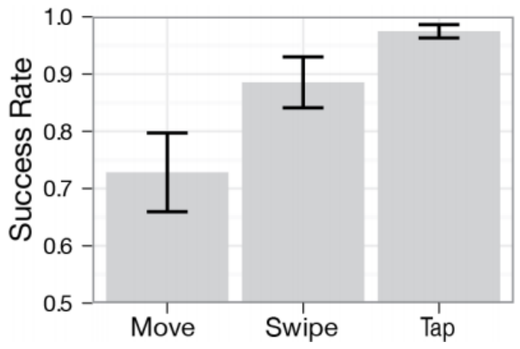
4.1.1 Response time

The response time is related to the difficulty of taking corresponding action after given the commands. The graph shows that the response time of Move and Swipe are slightly longer than the Tap.



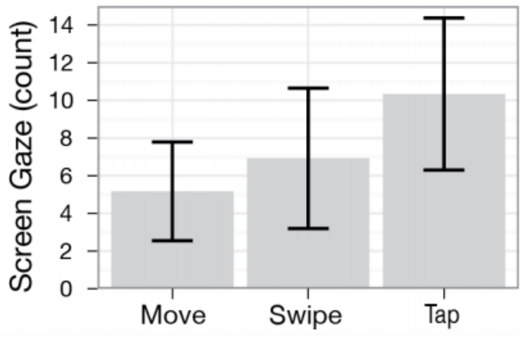
4.1.2 Success rate

The success rate is the percentage of performs the command successfully. The success rate of Tap and Swipe is higher than that of move.



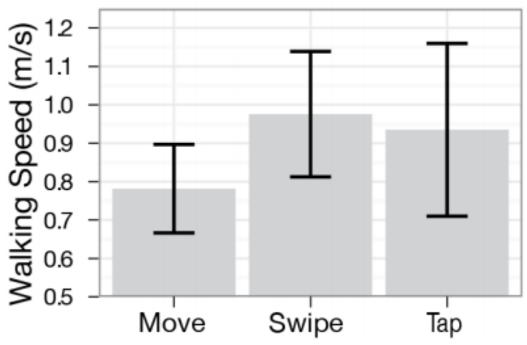
4.1.3 Screen gaze

For the screen gaze, the number of times that participants looked at the smartphone screen for Move and Swipe are less than that of Tag.



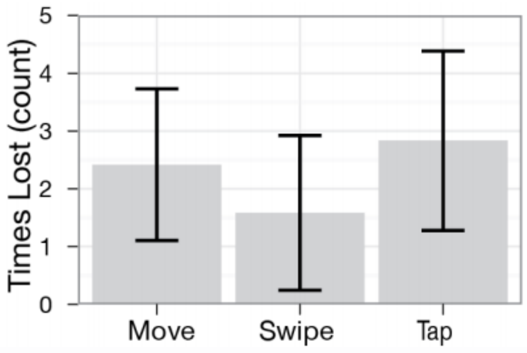
4.1.4 Walking speed

It is the walking speed when the participants perform these actions. The result is that the walking speed of Swipe is slightly higher than that of Move and Tap.



4.1.5 Times Lost

The times lost are the number of times participants had to stop and accommodate themselves. The number count of Swipe is the least among others.



According the above figures, we found that the performance of Swipe in different aspects are the most stable one on moving scenario.

4.2 Classification

Classification may refer to the categorization, which is the way that how the ideas or objects to be recognized, differentiated and understood. For a search engine, it is important to use classification to classify the data in order to search information efficiently. A good classification not only can reduce the group of data for searching, but also provides an overview of the data so that the users can view it clearly.

The classification has many types, but we will focus on two types: Hierarchical and Tagging.

4.2.1 Hierarchy

Hierarchical classification is a method of grouping in which terms are arranged from general to specific that the structure is initially arranged in broad groups which are then mutual exclusivity subdivided into narrower groups. It can be applied to the parent-child hierarchies. The examples of the application of taxonomy are animate objects, places, concepts, events, properties, and relationships.

The advantage on using a hierarchical classification is efficient in both learning and representation. Each sub-problem is smaller than the original problem, and it is sometimes possible to use a much smaller set of features for each.

4.2.2 Tagging

Tag is a non-hierarchical keyword or term assigned to a piece of information. It is generally chosen informally and personally by the item's creator or by its viewer, depending on the system.

4.2.3 Hierarchy vs. Tagging

Hierarchy gives a better structure and efficiency in searching. It also has a high consistency. With hierarchies user is limited to one category in which to place information. Classifying the CTag using location is not satisfied. For example, some may want to look for restaurants; others may want to look for teaching buildings of Medicine. Depends on the usage, Hierarchical classification does not suit this kind of query. However, tag can fit into many ‘categories’ and the results are much more contextual.

4.2.4 Classification in Cucom

CTag represents a location in Chinese University, tagging suits the use. However, we do not allow user to create CTag freely, so it is possible to make use of the benefit of Hierarchical classification. Therefore, we choose to use nested tagging. Each CTag is attached with a tag. All tags are classified into 3 categories; District, Faculty and Facility. The classification structure is similar to that of OpenRice. This classification enhances user-friendliness, user can loop for multiple tag with AND and OR relations

5 Cucom Overview

Cucom (Chinese University Communication) is a social app where users can use it to comment on physical environment in The Chinese University of Hong Kong. When we visit some places, we may want to know the information of that place. Besides, we may want to discover some new famous places to visit through the internet. The best way to know the TRUE picture is by reading others comments. Most of us simply use the mobile phone to browse some searching engine and find the place when they are outside. But it has some problems:

1. Searching is troublesome. We need to open the browser, type the keywords. It is even worse when we use mobile phone outside as typing keyword in a tiny screen is difficult.
2. Searching is time-consuming. We need to scan the search results in order to find some useful comments.
3. Users do not aware with the outside world as they only focus on reading in mobile phone.
4. The information of search results may not be true since like and dislike is easily available. For example, some of the companies will hire people to

register fake accounts in order to “like” the company facebook page or leave bias comment to increase popularity.

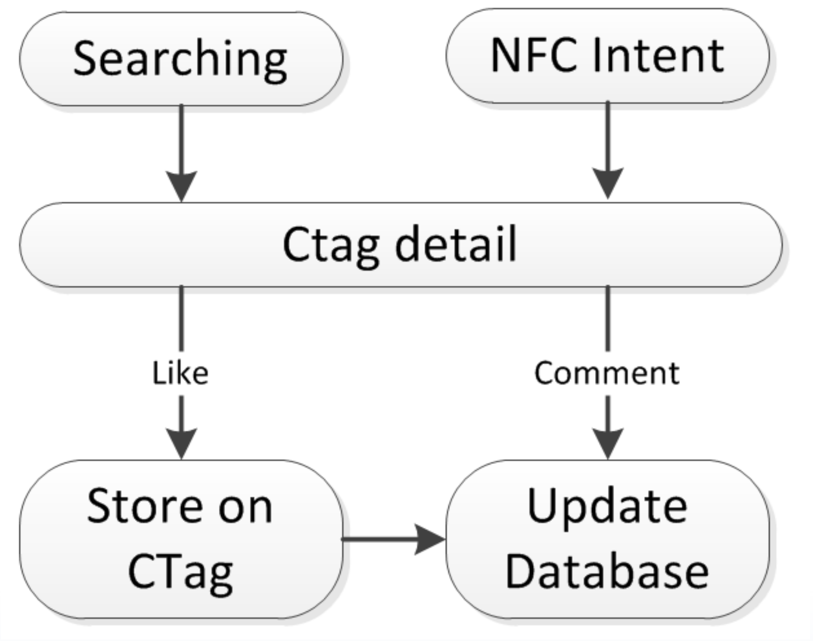
If the physical object itself allows us to view its details, it can solve the above problems. And it will be very useful and interesting for us. In this scenario, applying NFC Tag in the place can attain this effect and solve these problems:

1. Scanning the NFC tag that store the information of that place are easier for users.
2. NFC Scanning is faster than reading numbers of search results.
3. Users will interact with outside world when they scan the NFC tag
4. Editing NFC tag requires users to go to the corresponding location which lower the biasness

Cucom makes uses of these advantages and creates a new culture in the Chinese University. There are numbers of CTag (NFC Tag) located in different places that people visit frequently, when users visit these locations and want to know the details of those places, they can simply scan the CTag and Cucom will pop up listing the information about it. Users do not need to do searching any more. After that, users can vote and leave comments to share with other users. We can see most of the people like to express their feelings and opinion by the success of

Facebook. Hence, adding voting and commenting function will make the app more attractive. Also, users can discover other places using Cucom to see the comments and voting results.

However, only uses the NFC to leave comments are not convenient. To enhance the availability, we also keep the search engine in the app. User can search for any CTag through the internet, but they are not allowed to give like and dislike on the CTag.



Cucom includes three main parts: Server, Android App and CTag. Server is mainly used for query and storing the CTag information. Android App interacts between CTag and Server. CTag is used for storing the information of the location.

5.1 Server

The server is used to manipulate the information related to CTag. In order to manipulate all the data efficiently and effectively, we chose to use a server as our database. Since the server is easier for us to synchronize between mobile devices, so it gives the users the update-to-date information. It also provides a large memory storage space as the memory of mobile phone is comparatively small. Besides, the server ensures reliability. If any CTag is damaged, malfunctioned or lost, it can be replaced by a new one and retrieves the information from server.

The server divides into two parts, database and interface. The database is used for storing the information of CTag, comments, and statistic of voting. Behind this we needs an interface to handles the action on the database, client side can gives query and update the database through the interface.

5.2 Android App

The android app has four modules, NFC, Search, Comment, and Vote.

The CTags will be located in different places that people will visit, for example, restaurant and lecture theater etc. After the users installed Cucom, the users need to enable the NFC in their mobile setting. Then, they can scan the CTag whenever they want to see the description of that place. The app will be opened automatically when detecting the CTag and read the information inside the CTag. It will shows the place information that the CTag representing. The information includes name, description, statistics on likes and dislikes, and comments. The users can have a voting on it. There are two choices, like and dislike. After choosing it, users just need to scan the CTag again and the app will updates the CTag statistic.

Also, they can leave the comments comment of the place represent by CTag to share the opinions with other users. Each of the comment includes the content and the time of comment that created. The app will show the newest comments.

The locations of CTag are classified into several categories. There are three main categories: Distinct, Facility and Faculty. The users can use the app to search the places according to these categories or simply type in the keyword they are interested in. After that, the app will show the list of relevant results. The list includes the photo, name and numbers of likes and dislikes of each result.

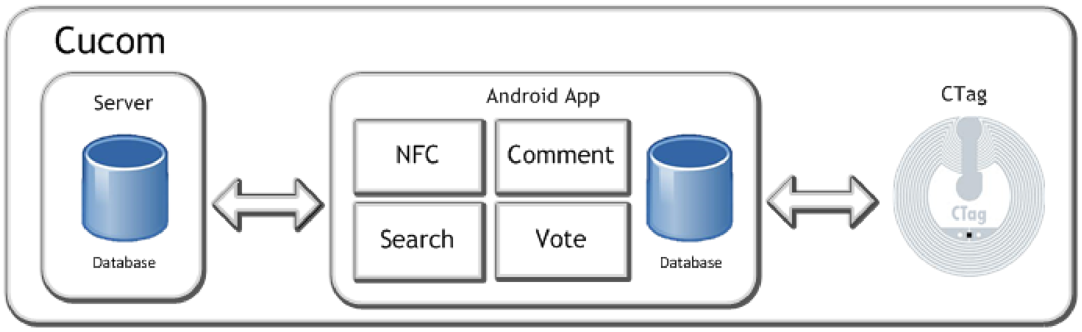
The users can choose the result to view the voting statistics and the other user comments. They can also leave comments. But they cannot vote it unless they go to the location of the CTag. The reason is that we encourage the users to pay more attention to the physical world. The users need to go out and find the CTag location for vote. The vote will be reflect their true feeling as they are really been that place. Also, it avoids fake statistics of the vote. Users cannot create fake account to vote as most of us will only have one mobile phone.

5.3 CTag

CTag is an NFC Tag that used by Cucom. CTag likes a small database, which can be read and updated by the users. Each CTag represents a physical environment, so the users can retrieve the information efficiently without complex searching steps. Also, the mobile phone can retrieve the information without using network, as long as the NFC is on. Initially, CTag stores the information about the place and the statistics of likes and dislikes are zero. When the users use NFC to scan it, the app can retrieve the information and show it to the users. If the users vote it, Cucom will update the CTag so that the statistics of likes and dislikes will be accumulated.

6 Cucom Design

As mentioned above, Cucom includes three parts: Server, Android App and CTag. Server is mainly used for query and storing the CTag information. Android App is used to interact between CTag and Server. CTag is used for storing the information of the location.



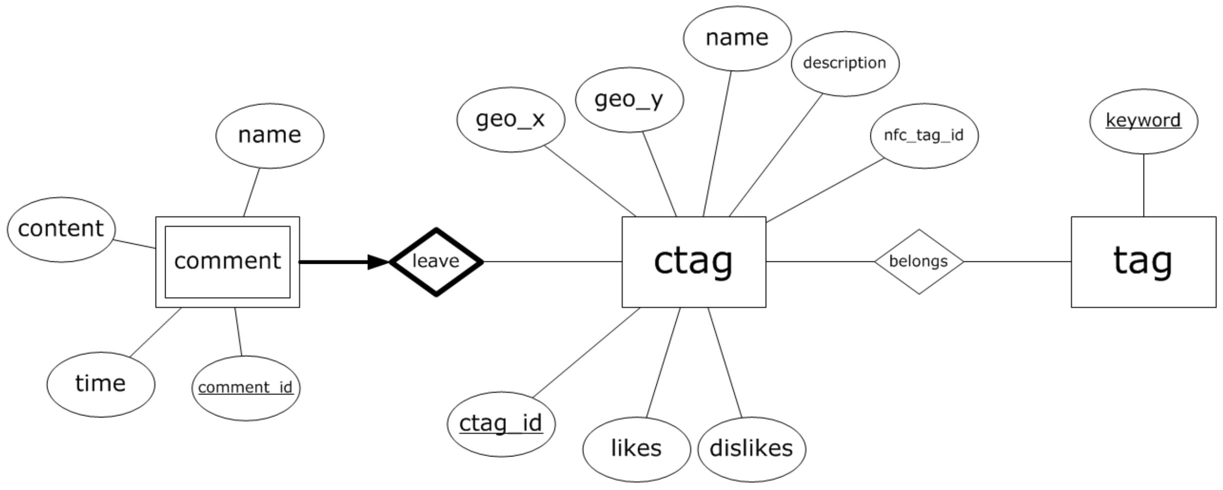
6.1 Server

The server divides into two parts, database and interface. The database is used for storage while interface us used for query the database according to the request of the app.

6.1.1 Database

Cucom shows the information about the location, comments details and the voting results to the users. The information about the place includes name, description, categories and geometric location. The comment includes the time it is created. The voting results are the number of likes and dislikes. To make the information easier to retrieve and modify, we normalized these attributes and assigned into three tables, ctag, comment and tag.

6.1.2 ER Diagram



6.1.3 Schema

comment(ctag_id: Integer, comment_id: String, name: String, content: String, time: timestamp)
ctag(ctag_id: Integer, nfc_tag_id: String, name: String, description: String, geo_x: String, geo_y: String, likes: Integer, dislikes: Integer)
tag(keyword: String, ctag_id: Integer)

The ctag table stores the description of place and statistic of vote. comment stores the time and content. The tag table stores the classification information of CTag. We created the CTag ID to be uniquely identified and be the linkage among these tables. The reason why we are not using the NFC tag UID is that if the CTag is damaged, malfunctioned or lost, we can replace it with a new NFC tag. So, we can simply change the NFC tag ID and the origin structures (including comments and tagging system) are still remain unchanged.

The ctag table

The ctag table includes nfc_tag_id, ctag_id, name, description, geo_x, geo_y, likes and dislikes. The nfc_tag_id is the NFC tag unique id. The manufacturer will assign a unique id for each NFC tag and it cannot be changed. It is used for the validation when editing the parameters of CTag in server. The ctag_id is the primary key. The name and description are the information about the place represent by CTag. The geo_x and geo_y are the longitude and latitude for the place respectively.

They are used for showing the location in Google map in further development. Each of the CTag will have only one entry in this table.

Attribute	Description	Data Type
nfc_tag_id	The NFC tag unique id	String
ctag_id	The CTag id use by Cucom	Integer
name	The name for the place	String
description	The description for the place	String
geo_x	The longitude for the place For future development	Double
geo_y	The latitude for the place For future development	Double
likes	The number of like	Integer
dislikes	The number of dislike	Integer

The comment table

The comment table includes comment_id, ctag_id, name, content and time. The comment_id is the primary key. The name is the user name of Cucom for future development. The content is the content of the comment. The time is the time that the comment is uploaded to server. Each CTag can have more than one comment.

Attribute	Description	Data Type
comment_id	The unique comment id	String
ctag_id	The CTag id use by Cucom	Integer
name	The user name who give this comment For future development	String
content	The content of the comment	String
time	The time that the comment uploaded	timestamp

The tag table

The tag table includes ctag_id and keyword. The keyword is the categories for the CTag.

Attribute	Description	Data Type
ctag_id	The CTag id use by Cucom	Integer
keyword	The categories of the CTag	String

CTag have several sub-categories which belong to three main categories: Distinct, Facility and Faculty. In Distinct, we use the nine colleges and Main Campus to classify, i.e. Chung Chi College, New Asia College, Shaw College, United College, S.H. Ho College, Lee Woo Sing College, Morningside College, Wu Yee Sun College, C.W. Chu College and Main Campus. In Facility, we classified into Library, Lectures Theatres, Restaurant, View and Hall. In Faculty, we classified into Art, Business Administration, Education, Engineering, Law, Medicine, Science, and Social. Each CTag can have more than one keyword.

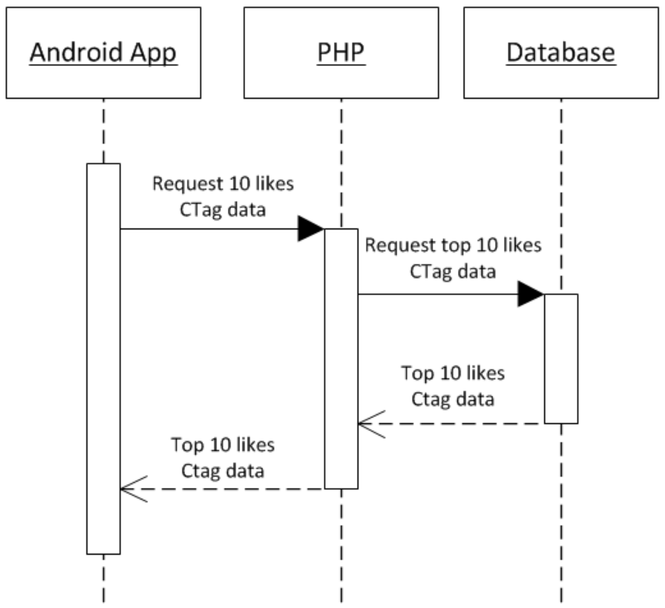
Distinct	Facility	Faculty
• Chung Chi College	• Library	• Art
• New Asia College	• Lectures Theatres	• Business Administration
• Shaw College	• Restaurant	• Education
• United College	• View	• Engineering
• S.H. Ho College	• Hall	• Law
• Lee Woo Sing College		• Medicine
• Morningside College		• Science
• Wu Yee Sun College		• Social
• C.W. Chu College		
• Main Campus		

6.1.4 Interface

The interface is used to query the database and returns the relevant data from database to app. The interface involves five activities: get rank, search CTag, get CTag comment, submit CTag comment, and update votes.

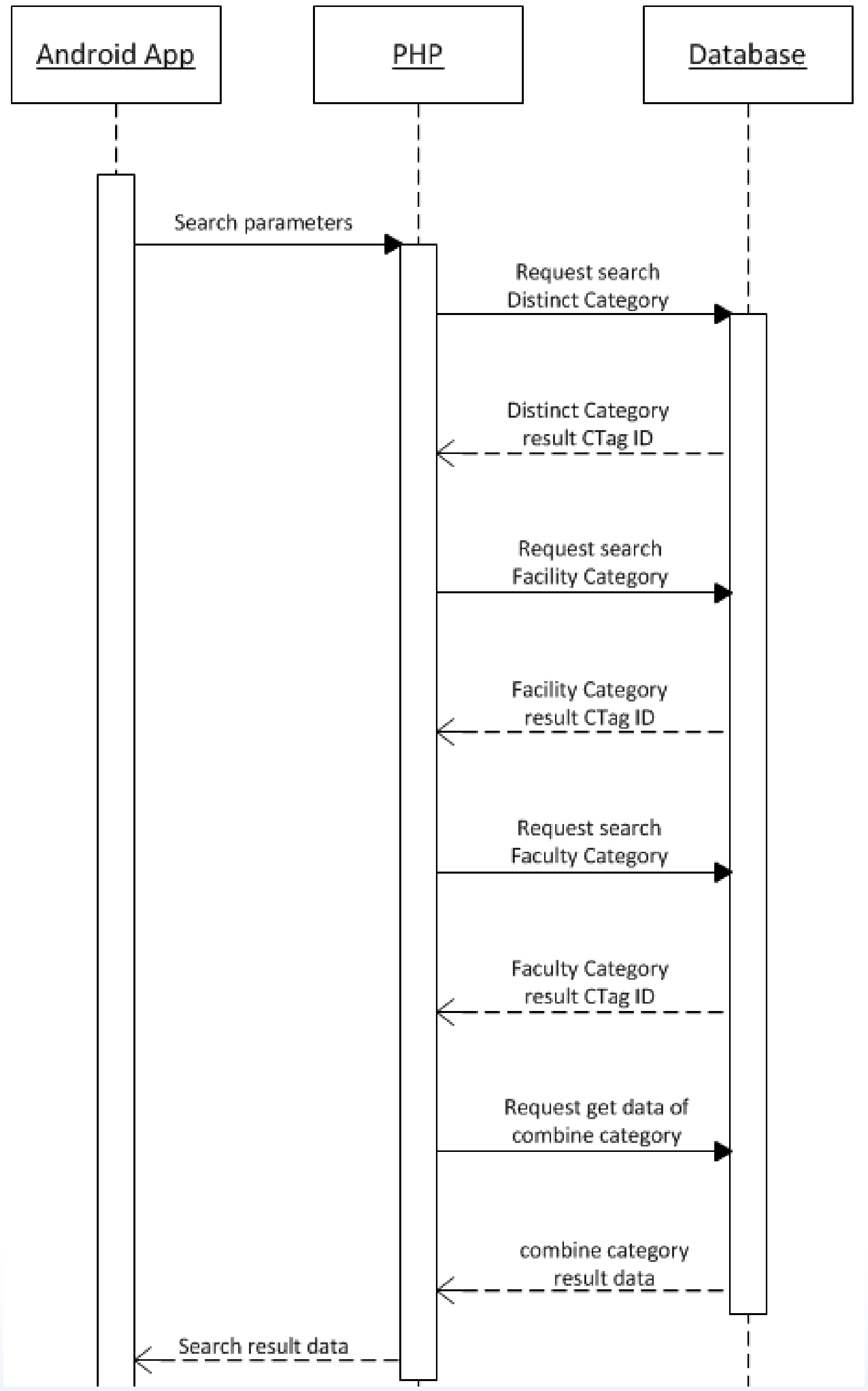
Get rank

The home page of Cucom will show the top ten likes CTag to the users. The app will send a request to get top ten likes of CTag data to PHP. Then, the PHP will query the database to retrieve top ten CTag data including name, description, likes, and dislikes. If it is success, PHP will return the results to the app.



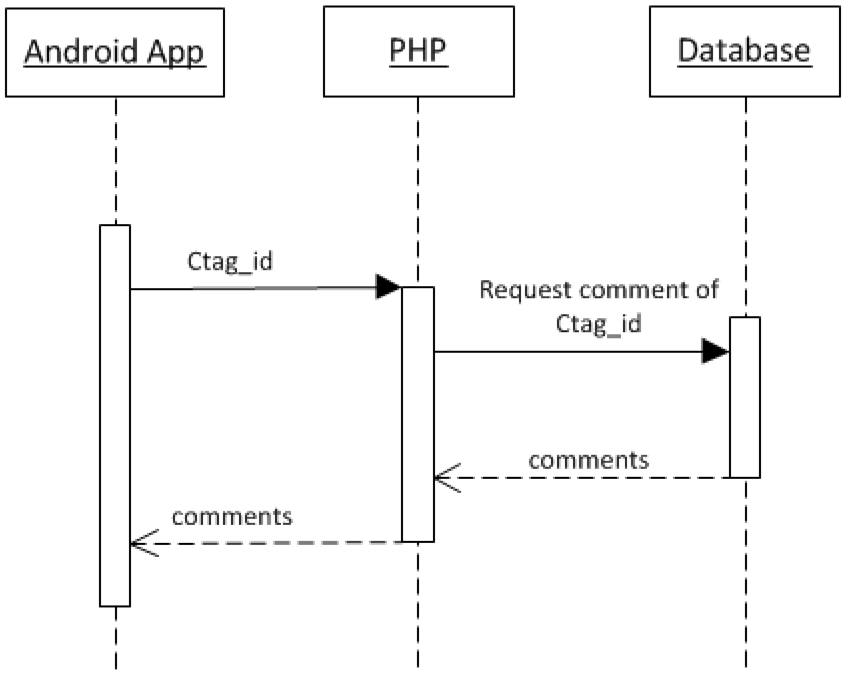
Search CTag

When users enter the search word or select the categories in searching, android app will send the search parameters to the PHP. The search parameters involve categories and keywords. After PHP received, it will request server to return the CTag IDs that belong to the categories chosen by the users. Then, it will combine these results and request server to return the name, description, likes, and dislikes of the CTags that fulfilled with the keywords using the combined result.


Android App
PHP
Database
Search parameters
Request search Distinct Category
Distinct Category result CTag ID
Request search Facility Category
Facility Category result CTag ID
Request search Faculty Category
Faculty Category result CTag ID
Request get data of combine category
combine category result data
Search result data

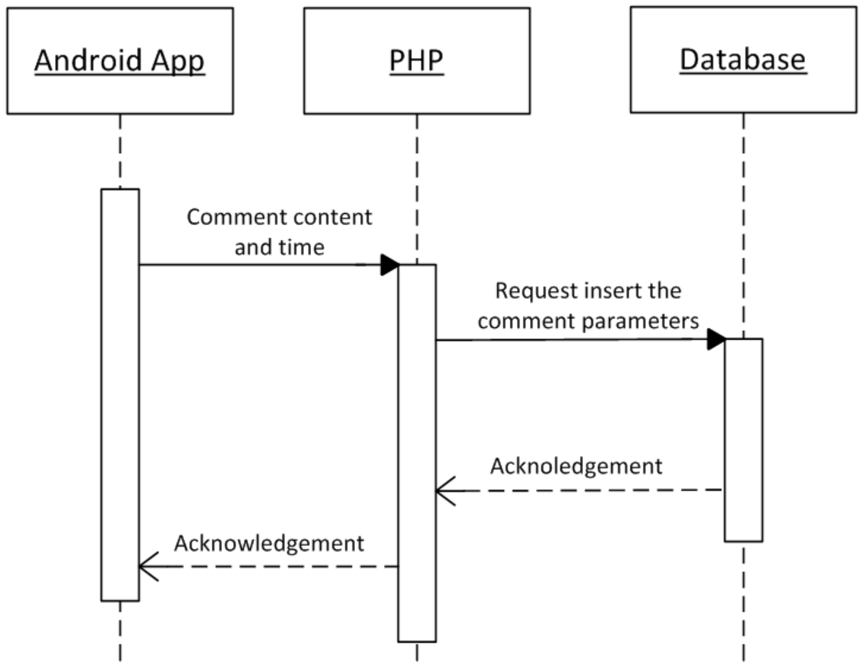
Get CTag comment

To get a particular CTag comment, the app will send the CTag ID to the PHP. The PHP will query the database for getting the content and time of comments. After it retrieved the comments successfully, it will return the data to the app.



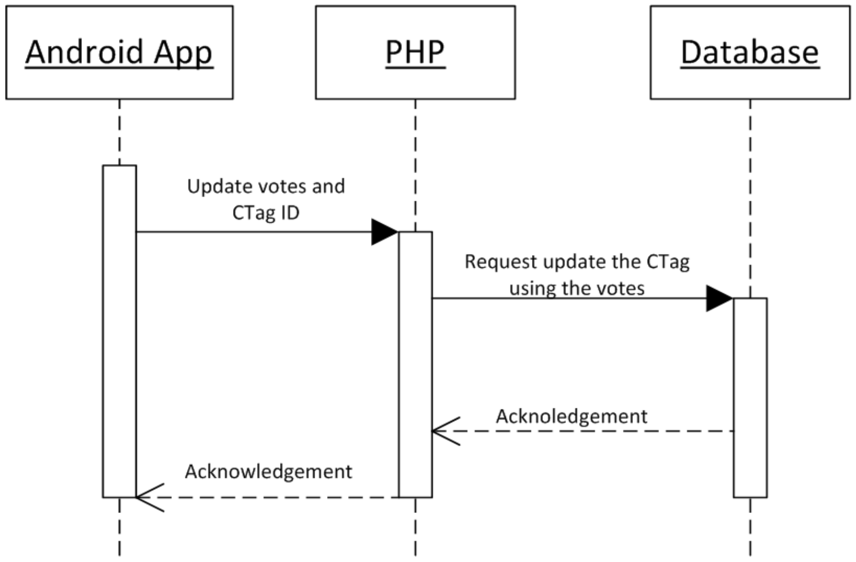
Submit CTag comment

When the users comment with a CTag, the app will send the comment content and CTag ID to the PHP. The PHP will request inserting the comment content into database. The database will generate a time for the comment. Then, PHP will return an acknowledgement to app if it is success.



Update Votes

When the users vote on the CTag with network, the app will send the vote to the PHP. PHP will request the database to update the vote using the CTag ID. Then, it will send an acknowledgement to the app.

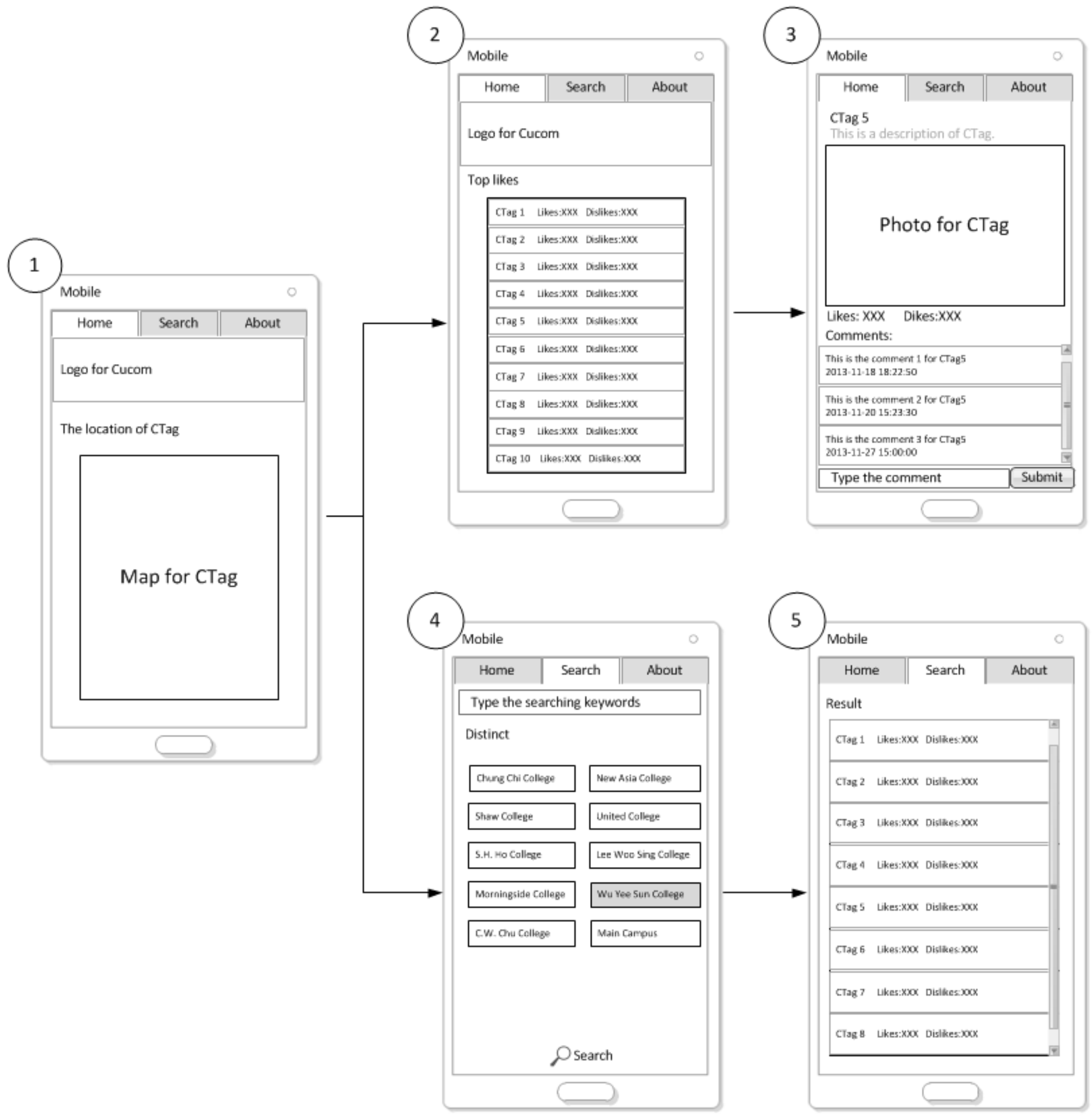


6.2 Android App

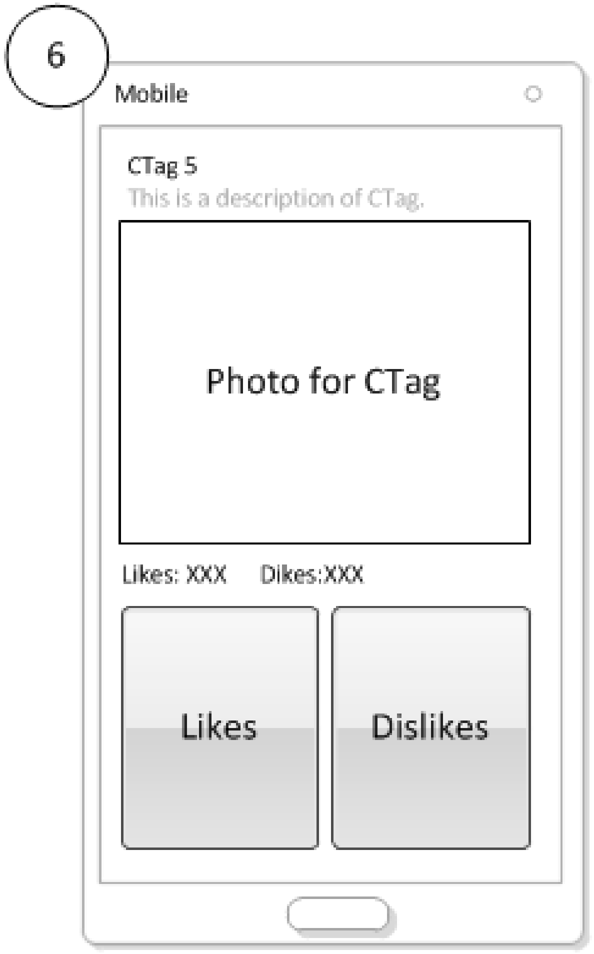
The android app provides the user interface to show the information to the users. Also, it will listen to the user request and respond it. The app will perform corresponding activities by interacting with the CTag and the server. The following will describe the User Interface and the Modules.

6.2.1 User Interface

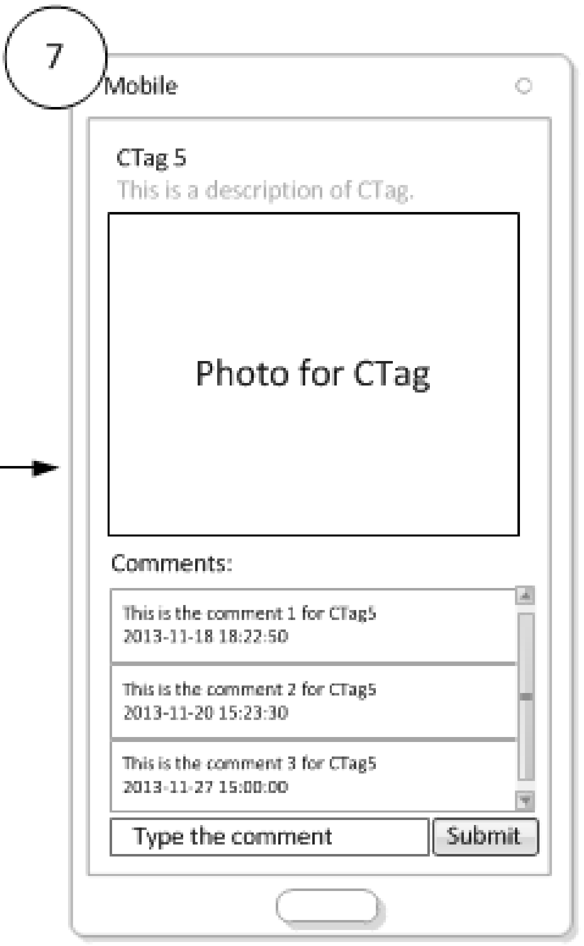
The user interface is one of the most concerns in our project. A good user interface can attract the users to make use of the application. We think that a good interface not only about the fancy layout, but also the user experience. Thus, our interface mainly focuses on user experience. The following is the wireframes that show the basic user interface design of Cucom.


1
Mobile
Home
Search
About
Logo for Cucom
The location of CTag
Map for CTag
2
Mobile
Home
Search
About
Logo for Cucom
Top likes
CTag 1 Likes:XXX Dislikes:XXX
CTag 2 Likes:XXX Dislikes:XXX
CTag 3 Likes:XXX Dislikes:XXX
CTag 4 Likes:XXX Dislikes:XXX
CTag 5 Likes:XXX Dislikes:XXX
CTag 6 Likes:XXX Dislikes:XXX
CTag 7 Likes:XXX Dislikes:XXX
CTag 8 Likes:XXX Dislikes:XXX
CTag 9 Likes:XXX Dislikes:XXX
CTag 10 Likes:XXX Dislikes:XXX
3
Mobile
Home
Search
About
CTag 5
This is a description of CTag.
Photo for CTag
Likes: XXX Dikes:XXX
Comments:
This is the comment 1 for CTag5 2013-11-18 18:22:50
This is the comment 2 for CTag5 2013-11-20 15:23:30
This is the comment 3 for CTag5 2013-11-27 15:00:00
Type the comment
Submit
4
Mobile
Home
Search
About
Type the searching keywords
Distinct
Chung Chi College
New Asia College
Shaw College
United College
S.H. Ho College
Lee Woo Sing College
Morningside College
Wu Yee Sun College
C.W. Chu College
Main Campus
Search
5
Mobile
Home
Search
About
Result
CTag 1 Likes:XXX Dislikes:XXX
CTag 2 Likes:XXX Dislikes:XXX
CTag 3 Likes:XXX Dislikes:XXX
CTag 4 Likes:XXX Dislikes:XXX
CTag 5 Likes:XXX Dislikes:XXX
CTag 6 Likes:XXX Dislikes:XXX
CTag 7 Likes:XXX Dislikes:XXX
CTag 8 Likes:XXX Dislikes:XXX

1. Home page
2. Top 10 likes page
3. Description of CTag page
4. Searching page
5. Search result page



6. Vote page (NFC scanning)



This is the comment 1 for CTag5
2013-11-18 18:22:50

This is the comment 2 for CTag5
2013-11-20 15:23:30

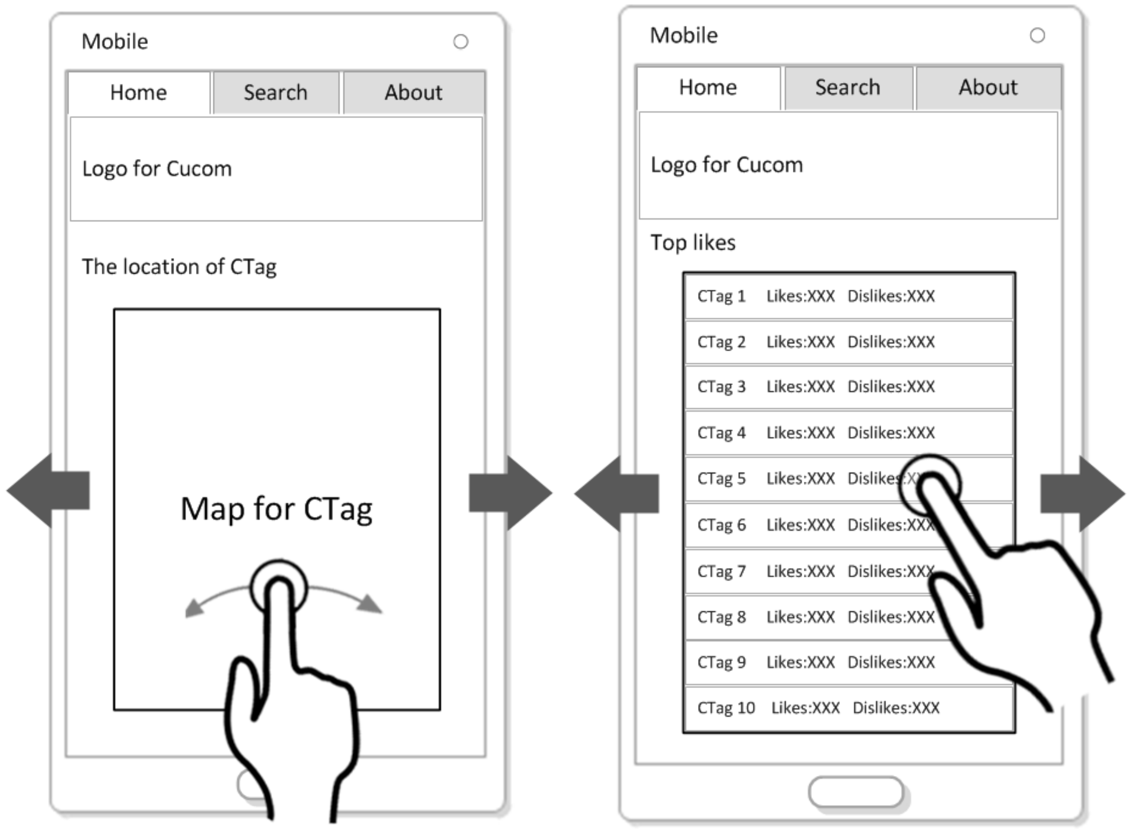
This is the comment 3 for CTag5
2013-11-27 15:00:00

7. Comment page (NFC scanning)

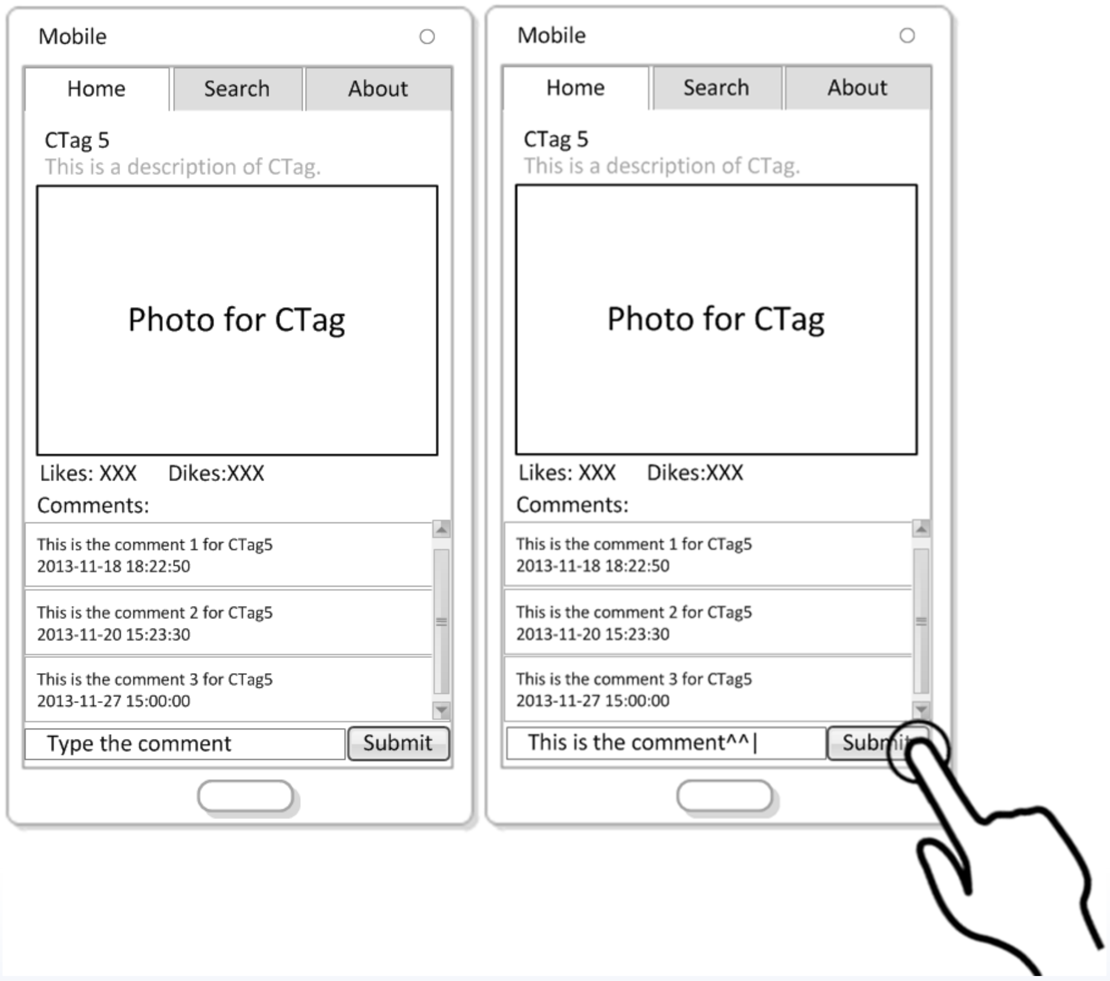
There are two ways to intend the app: launch the app by tapping the icon and scan the CTag using NFC. The app will show the 1 - 5 layouts when the users open the app. 6 and 7 are the layouts for users using NFC. All of the layouts are designed to be as simple as possible so that users feel easier to use our app. Besides, we use the tab bar in the top of the app for a clear view. There are three tabs: Home, Search and About. Also, we applied the swipe gesture to the app. It is similar to the android OS theme, so that users are familiar with it.

The layout 1 is the home page of the app, it shows the location of CTag and top 10 likes ranking list in layout 2. The users can view more details about the CTag by touching it. It will show the details in layout 3. For searching CTag, users can use the layout 4 to choose the categories and type in the keyword that they want to search. The search result will show in layout 5 that similar to the list in top 10 likes. Same as top 10 likes, users can view more by touching the result and it will show the detail in layout 3. The layout 6 will be show when the users scan the CTag. Users can vote on it and leave the comment in layout 7. Note that layout 3 is different to layout 7 that layout 7 does not show the number of likes and dislikes. The numbers are shown in layout 6. We will explain the layout one by one.

Home Tab

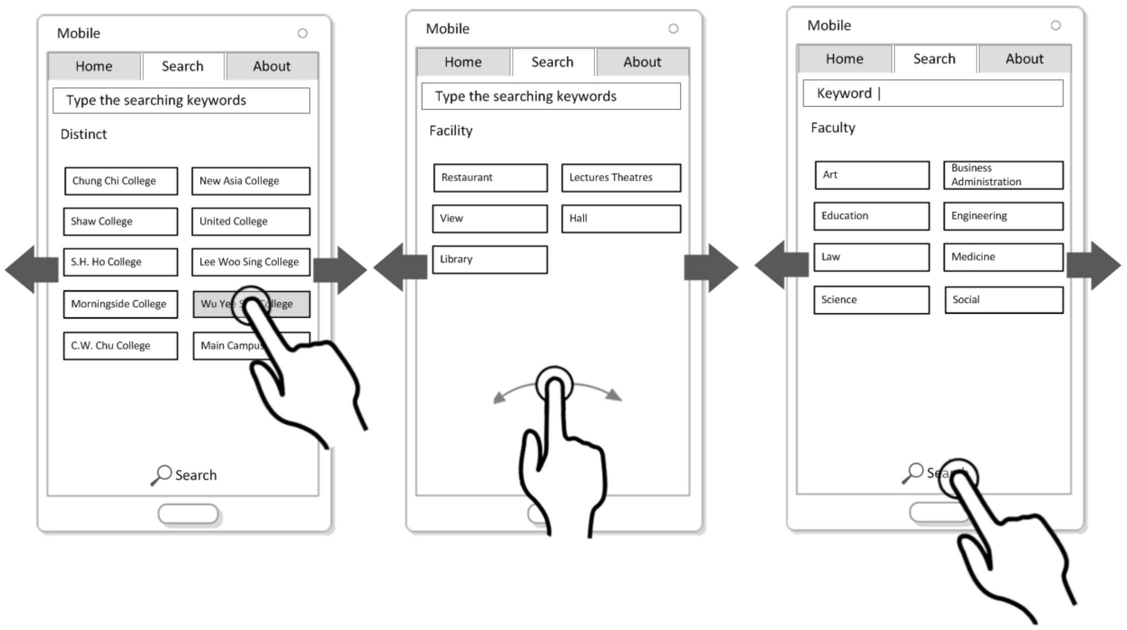

Mobile
Home
Search
About
Logo for Cucom
The location of CTag
Map for CTag
Mobile
Home
Search
About
Logo for Cucom
Top likes
CTag 1 Likes:XXX Dislikes:XXX
CTag 2 Likes:XXX Dislikes:XXX
CTag 3 Likes:XXX Dislikes:XXX
CTag 4 Likes:XXX Dislikes:XXX
CTag 5 Likes:XXX Dislikes:XXX
CTag 6 Likes:XXX Dislikes:XXX
CTag 7 Likes:XXX Dislikes:XXX
CTag 8 Likes:XXX Dislikes:XXX
CTag 9 Likes:XXX Dislikes:XXX
CTag 10 Likes:XXX Dislikes:XXX

When the users open the app, it will show the home tab first. In the top of the tab will be the logo or banner for Cucom. In bottom, it has a container that contains two subpages: the location of CTag and Top likes. Users can use the swipe gesture to shift between pages. The location of CTag will show in the Google map so that users can find the CTag for voting. The top likes will show the 10 CTag records ordered by the number of likes using a list. Each row will show the photo, likes and dislikes of the CTag. The reason why we are using a list to present is that it gives a lucid overview to the user. If the users are interested in the CTag, they can simply touch the result and it will show the details.



When the users touch for details, it will change to description page. On the top of the page will show the name and description of the CTag. In the center will be the photo of the CTag. Below the photo will show the number of likes and dislikes and the comments. We have use a list with a scroll bar to present some of the comment. Each row show the content of comment and the time of created. The newest comments will be show in the below and user can view the previous comments by scrolling up the list. At the bottom, it provides a textbox for users to type in their opinion. After that, they can touch the submit button to submit the comment. If it success, the comment list will be updated.

Search Tab

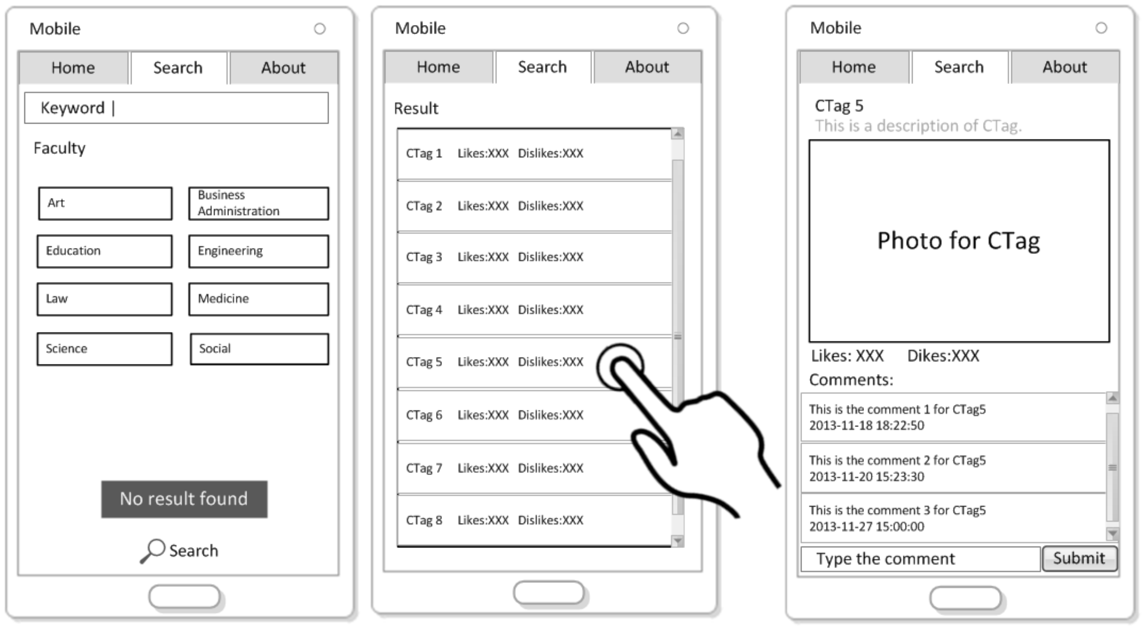


At the top of the search tab, it provides a textbox for users to type in the keyword for the searching CTag. The center of the tab is a container that contains three subpages: Distinct, Facility and Faculty. Similar to the home tab, the container can detect the swipe gesture and shift between pages.

In each subpage, there are several categories. The users can touch the categories in order to select it. The categories will be highlighted when it is selected. Users can select more than one choice. They can select other categories in the subpages.

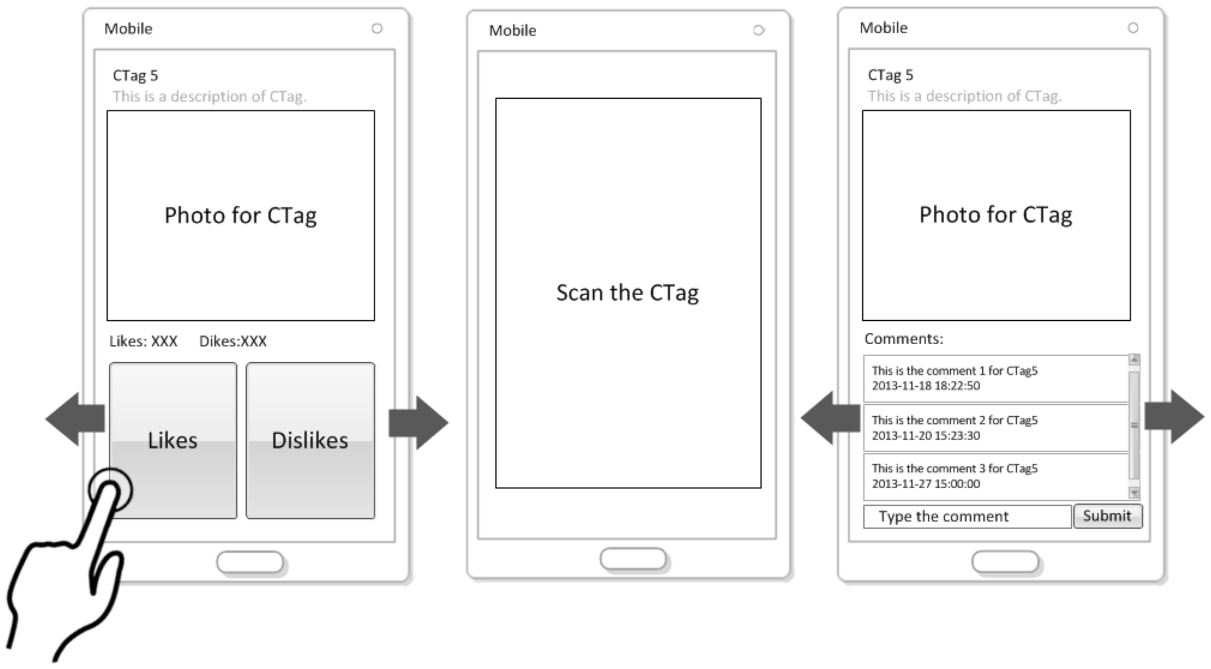
The logical relationship of the selected categories in the same subpage is OR. For selected categories in the different subpages, the logical relationship between them is AND. For example, if the user ‘United College’ and ‘Main Campus’ in Distinct, ‘Lectures Theatres in Facility’, and ‘Art’ in the Faculty, the result will be the art lectures theatres at United College or Main Campus. For the keyword, the logical relationship with the selected categories is AND. For example, if the users selected ‘Restaurant’ and type in ‘Coffee’, the result will be the CTag under ‘Restaurant’ and the name or description include ‘Coffee’.

After that, the users can touch the search button to start searching the CTag.


Mobile
Home
Search
About
Keyword |
Faculty
Art
Business Administration
Education
Engineering
Law
Medicine
Science
Social
No result found
Search
Mobile
Home
Search
About
Result
CTag 1 Likes:XXX Dislikes:XXX
CTag 2 Likes:XXX Dislikes:XXX
CTag 3 Likes:XXX Dislikes:XXX
CTag 4 Likes:XXX Dislikes:XXX
CTag 5 Likes:XXX Dislikes:XXX
CTag 6 Likes:XXX Dislikes:XXX
CTag 7 Likes:XXX Dislikes:XXX
CTag 8 Likes:XXX Dislikes:XXX
Mobile
Home
Search
About
CTag 5
This is a description of CTag.
Photo for CTag
Likes: XXX Dikes:XXX
Comments:
This is the comment 1 for CTag5 2013-11-18 18:22:50
This is the comment 2 for CTag5 2013-11-20 15:23:30
This is the comment 3 for CTag5 2013-11-27 15:00:00
Type the comment
Submit

If no result found, the app will pop up a 'No result found' toast message to notice users. So, they can start a new searching. If the search success, the search results will show in a list that just like to the top ten likes. When the users touch the result, it will show the description of the CTag.

Scanning CTag

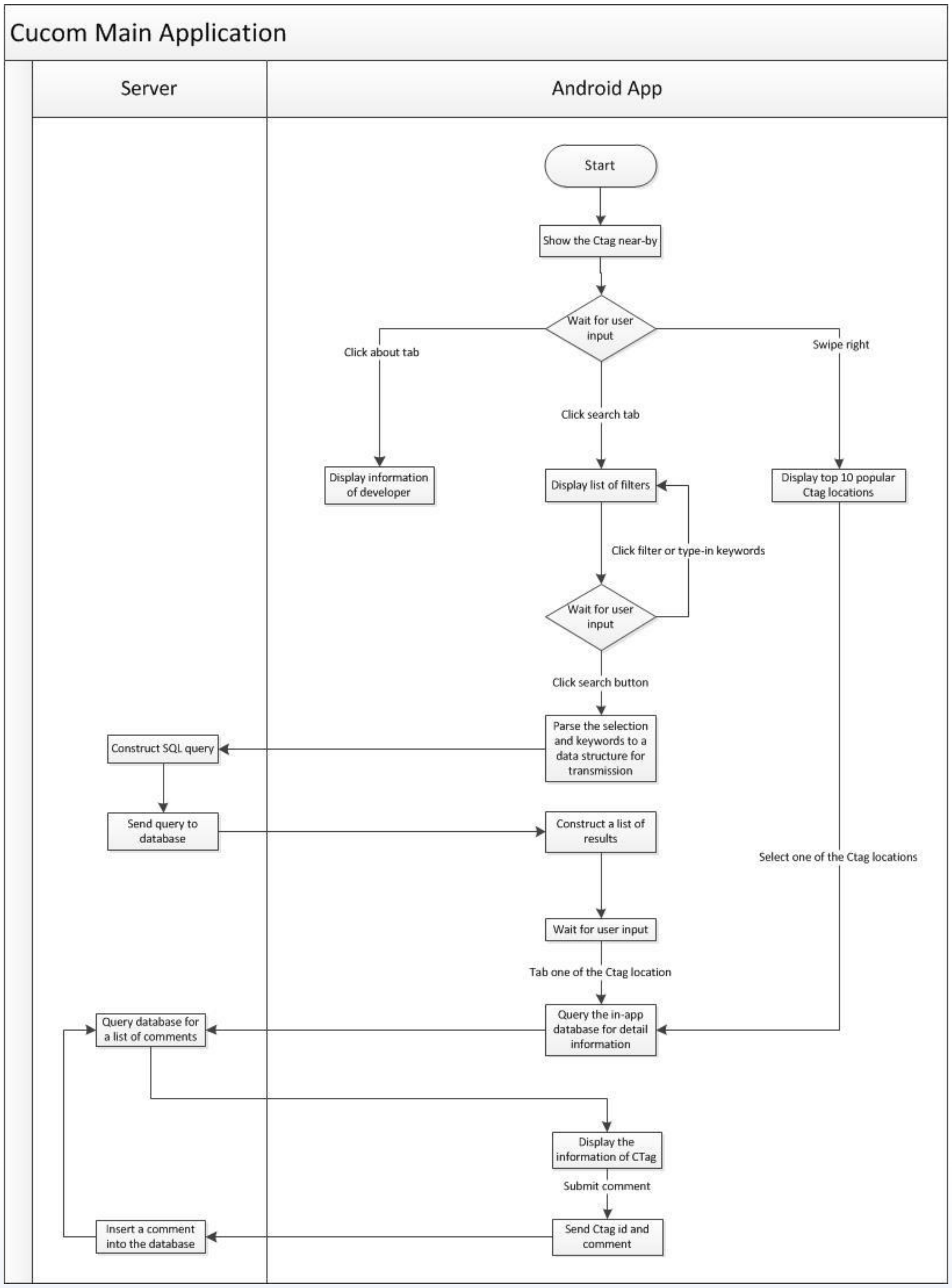


The layout of description page for scanning CTag is different compare with that previous page that users open the app. The top of the page is same as previous description page. But there is a container that contains two subpages: vote and comment. Users can swipe to shift the pages. The previous page does not include the vote page. The vote subpage involves two buttons: Likes and Dislikes. Users can choose one of it for voting. The app will instruct users to scan the CTag to vote. If it is success, the app will return to the vote subpages and update the number of likes and dislikes.

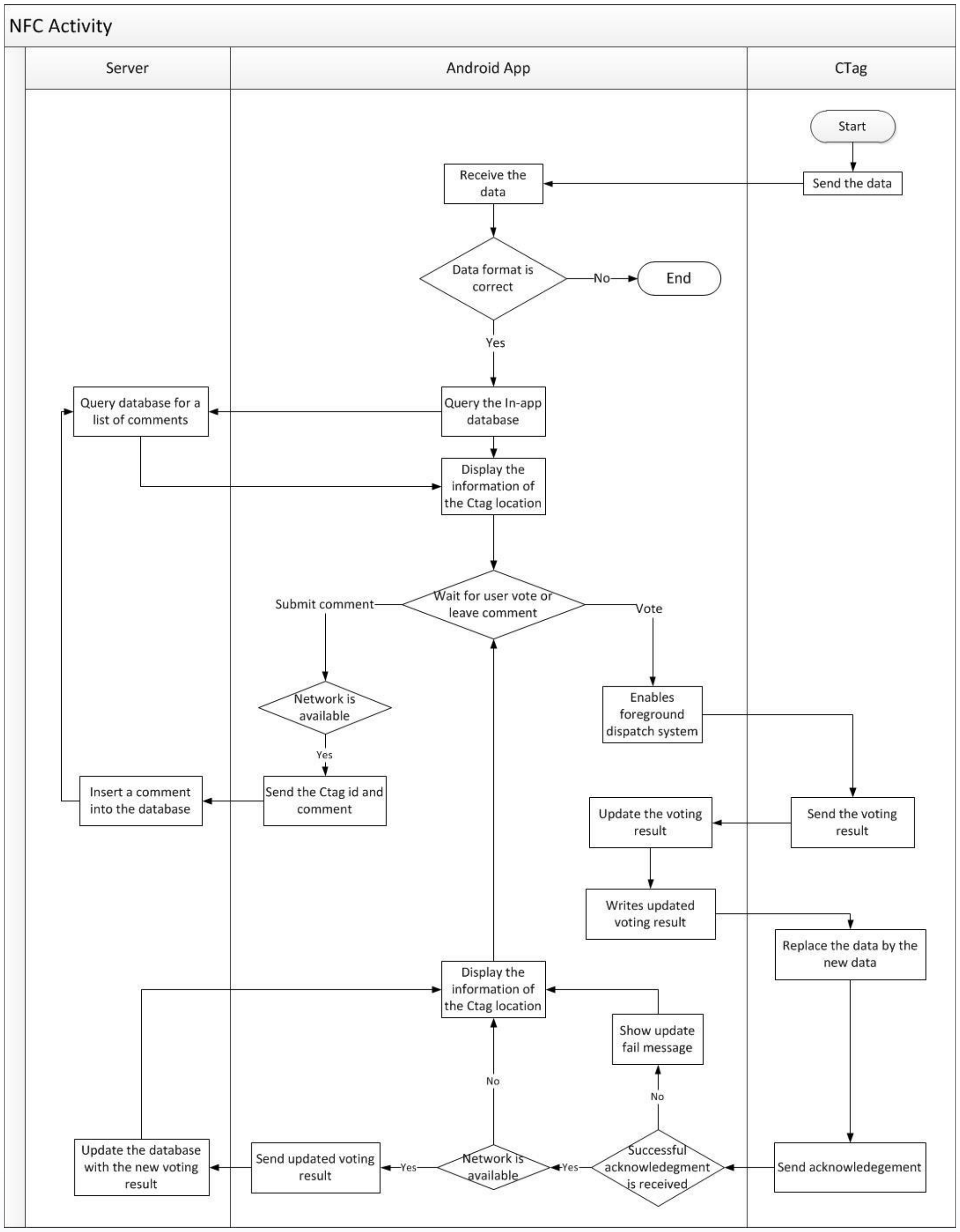
6.2.2 Modules

Cucom is divided into two parts, Main application and NFC activity. Main application uses internet to access CTag location information while NFC activity can only be intent by CTag using NFC.

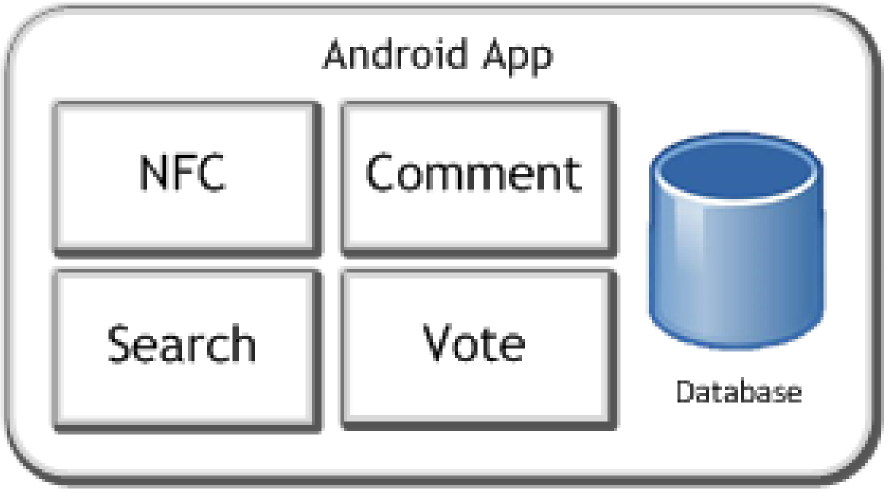
Main application allows user to do searching and commenting. In the home activity, user is free to browse. When user swipes the center of the screen, the fragment will display a list of locations representing the top popularities. If user clicks on the search tab, fragment will be replaced by a set of filter options as well as a search bar. Swiping the screen shows more filter options. User can select multiple filters and type in the keyword. After user tab the search button, application will query the database in the server and then display a list of search results. User can click one of them to see the detail of the CTag location. At this stage, user can even submit a comment to the database. The activity flow is shown in the following diagram.


Cucom Main Application
Server
Android App
Start
Show the Ctag near-by
Wait for user input
Click about tab
Swipe right
Click search tab
Display information of developer
Display list of filters
Display top 10 popular Ctag locations
Click filter or type-in keywords
Wait for user input
Click search button
Construct SQL query
Parse the selection and keywords to a data structure for transmission
Send query to database
Construct a list of results
Select one of the Ctag locations
Wait for user input
Tab one of the Ctag location
Query database for a list of comments
Query the in-app database for detail information
Display the information of CTag
Submit comment
Insert a comment into the database
Send Ctag id and comment

On the other hand, NFC activity is launched by intent filter by Android. At the beginning, the NFC tag is attached in the intent. The NFC activity receives the intent and reads information from the tag. After validation, the activity loads the database in the application for name, description and image file. Then it will get the comments from the Cucom webserver if network connection is available. After all information is prepared, the activity will display it on to the screen. User can either vote the location or writes comment to it. If user votes the location, the NFC activity intent to a new activity which handles the foreground dispatch system. User has to put his mobile phone back to the same CTag. When an NFC tag is found, the activity will validate the CTag to see whether it is the same tag or not. After validation, the activity will read again the data inside the tag, and then update the vote result back to the CTag. This can ensure the atomicity of the voting operation. User cannot holds the data of a CTag for a long time then write it back to the CTag. The activity flow is shown below.


NFC Activity
Server
Android App
CTag
Start
Send the data
Receive the data
Data format is correct
No
End
Yes
Query the In-app database
Query database for a list of comments
Display the information of the Ctag location
Wait for user vote or leave comment
Submit comment
Vote
Network is available
Yes
Send the Ctag id and comment
Insert a comment into the database
Enables foreground dispatch system
Send the voting result
Update the voting result
Writes updated voting result
Replace the data by the new data
Display the information of the Ctag location
Show update fail message
No
No
Send acknowledgement
Successful acknowledgement is received
Yes
Network is available
Yes
Send updated voting result
Update the database with the new voting result

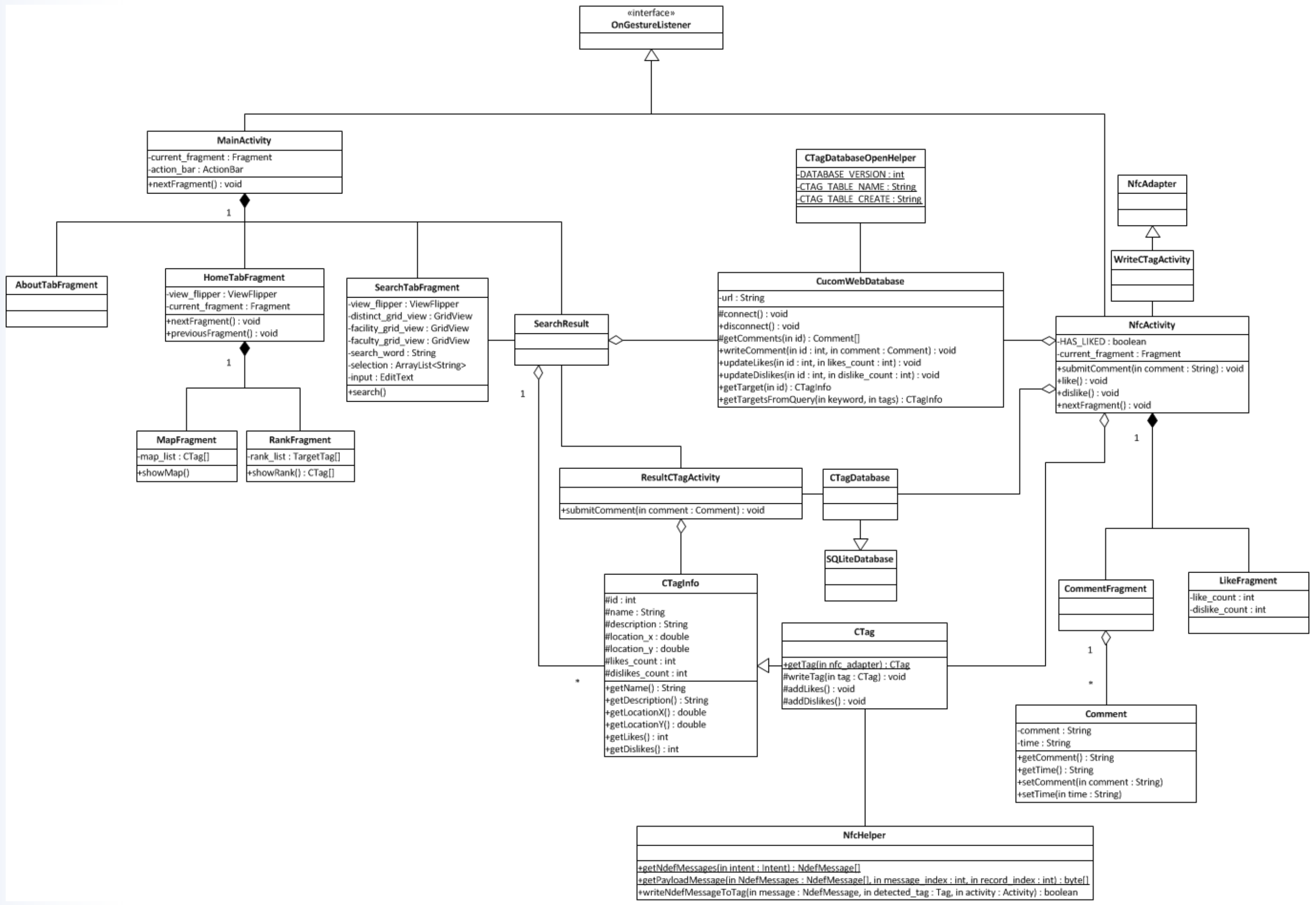
With these two parts Cucom can be divided into four modules: Search engine, Voting system, Commenting System and NFC dispatch system.



Search engine allows user to search the information stored in CTag, including name of location, photo, description, voting results and comments. According to our research, we have implemented a user friendly classification in the search engine. Voting system is the mechanism to like and dislike on the CTag, this can only be done when the user scans the CTag. Commenting System allows user to share their feeling and comment to each other, user can do it via the internet or on the CTag. NFC dispatch system connects the user to the reality.

This system allows user to scan CTags and manipulate the information on the CTag. The system is divided into two parts; NFC tag intent dispatch system and NFC foreground dispatch system. Intent dispatch system allows Android to launch Cucom whenever a CTag is scanned. Foreground dispatch system helps writing data back to the CTag.

The relationship between objects is shown as follows

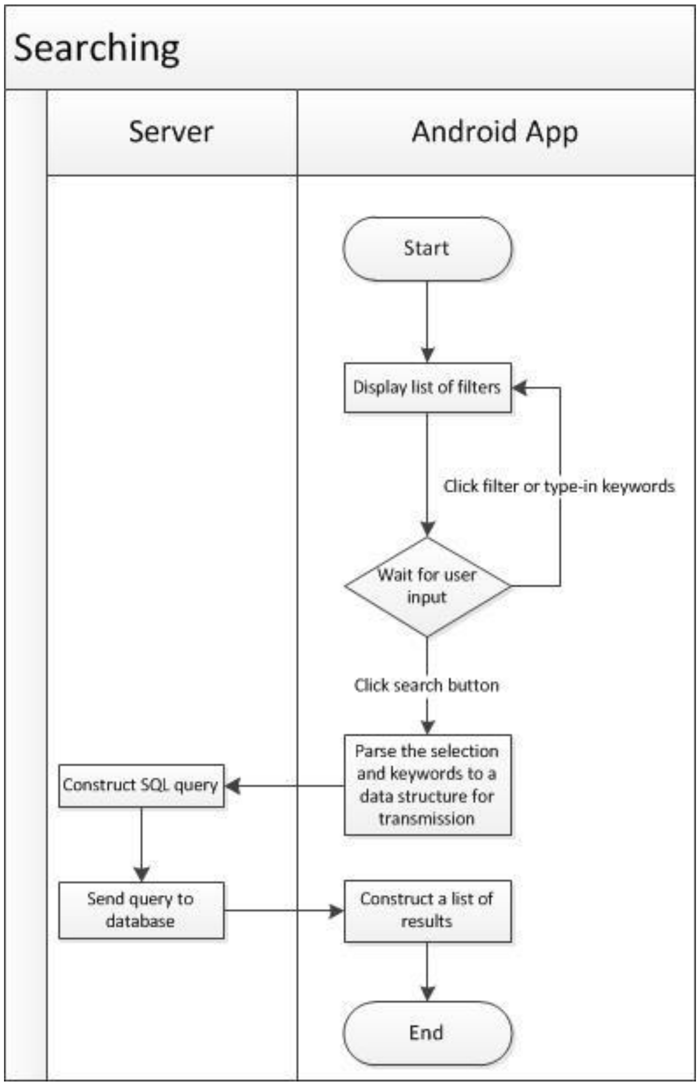

«interface»
OnGestureListener
MainActivity
-current_fragment : Fragment
-action_bar : ActionBar
+nextFragment() : void
1
AboutTabFragment
HomeTabFragment
-view_flipper : ViewFlipper
-current_fragment : Fragment
+nextFragment() : void
+previousFragment() : void
1
MapFragment
-map_list : CTag[]
+showMap()
RankFragment
-rank_list : TargetTag[]
+showRank() : CTag[]
SearchTabFragment
-view_flipper : ViewFlipper
-distinct_grid_view : GridView
-facility_grid_view : GridView
-faculty_grid_view : GridView
-search_word : String
-selection : ArrayList<String>
-input : EditText
+search()
SearchResult
1
CTagDatabaseOpenHelper
-DATABASE_VERSION : int
-CTAG_TABLE_NAME : String
-CTAG_TABLE_CREATE : String
CucomWebDatabase
-url : String
#connect() : void
+disconnect() : void
#getComments(in id) : Comment[]
+writeComment(in id : int, in comment : Comment) : void
+updateLikes(in id : int, in likes_count : int) : void
+updateDislikes(in id : int, in dislike_count : int) : void
+getTarget(in id) : CTagInfo
+getTargetsFromQuery(in keyword, in tags) : CTagInfo
NfcAdapter
WriteCTagActivity
NfcActivity
-HAS_LIKED : boolean
-current_fragment : Fragment
+submitComment(in comment : String) : void
+like() : void
+dislike() : void
+nextFragment() : void
1
ResultCTagActivity
+submitComment(in comment : Comment) : void
CTagDatabase
SQLiteDatabase
CTagInfo
#id : int
#name : String
#description : String
#location_x : double
#location_y : double
#likes_count : int
#dislikes_count : int
+getName() : String
+getDescription() : String
+getLocationX() : double
+getLocationY() : double
+getLikes() : int
+getDislikes() : int
*
CTag
+getTag(in nfc_adapter) : CTag
#writeTag(in tag : CTag) : void
#addLikes() : void
#addDislikes() : void
CommentFragment
1
*
LikeFragment
-like_count : int
-dislike_count : int
Comment
-comment : String
-time : String
+getComment() : String
+getTime() : String
+setComment(in comment : String)
+setTime(in time : String)
NfcHelper
+getNdefMessages(in intent : Intent) : NdefMessage[]
+getPayloadMessage(in NdefMessages : NdefMessage[], in message_index : int, in record_index : int) : byte[]
+writeNdefMessageToTag(in message : NdefMessage, in detected_tag : Tag, in activity : Activity) : boolean

Search Engine

The search engine in Cucom uses the server database to do the query. Input keywords will be used to match the name and description of the location. We also gives tags to all the locations, hence user is free to choose the filter of the location. Inspiring by OpenRice, we uses the filter technique to filter out the search result. Besides, in our research, we found that tagging system suits our usage.

Workflow

The workflow in this module is very strict forward, combine the keywords and filters, and then send it to the webserver. After retrieval, list of results can be displayed on the screen.



We have divided the tags into three categories; i.e. Faculty, District and Facility. User can select multiple filters on each category. We can define the searching mechanism in the following way. For category s and a tag t in s, let the tag be

(s,t). If the tag (s,t) is selected, then $i_{(s,t)} = 1$ otherwise $i_{(s,t)} = 0$. Hence, the search result is defined as the following:

$$\{exists\ a,b,c\ (i_{(1,a)} > 0\ and\ i_{(2,b)} > 0\ and\ i_{(3,c)} > 0\}$$

$$P$$

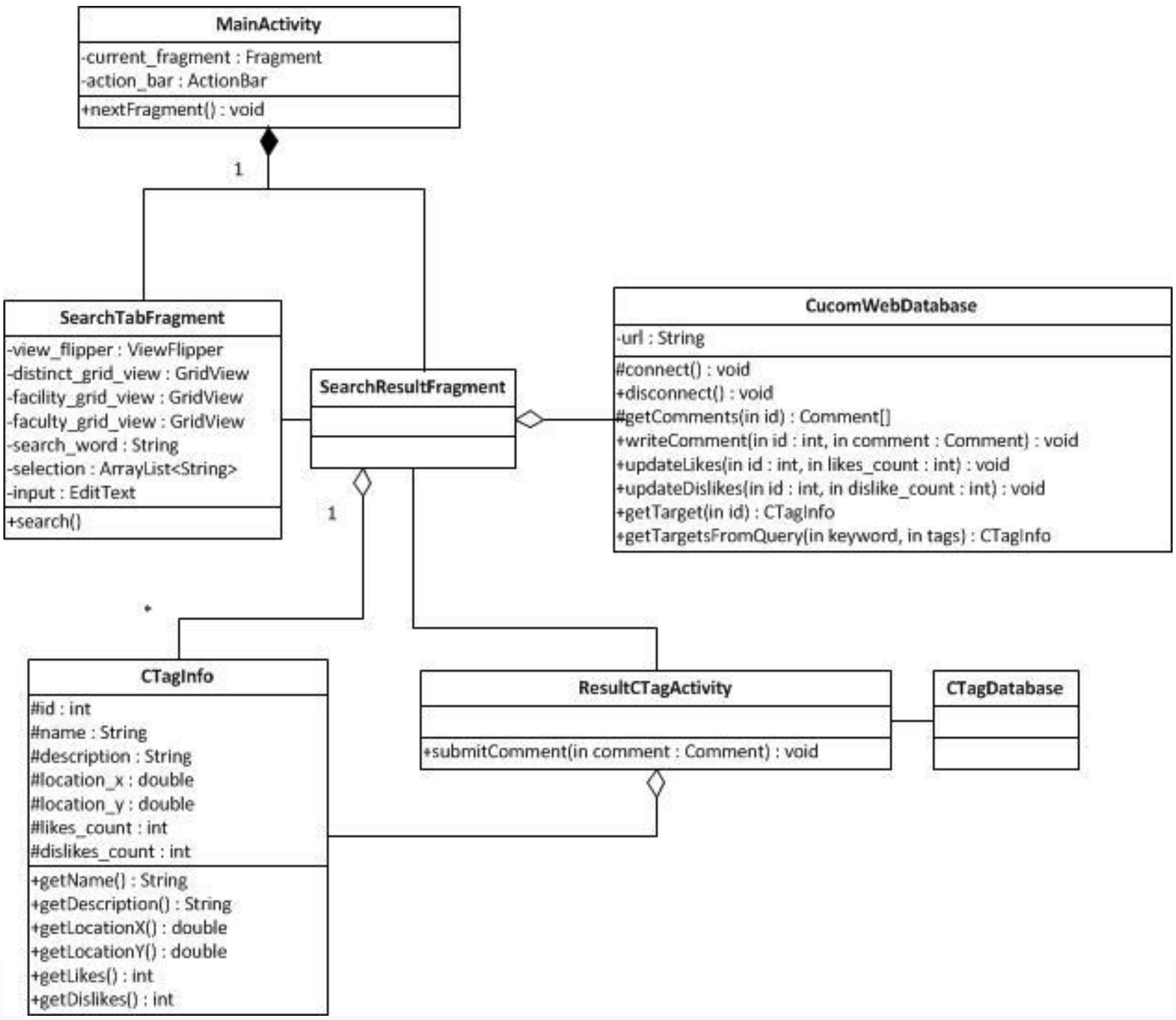
$$\{for\ all\ s,t\ \big((s,t)\ is\ tag\ of\ o\big), i_{(s,t)} > 0)\}$$

To improve the user experience, when no tag is selected, we assume that all the tag is being selected. For each category, we have to produce a list of result with all the tags. Then pick out the results which exist in all three lists and put it together into a new list. Using this list, filter out all the results that do not contain the keywords. Result can be obtained, by the following algorithm.

```
FUNCTION search(faculty_tags, district_tags, facility_tags, keyword)
      faculty_results ← queryDatabase(faculty_tags)
      district_results ← queryDatabase(district_tags)
      facility_results ← queryDatabase(facility_tags)

      filtered_results ← extractCommonResults(  faculty_results,
                                                district_results,
                                                facility_results)

      final_results ← findKeywords(filtered_results, keyword)
END
```

Methods

The classes in this module is shown below


MainActivity
-current_fragment : Fragment
-action_bar : ActionBar
+nextFragment() : void
1
SearchTabFragment
-view_flipper : ViewFlipper
-distinct_grid_view : GridView
-facility_grid_view : GridView
-faculty_grid_view : GridView
-search_word : String
-selection : ArrayList<String>
-input : EditText
+search()
SearchResultFragment
CucomWebDatabase
-url : String
#connect() : void
+disconnect() : void
#getComments(in id) : Comment[]
+writeComment(in id : int, in comment : Comment) : void
+updateLikes(in id : int, in likes_count : int) : void
+updateDislikes(in id : int, in dislike_count : int) : void
+getTarget(in id) : CTagInfo
+getTargetsFromQuery(in keyword, in tags) : CTagInfo
1
*
CTagInfo
#id : int
#name : String
#description : String
#location_x : double
#location_y : double
#likes_count : int
#dislikes_count : int
+getName() : String
+getDescription() : String
+getLocationX() : double
+getLocationY() : double
+getLikes() : int
+getDislikes() : int
ResultCTagActivity
+submitComment(in comment : Comment) : void
CTagDatabase

SearchTabFragment

This is the main fragment of this module it allows user to type in keywords and adding filters. It contains three grid view objects; district grid view, facility grid view and faculty grid view. Each grid view shows a set of tags buttons allowing user to select. Multiple selections are allowed.

Search()

This method will be invoked when the search button is pressed. It uses the CucomWebDatabase to give query. Detail design has been discussed in Workflow(P.120).

SearchResultFragment

This fragment lists out all the results after searching. User can pick one of the results by tabbing on it, the ResultCTagActivity will then be intent.

ResultCTagActivity

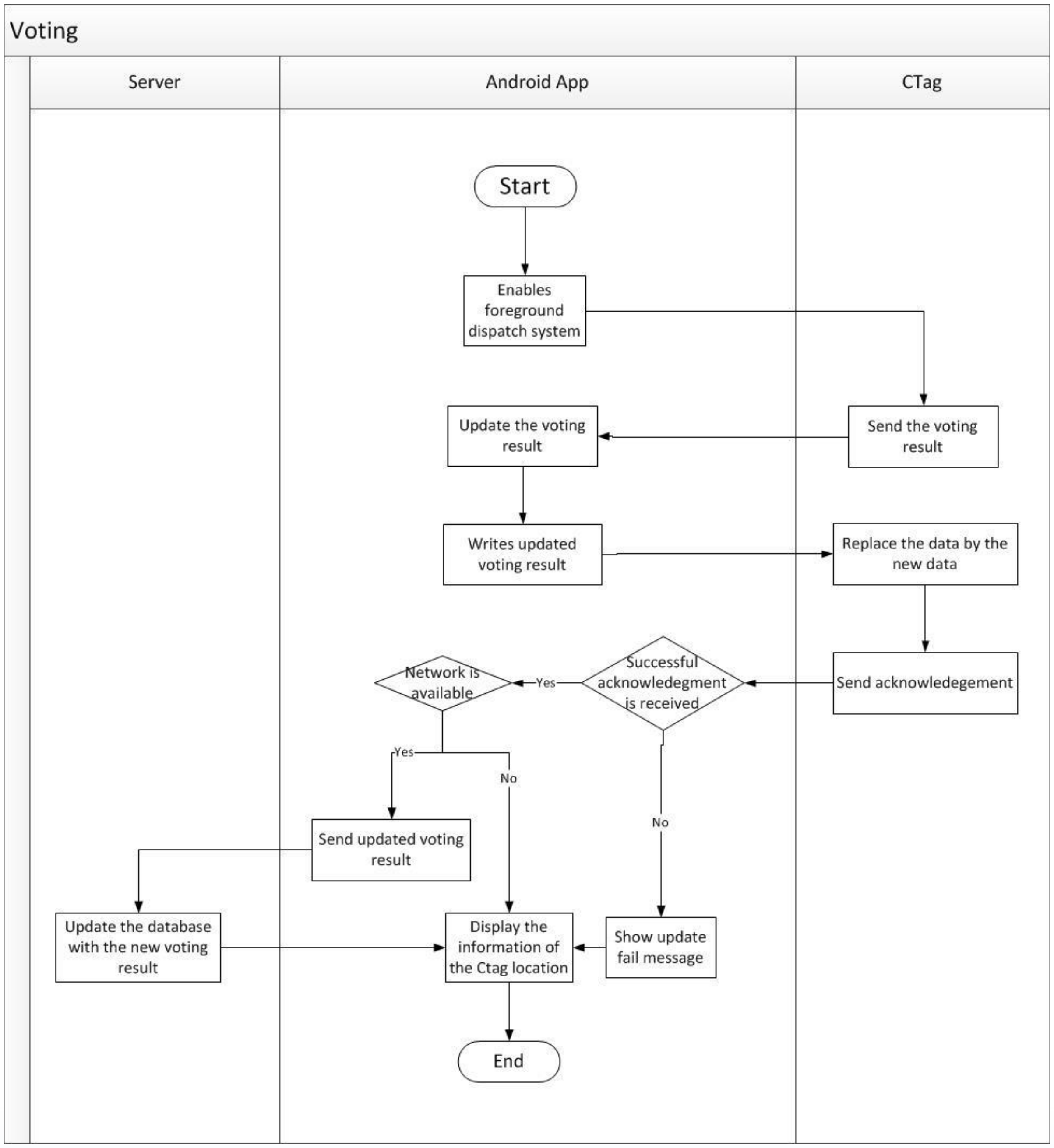
This activity displays all the information regarding to the CTag location. Since only a CTag ID is received, the activity needs to use the in-app database to retrieve location information such as image, name and description. It also allows user to submit comment.

Voting System

When the mobile phone touches a CTag, user can votes on that corresponding location. The voting operation can be done in offline mode. Hence all users, including for those do not have internet access, can use this service.

Workflow

At the beginning, the activity reads the votes in the CTag which can be done in offline. Then user can votes the location on his choice. When a like or dislike is being pressed, user has to put the phone on to the same CTag to write the updated results back on the CTag. In the meantime, the activity will check the network status. If internet is available, the updated results will upload to the webserver. If the internet is not available, the activity simply neglects the upload operation.

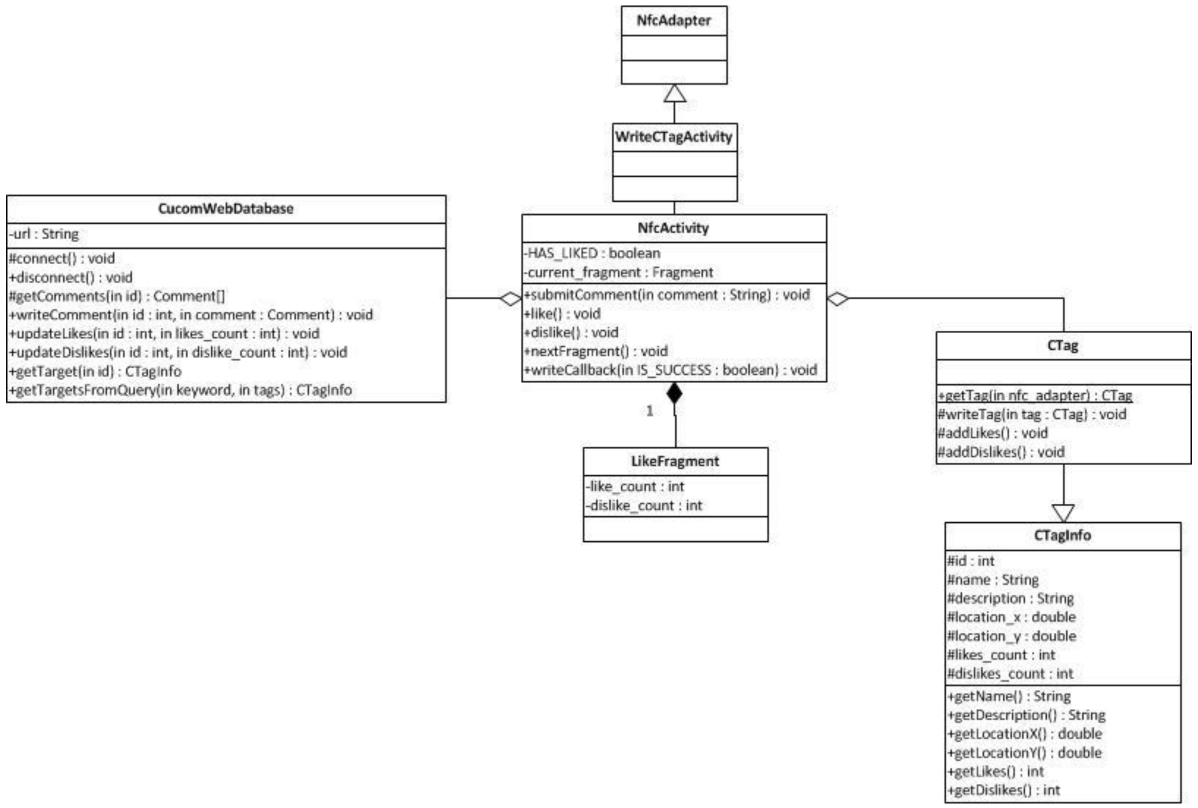

Voting
Server
Android App
CTag
Start
Enables foreground dispatch system
Send the voting result
Update the voting result
Writes updated voting result
Replace the data by the new data
Send acknowledegement
Successful acknowledegment is received
Yes
Network is available
Yes
No
No
Send updated voting result
Update the database with the new voting result
Display the information of the Ctag location
Show update fail message
End

Since the upload process is done after the NFC writing process. Before uploading to the webserver, we have to ensure the data is successfully written to the CTag. Hence, we have to add a callback function to monitor the dispatch result.

```
FUNCTION like(original_value)
    value  ← original value + 1
    intent ForegroundDispatchSystem(value, uploadCallback)
END

FUNCTION uploadCallBack(original_value, IS_SUCESS)
    if IS_SUCCESS
        updateServerDatabase(original_value + 1)
        lockLikeButtons()
END
```

We have to keep the original value fixed, because if we increment the original value every time user press like, then it may give a leak hole that user can click the like button several times, then like as many time as possible. Notice that when the data is successfully uploaded to the CTag, the like and dislike buttons will be locked. So the user cannot keep liking the CTag.

Methods



When a CTag is scanned, NfcActivity will be intent. It displays two buttons allowing user to like and dislike.

NfcActivity

This is the main activity on this side. It contains a locking key HAS_LIKED, when this Boolean value becomes true meaning that like/dislike button has been pressed, then user is not allowed to press them anymore. It also stores the current fragment to monitor the fragment flow.

like()

When the button "like" is pressed, this method will be invoked. It will prepare an NDEF message then send intent to the WriteCTagActivity.

Dislike()

Similar to that of *like()*

nextFragment()

When a swipe operation is listened, the current fragment will change to the next one.

writeCallback(IS_SUCCESS)

This is a callback method, when the data is successfully written to the CTag, a true value will be passed to this method. This method will update the webserver and disable the like and dislike buttons.

CTag

This class represents the CTag object, it will be returned when a CTag is scanned

addLikes()

This method will increment the like value in the CTag.

addDisikes()

This method will increment the dislike value in the CTag.

LikeFragment

This is the fragment that shows two buttons with like counts and dislike counts. The controller of this view is in NfcActivity.

CucomWebDatabase

This class represents the data base on the webserver. All the uploading method is included in here.

updateLikes(id, like_count)

In the *writeCallback()* method, when write operation is success, the like count has to be uploaded to the webserver by invoking this method. This method simply sends the CTag id and updated voting results to the webserver.

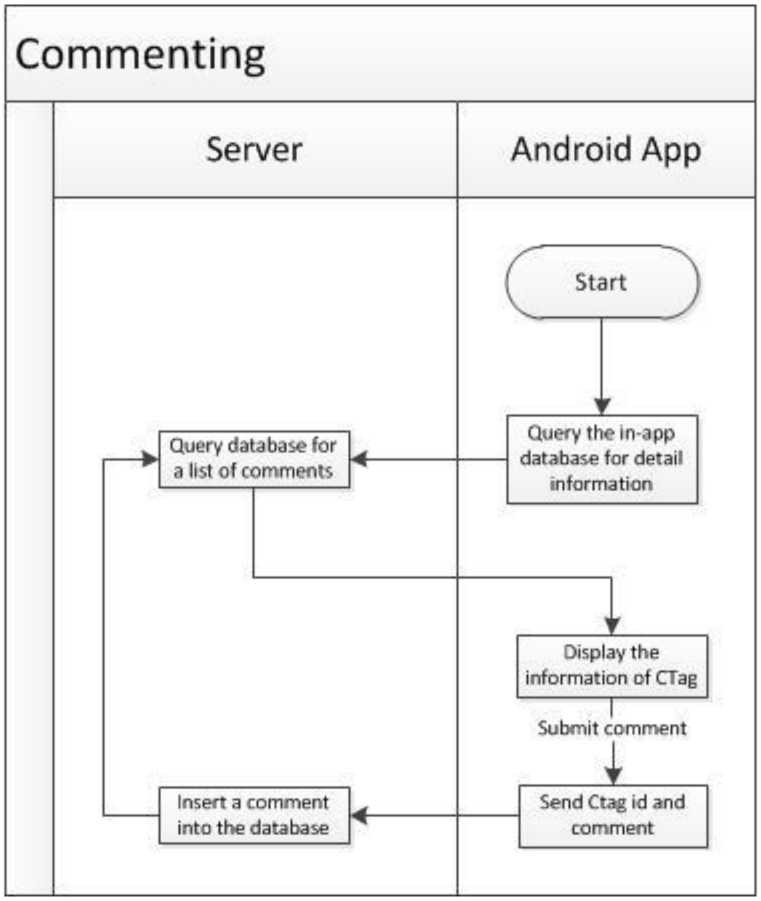
Commenting System

In both main application and NFC activity, user can comments the location whenever network is available. For each comment, it has its own timestamp for sorting and record. In database, it also contains a name attribute, for development of accounting system.

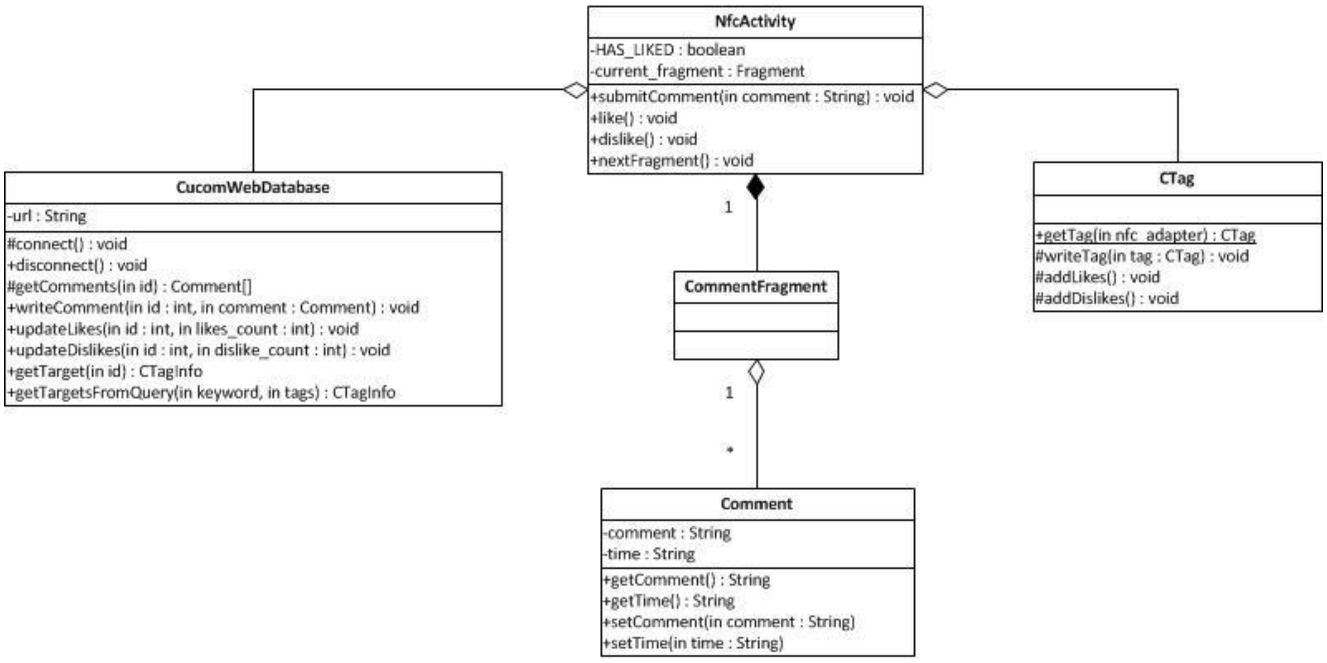
Workflow

After location information is ready and displayed on the screen, user can view a list of comments in descending order. This can be done by requesting comment list from the webserver. When user submit a comment, the application will send the comment to the webserver, the database will automatically generate a timestamp for the creation time.

The activity diagram is given below


Commenting
Server
Android App
Start
Query the in-app database for detail information
Query database for a list of comments
Display the information of CTag
Submit comment
Send Ctag id and comment
Insert a comment into the database

Methods


NfcActivity
-HAS_LIKED : boolean
-current_fragment : Fragment
+submitComment(in comment : String) : void
+like() : void
+dislike() : void
+nextFragment() : void
CucomWebDatabase
-url : String
#connect() : void
+disconnect() : void
#getComments(in id) : Comment[]
+writeComment(in id : int, in comment : Comment) : void
+updateLikes(in id : int, in likes_count : int) : void
+updateDislikes(in id : int, in dislike_count : int) : void
+getTarget(in id) : CTagInfo
+getTargetsFromQuery(in keyword, in tags) : CTagInfo
CTag
+getTag(in nfc_adapter) : CTag
#writeTag(in tag : CTag) : void
#addLikes() : void
#addDislikes() : void
1
CommentFragment
1
*
Comment
-comment : String
-time : String
+getComment() : String
+getTime() : String
+setComment(in comment : String)
+setTime(in time : String)

In the initialization of NfcActivity, it gets comments using the CucomWebDatabase and updates the CommentFragment. On the other hand, comment can be submitted via a CucomWebDatabase object. NFCActivity listen to the submit button, whenever the button is pressed, it send the CTag ID and comment to the CucomWebDatabase.

NfcActivity

The view of this activity contains a UIButton that used to submit the comment. When the button is pressed, *submitComment(comment)* will be called

submitComment(comment)

This method reads the input comment from the edit text. Then send the comment string and CTag ID to the CucomWebDatabase.

CommentFragment

This fragment contains a listview that reads an array list of commands from the server. In the initialization stage of NfcActivity, it will get all the related comments from the CucomWebDatabase, then list it on the listview in this fragment.

Comment

This class represent the logical comment in the CTag. Each instance of Comment represents an individual comment with a unique timestamp.

getComment()

return the comment content

getTime()

return the comment creation time

setComment(comment)

save the comment into instance attribute

setTime(time)

save the time into instance attribute

CucomWebDatabase

This object represents the database in the webserver. Operations like reading and writing comments from/to the internet uses this class to function.

getComments(ID)

Providing the CTag ID, it request the webserver to send a list of comments corresponding to this CTag

writeComments(ID, comment)

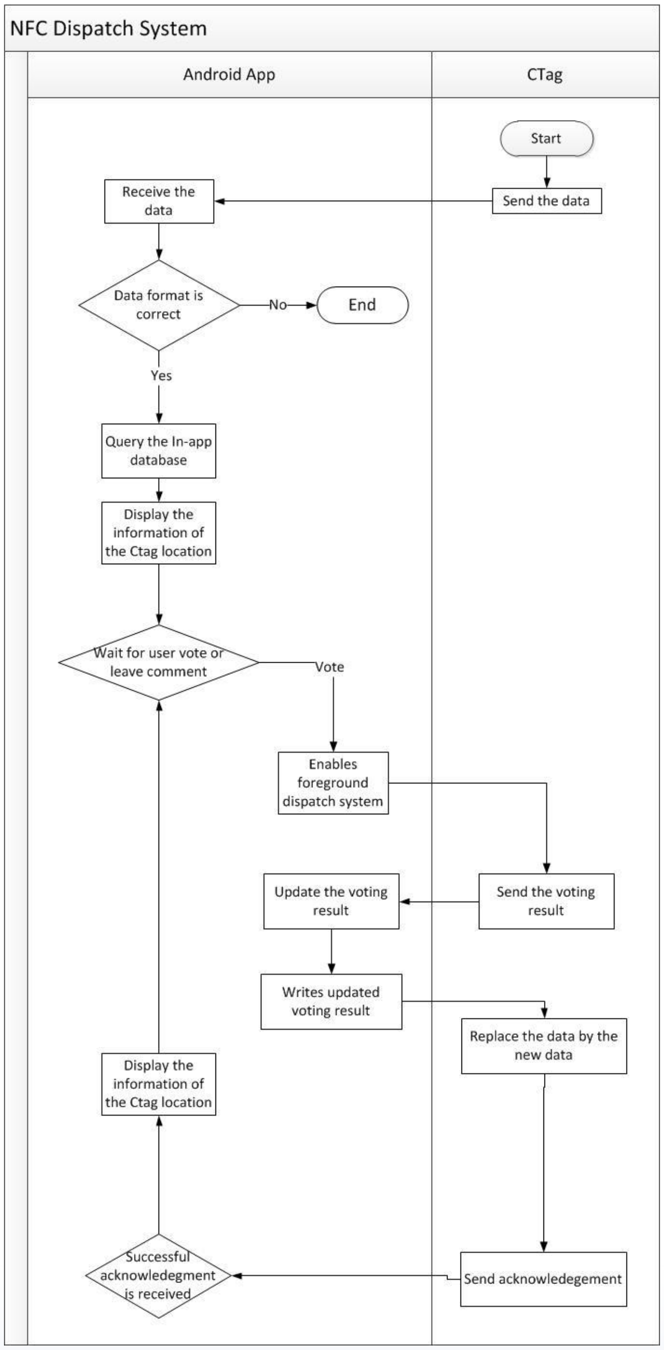
This method directly sends the CTag ID and string of comment to the webserver. The database in webserver will automatically generates a timestamp in insertion.

NFC Dispatch System

The NFC dispatch system is divided into two parts; NFC tag intent dispatch system and foreground dispatch system. Cucom has registered to a self creates external type. The registration can be found in the manifest file. The intent dispatch system allows Cucom to be pop up when user scans the CTag.

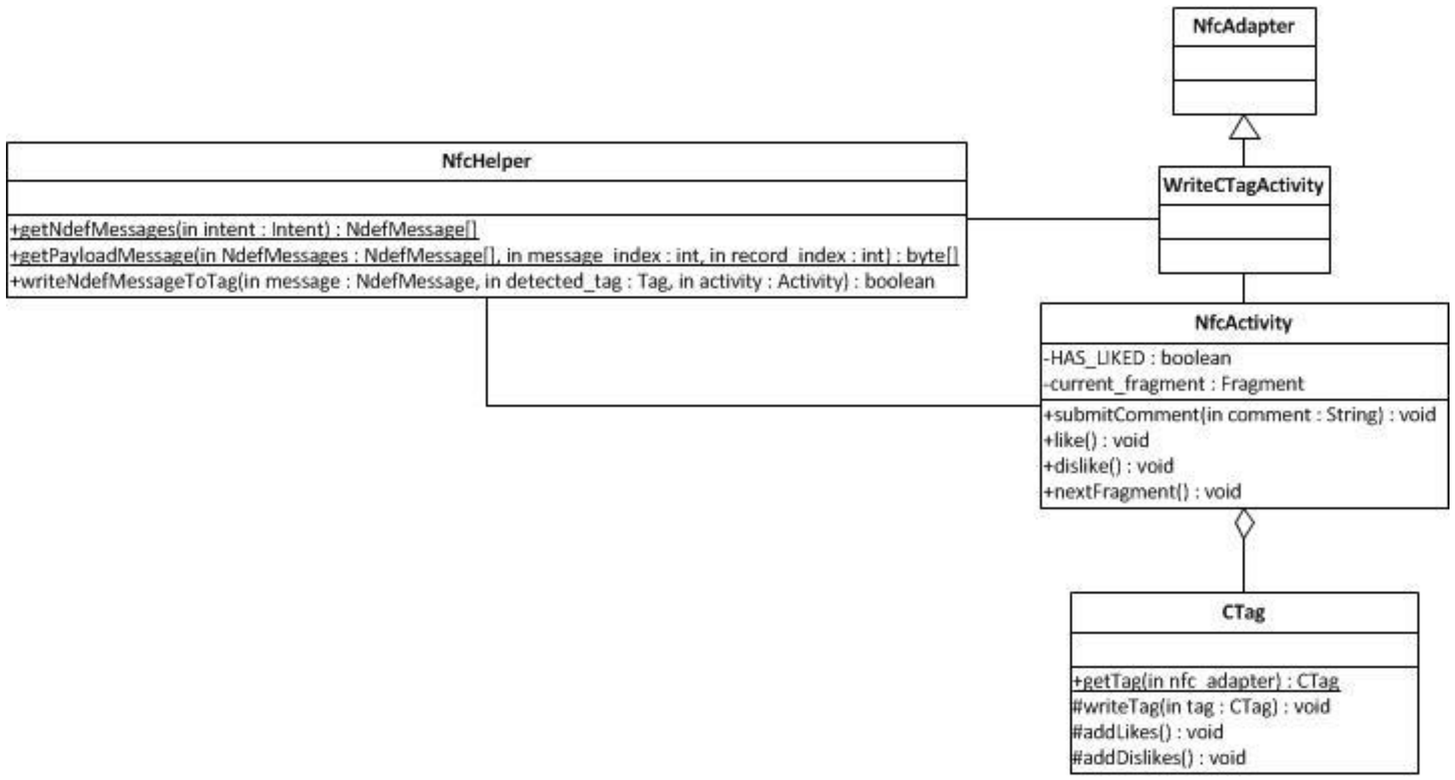
Workflow

After CTag data format is found, Cucom validate the data again for verifying the data format in payload. When user pressed the voting button, say “like”, the foreground dispatch system will be enabled. When the user touches its phone to an NFC tag, the Cucom will check the type name format, data type and data format inside the tag. If it discovered the same CTag is touched, the activity will read the data from the CTag. After data is ready, the corresponding like value will be incremented by 1, and then write it back to the same CTag. The whole process only needs 0.2s to finish. This atomicity avoids user to cheat on CTag and enhances the robustness.


NFC Dispatch System
Android App
CTag
Start
Send the data
Receive the data
Data format is correct
No
End
Yes
Query the In-app database
Display the information of the Ctag location
Wait for user vote or leave comment
Vote
Enables foreground dispatch system
Send the voting result
Update the voting result
Writes updated voting result
Replace the data by the new data
Display the information of the Ctag location
Send acknowledegement
Successful acknowledgment is received

Methods

Besides the given NfcAdapter and foreground dispatch system, We have added an NfcHelper that simplify the NFC read/write job. The class CTag also embedded a constructor that uses an NfcAdapter as the parameter. It simplifies all the reading procedures using the constructor.


NfcAdapter
NfcHelper
+getNdefMessages(in intent : Intent) : NdefMessage[]
+getPayloadMessage(in NdefMessages : NdefMessage[], in message_index : int, in record_index : int) : byte[]
+writeNdefMessageToTag(in message : NdefMessage, in detected_tag : Tag, in activity : Activity) : boolean
WriteCTagActivity
NfcActivity
-HAS_LIKED : boolean
-current_fragment : Fragment
+submitComment(in comment : String) : void
+like() : void
+dislike() : void
+nextFragment() : void
CTag
+getTag(in nfc_adapter) : CTag
#writeTag(in tag : CTag) : void
#addLikes() : void
#addDislikes() : void

NfcHelper

This is a static helper class that handles all the NDEF problems.

getNdefMessages(intent)

This method reads the intent of activity. If it contains an NFC tag, this method will return the array of NDEF messages inside the Tag.

getPayloadMessage(ndef_messages, message_index, record_index)

In the array of *NDEFMessage*, this method extracts the message by stating the message_index. Within the extracted message, it can further extracts from records to a specific record using the record_index.

writeNdefMessageToTag(message, detected_tag, activity)

This method writes the predefined *NDEFMessage* to a *Tag* object. This can be called in foreground dispatch system. The activity parameter is used to show error message.

WriteCTagActivity

This activity contains a foreground dispatch system object from NfcAdapter. In the initialization of this activity, it enables the foreground dispatch system allowing incoming NFC tag. When the read write process is done, it will invoke the callback method in NFCActivity to notify upload operation.

CTag

This class simulates the physical CTag object. It provides methods to manipulate data inside the CTag, such as *addLikes()* and *addDislikes().*

getTag(nfc_adapter)

This is the constructor of CTag. CTag can be constructed using an *NfcAdapter*, hence it simplifies all the job to get a CTag. Within this method, firstly, it gets an NFC tag from *NfcAdapter*, and then passes it to NfcHelper. NfcHelper will give a clean payload data. With parsing, the CTag object can be constructed and instantiated.

6.3 CTag

CTag is the NFC tag uses in Cucom. It represents the location where it sticks on. In the user perspective, it contains all the details of the location including name, description, photo and voting result. However in practice, it is impossible to store these large data (especially photo) into a single NFC tag. Not only because the capacity of NFC tag, but also the transferring speed is limited to 424 Kbps. To achieve this, we have saved all the fixed data into the mobile app such as photo, name and description. In the CTag, an ID and voting result is needed to store.

We have designed the data structure in the payload as

ID4	ID3	ID2	ID1	ID0	L7	L6	L5	L4	L3	L2	L1	L0	D7	D6	D5	D4	D3	D2	D1	D0

Where ID stands for CTag ID, L stands for Like counts and D stands for Dislike counts. ID has 5 digits, hence the maximum value can up to 100 thousands. For like count and dislike count, each of them has 8 digits, which enables up to 10 million likes.

Tag type

Since, CTag requires user to write data back into the tag, the tag must not be read-only. For read/write NFC tag, it is free to write data into the tag, hence security is another major concerned. The available tags that support encryption or authentication are Mifare Classic and Mifare Ultralight C. Mifare Classic is an ordinary RFID card, but new smartphones gradually do not support Mifare Classic, so Classic is not a good choice. Another option is using Mifare Ultralight C, since it gives security to the NFC tag. This is the ideal CTag tag type. However, to manipulate the authentications, a confidential datasheet is required. Hence we cannot use this tag type for now.

7 Cucom Implementation

In this chapter, we will go through the implementation of Cucom both in server-side and client side. Server-side mainly performs database operation while the client-side is the main application of Cucom.

7.1 Server

We have considered two database management systems: SOLite and MySQL. The following are the advantages of MySQL and SQLite.

MySQL	SQLite
Perfect for concurrency transactions on the data	Easy to set up
Well suited for multi-user environment	Great for databases need on a temporary basis or for test purpose
Perfect in a Client Server Architecture setup	Suitable for using in embedded applications
Suited for managing users and permissions	Suitable for using on small to medium website

According to these advantages, we have chosen MySQL as our database management system. The reason to choose MySQL is that our project is a multi-user environment, users can concurrent submit their comments to server using MySQL. It is perfect in Client and Server architecture setup. Moreover, it is suitable to manage users and their permissions. We can take benefit on it for further development to add user account.

The version of MySQL we are using is 5.0.51a-24+lenny5 - (Debian). Also, we use the PHP as the interface.

7.1.1 MySQL

We have created 3 tables: comment, ctag, and table tag. The following is the SQL command that create these tables.

The ctag table

```
CREATE TABLE ctag
    (ctag_id        INT(5),
     nfc_tag_id     TEXT,
     name           TEXT,
     description    TEXT,
     geo_x          DOUBLE,
     geo_y          DOUBLE,
     likes          INT(11),
     dislikes       INT(11),
     PRIMARY KEY (ctag_id ASC));
```

This SQL command creates the ctag table with ctag_id, nfc_tag_id, name, description, geo_x, geo_y, likes and dislikes (Refer to Design for the description of attributes). The ctag_id is set as the primary key. The data are stored in ascending order of ctag_id.

The comment table

```
CREATE TABLE comment
    (ctag_id    INT(5),
     comment_id TEXT,
     name       TEXT,
     content    TEXT,
     time       TIMESTAMP,
     PRIMARY KEY (ctag_id ASC, comment_id ASC));
```

This SQL command creates the comment table with ctag_id, comment_id, name, content, and time. The ctag_id and comment_id are set to be the primary key. The data are stored in ascending order of ctag_id and comment_id.

The tag table

```
CREATE TABLE tag
    (ctag_id    INT(5),
     keyword    TEXT,
    PRIMARY KEY (ctag_id, keyword))
```

This SQL command creates the tag table with ctag_id and keyword. Both of them are the primary key of the table.

7.1.2 PHP

The format of parameter passing between Android app and PHP is using JSON. The reason we are using JSON but not XML is that Java and PHP have libraries on JSON. With this support, it is easy for us to convert raw data to JSON and perform the data transmit.

Our PHP involves 6 modules: *db_connect*, *get_comments,* *get_rank*, *get_search_result*, *submit_comment,* and *update_ctag*.

db_connect

It is a class used to connect with the database. The *connect()* is the function to connect the database using the login parameters. The *getDatabase()* will return the database object.

```
<?
    class DB_CONNECT {
        private $db;
        // constructor
        function __construct() {
            // connecting to database
            $this->connect();
        }
        // destructor
        function __destruct() {
            // closing db connection
            $this->close();
        }
        /**
         * Function to connect with database
         */
        function connect() {
            require_once __DIR__ . '/db_config.php';
            // Connecting to mysql database
            $this->db = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or
                    die(mysqli_error());
            // returing connection cursor
            return $this->db;
        }
```

```
                /**
                * Function to get database object
                */
                function getDatabase(){
                        return $this->db;
                }
                /**
                 * Function to close db connection
                 */
                function close() {
                        // closing db connection
                }
        }
?>
```

get_comments

It uses *db_connect* to connect the database. Then, it requests to query the comment table in database using the SELECT statement with the *$_POST['ID']*, which is the *ctag_id* pass by the Android app. After that, it fetches the result and encodes the result from array to JSON.

```
<?
	// Connect to database
	require_once __DIR__ . '/db_connect.php';
	$db = new DB_CONNECT();
	$db = $db->getDatabase();

	// Query
	$sql = "SELECT * FROM `comment` WHERE `ctag_id` = '" . $_POST['ID'] .
			"' ORDER BY `time` ASC";
	$result = $db->query($sql);

	//Fetch the result
	$comments = array();
	$times = array();
	$i = 0;
	while(($row = $result->fetch_object()) != NULL)
	{
		$comments[$i] = $row->content;
		$times[$i] =  $row->time;
		$i++;
	}

	//convert to JSON
	$result = array();
	$result[0] = $times;
	$result[1] = $comments;
	echo json_encode($result);
?>
```

get_rank

If the database is connected successful, it will request querying the ctag table in database using the SELECT statement to select 10 records that in descending order. It fetches the result and encodes the result from array to JSON.

```
<?php
        // Connect to database
        require_once __DIR__ . '/db_connect.php';
        $db = new DB_CONNECT();
        $db = $db->getDatabase();

      //Query top 10 likes
        $sql = " SELECT DISTINCT ctag_id, name, description, likes, dislikes FROM
                        ctag ORDER BY likes desc LIMIT 10";
        $search_result = $db->query($sql);
      //Fetch the result
      while(($obj = $search_result->fetch_object())!==NULL)
      {
              $return[$i]->ctag_id = $obj->ctag_id;
              $return[$i]->name = $obj->name;
              $return[$i]->description =$obj->description;
              $return[$i]->likes = $obj->likes;
              $return[$i]->dislikes = $obj->dislikes;
              $i++;
      }
      //return “nothing” or JSON Object
      if($i != 0)
              echo json_encode($return);
      else
              echo "nothing";

?>
```

get_search_result

It is used to get the user search parameter using POST method. Then decode the search parameter. The first element of *$send* is the keyword the user type in. The second element is the distinct categories selected. The Third element is the facility categories selected. The forth element is the faculty categories selected.

```
//get the input from android
	$string = $_POST['send'];
	$send = json_decode($string);
	$search_word = $send[0];
	$send[0] = NULL;
```

Then, for each of the sub-category, select the *ctag_id* and store it and the number of occurrence

```
	foreach($send as $arr)
	{
		//skip the search_word
		if($skip == 0)
			$skip = 1;
		else
		{
			if($arr!= NULL)
			{
				$input_category = 1;
				$no_of_category ++;
				$sql=generateCategorySql($arr, $page_no);
```

```
                    // Query
                    $search_result = $db->query($sql);
                    while(($obj = $search_result->fetch_object())!==NULL)
                    {
                                                                        if(array_search($obj-
                        >ctag_id,$id_arr)==NULL||array_search($obj->ctag_id,$id_arr)==false )
                         {
                              $id_arr[$i] = $obj->ctag_id;
                              $i++;
                              $count[$obj->ctag_id] = 1;
                         }
                         else
                              $count[$obj->ctag_id]++;
                    }
               }
          }
     }
```

The following is the function that generates the SQL that request select ctag_id from database in tag table.

```
//generate sql for searching keyword within a category
//input: $arr, $page_no
//output: $sql
function generateCategorySql($arr)
{
        $sql = "SELECT ctag_id FROM tag WHERE keyword = ";
        $first = 1;
        foreach($arr as $var)
        {
                if($first == 1)
                {
                        $sql = $sql." '".$var."'";
                        $first = 0;
                }
                else
                        $sql = $sql." OR keyword = '".$var."'";
        }
        return $sql;
}
```

Then combine the sub-categories to filter the *ctag_id* by filter the number of occurrence equal to that of selected main categories. Then, search the keyword within these ctag_id.

```
//if user choose category
    if($input_category == 1)
    {
        $sql = "SELECT DISTINCT ctag_id, name, description,likes,dislikes FROM cucom_ctag WHERE";
        if (sizeof($id_arr)>=1)
        {
            $i=0;
            //for each result check if it exists in all category
            foreach($id_arr as $id)
            {
                if($count[$id]==$no_of_category)
                {
                    $get_info[$i]=$id;
                    $i++;
                }
            }
            $sql = $sql." (ctag_id =";
            $first = 1;

            foreach($get_info as $var)
            {
                if($first == 1)
                {
                    $sql = $sql." '".$var."'";
                    $first = 0;
                }
                else
                {
                    $sql = $sql." OR ctag_id = '".$var."'";
                }
            }
// if user input search_word only
          else
          {
               echo "nothing";
               exit;
          }

     }
     else if($search_word!="")
     {
          $search_word = filter_symbol($search_word);
          $sql = "SELECT DISTINCT ctag_id, name, description,likes,dislikes FROM cucom_ctag WHERE
                    name LIKE '%".$search_word."%' OR description LIKE '%".$search_word."%'";
     }
     else
     {
          echo "nothing";
          exit;
     }
```

Finally it will fetch the result and return it in form of JSON.

```
                $sql = $sql.")";
                // if user input search_word
                if($search_word!="")
                     $sql = $sql." AND (name LIKE '%".$search_word."%' OR description LIKE
                           '%".$search_word."%')";
           }
$i = 0;
     $info = $db->query($sql);

     //fetch result
     if($info!=false)
          while(($obj = $info->fetch_object())!==NULL)
          {
                $return[$i]->ctag_id = $obj->ctag_id;
                $return[$i]->name = $obj->name;
                $return[$i]->description =$obj->description;
                $return[$i]->likes = $obj->likes;
                $return[$i]->dislikes = $obj->dislikes;
                $i++;
          }
     //return to android
     if($i != 0)
          echo json_encode($return);
     else
          echo "nothing";
```

submit_comment

It is used to request inserting the comment into the comment table in database using the *$_POST['comment']* that passed by Android app.

```
<?
        // Connect to database
        require_once __DIR__ . '/db_connect.php';
        $db = new DB_CONNECT();
        $db = $db->getDatabase();

        // Query for inserting
        $stmt = $db->prepare("INSERT INTO `comment` VALUES (?, NULL, NULL, ?,
                        CURRENT_TIMESTAMP)");
        $stmt->bind_param('ss', $_POST['ID'], $_POST['comment']);
        $stmt->execute();

 ?>
```

update_ctag

It is used to update the likes and dislikes of ctag table in database using the $_POST['like_count'] and $_POST['dislike_count'].

```
<?
    // Connect to database
    require_once __DIR__ . '/db_connect.php';
    $db = new DB_CONNECT();
    $db = $db->getDatabase();

    // Query to update the number of likes and dislikes
    if(isset($_POST['ID']) && isset($_POST['like_count']) && isset($_POST['dislike_count']))
    {
        $command =  "UPDATE `ctag` SET `likes` = '" . $_POST['like_count']  . "' WHERE
                        `ctag_id` = " . $_POST['ID'];
        $db->query($command);
        $command =  "UPDATE ctag` SET `dislikes` = '" . $_POST['dislike_count']  . "' WHERE
                        `ctag_id` = " . $_POST['ID'];
    $db->query($command);
    }
?>
```

7.2 Android App

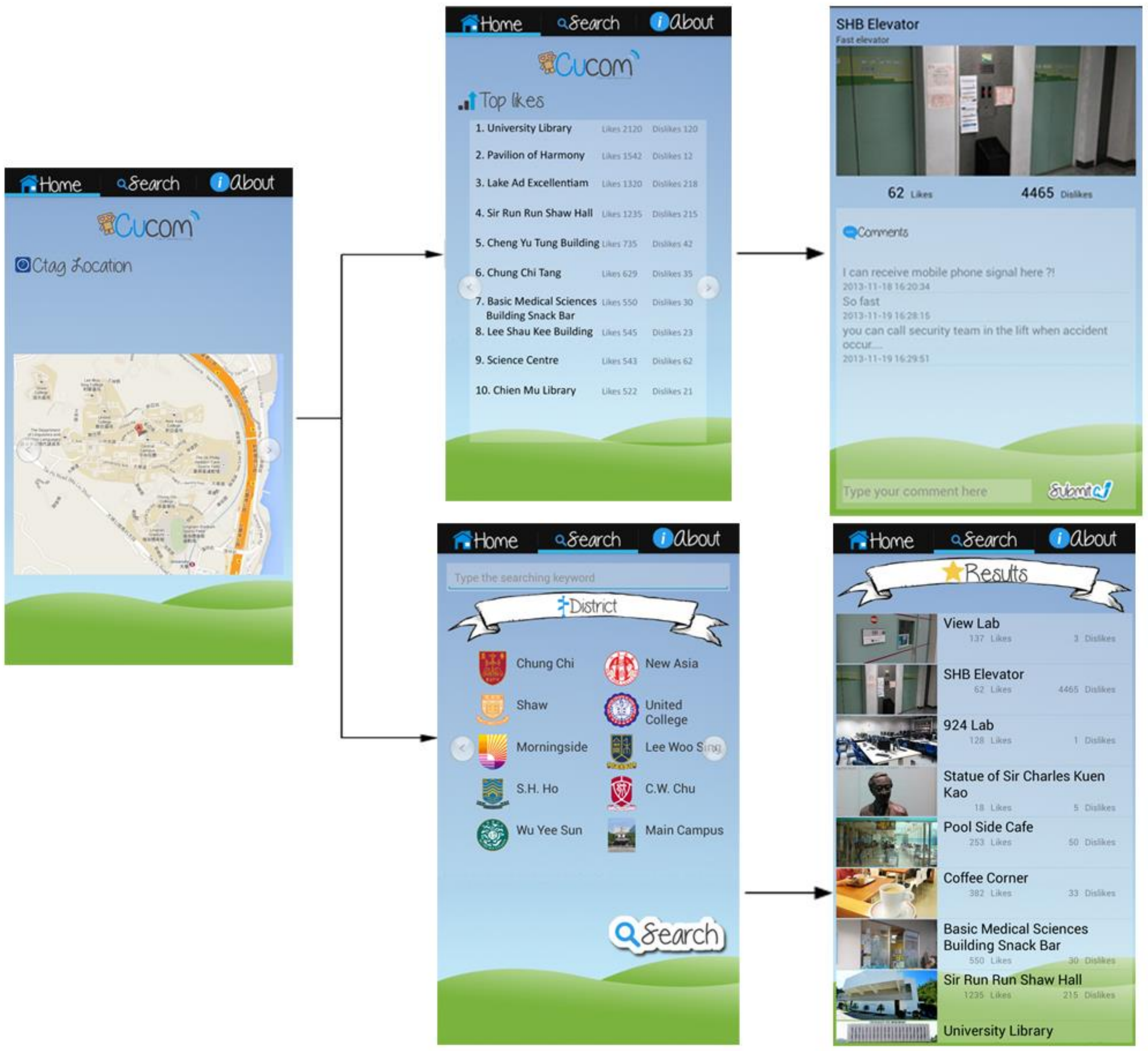
In the Cucom application, we have chosen to use Java with Eclipse (provided by official Android SDK) in Windows. Since we are developing NFC application which is relative new in the Android market, using official Android SDK can get better support. We also limits the version of Android, from version 3.0 (Honeycomb) to version 4.2 (Jelly Bean), in order to support NFC foreground dispatch system.

7.2.1 User Interface

The following is the user interfaces design of Cucom when users open the apps. The top of the interface used the tab bar which contains Home, Search, and About.

The following is the user interfaces design of Cucom when users open the apps.

Express yourself in a new way



1. University Library	Likes 2120	Dislikes 120
2. Pavilion of Harmony	Likes 1542	Dislikes 12
3. Lake Ad Excellentiam	Likes 1320	Dislikes 218
4. Sir Run Run Shaw Hall	Likes 1235	Dislikes 215
5. Cheng Yu Tung Building	Likes 735	Dislikes 42
6. Chung Chi Tang	Likes 629	Dislikes 35
7. Basic Medical Sciences Building Snack Bar	Likes 550	Dislikes 30
8. Lee Shau Kee Building	Likes 545	Dislikes 23
9. Science Centre	Likes 543	Dislikes 62
10. Chien Mu Library	Likes 522	Dislikes 21

SHB Elevator

Fast elevator

62 Likes 4465 Dislikes

Comments

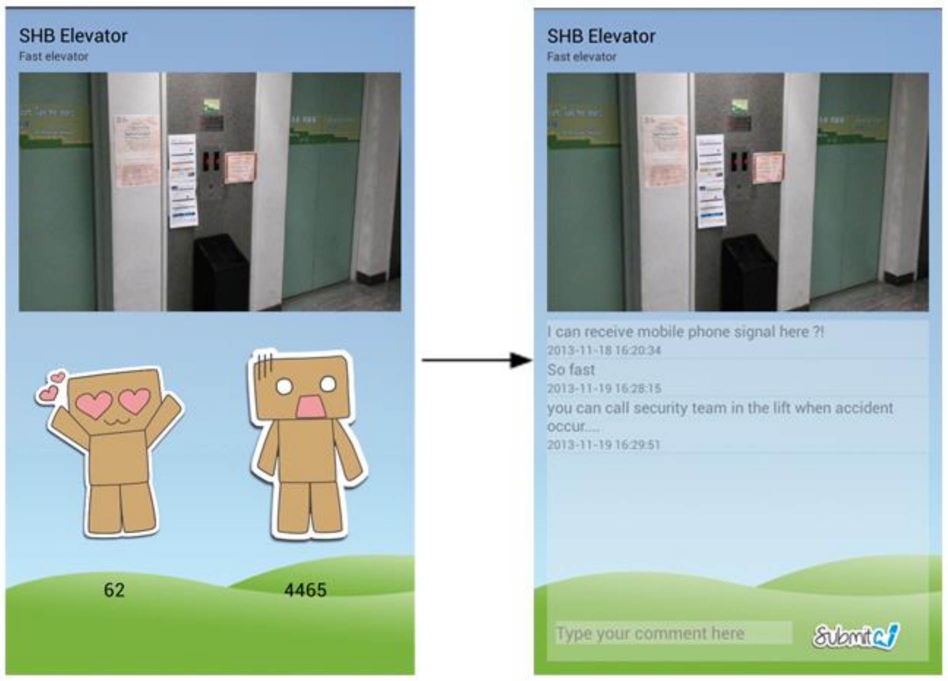
I can receive mobile phone signal here ?!
2013-11-18 16:20:34

So fast
2013-11-19 16:28:15

you can call security team in the lift when accident occur....
2013-11-19 16:29:51

View Lab	137 Likes	3 Dislikes
SHB Elevator	62 Likes	4465 Dislikes
924 Lab	128 Likes	1 Dislikes
Statue of Sir Charles Kuen Kao	18 Likes	5 Dislikes
Pool Side Cafe	253 Likes	50 Dislikes
Coffee Corner	382 Likes	33 Dislikes
Basic Medical Sciences Building Snack Bar	550 Likes	30 Dislikes
Sir Run Run Shaw Hall	1235 Likes	215 Dislikes
University Library		

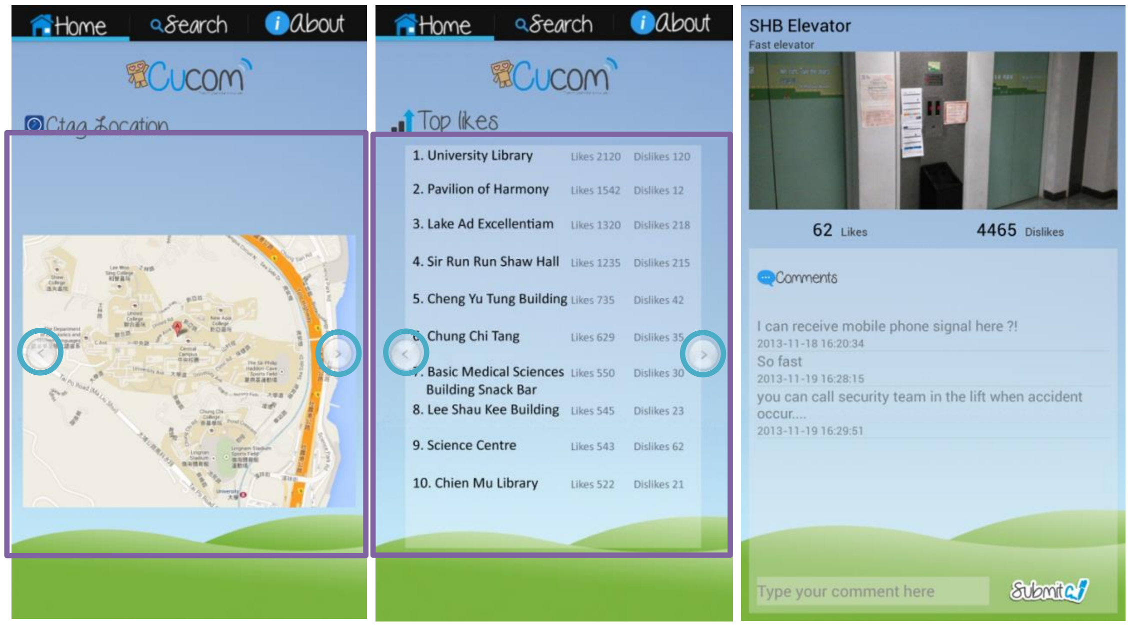
The following is the user interfaces design of Cucom when users user NFC to scan CTag.


SHB Elevator
Fast elevator

I can receive mobile phone signal here ?!
2013-11-18 16:20:34
So fast
2013-11-19 16:28:15
you can call security team in the lift when accident occur....
2013-11-19 16:29:51

In order to fit the background for different display size of smartphone, we have use draw 9-patch tool to create a NinePatch graphic. Now, we will describe the implementation of each layout.

Home Tab



1. University Library	Likes 2120	Dislikes 120
2. Pavilion of Harmony	Likes 1542	Dislikes 12
3. Lake Ad Excellentiam	Likes 1320	Dislikes 218
4. Sir Run Run Shaw Hall	Likes 1235	Dislikes 215
5. Cheng Yu Tung Building	Likes 735	Dislikes 42
6. Chung Chi Tang	Likes 629	Dislikes 35
7. Basic Medical Sciences Building Snack Bar	Likes 550	Dislikes 30
8. Lee Shau Kee Building	Likes 545	Dislikes 23
9. Science Centre	Likes 543	Dislikes 62
10. Chien Mu Library	Likes 522	Dislikes 21

SHB Elevator
Fast elevator

62 Likes 4465 Dislikes

I can receive mobile phone signal here ?!
2013-11-18 16:20:34
So fast
2013-11-19 16:28:15
you can call security team in the lift when accident occur....
2013-11-19 16:29:51

Type your comment here

At the home tab, we created two subpages in the form of fragments. Then, we use the View Flipper to be the container for CTag location and Top likes fragments. There are two buttons: previous and next. The user can either swipe or tab the buttons to shift pages.

```
    // Get the object from layout
    View rootView = inflater.inflate(R.layout.home, container, false);
    view_flipper = (ViewFlipper) rootView.findViewById(R.id.home_viewflipper);
    next = (ImageButton) rootView.findViewById(R.id.home_next_button);
    previous = (ImageButton) rootView.findViewById(R.id.home_previous_button);

    // Set the animation
    enter_left_to_right = AnimationUtils.loadAnimation(getActivity(), R.anim.enter_left_to_right);
    exit_left_to_right = AnimationUtils.loadAnimation(getActivity(), R.anim.exit_left_to_right);
    enter_right_to_left = AnimationUtils.loadAnimation(getActivity(), R.anim.enter_right_to_left);
    exit_right_to_left = AnimationUtils.loadAnimation(getActivity(), R.anim.exit_right_to_left);

    // Set the onClickListener to the next button
      next.setOnClickListener(new OnClickListener(){
           public void onClick(View view){
             showNextView();
           }
       });

       // Set the onClickListener to the back button
       previous.setOnClickListener(new OnClickListener(){
            public void onClick(View view){
              showPreviousView();
            }
       });
```

The above is the code in the *onCreate*. When the home tab is created, it will get the object from the layout and create *animation*. The *enter_left_to_right* is the *animation* of swipe the new *fragment* from left to right. The *enter_right_to_left* is the *animation* of swipe from right to left. The *exit_left_to_right* is the *animation* of swipe from left to right. The *exit_right_to_left* is the *animation* of swipe from right to left. Then, it sets the onClickListener for handling the click event in the next and previous buttons. The next button will call the *showNextView* function and the previous button will call the *showPreviousView* function in *onClick* function.

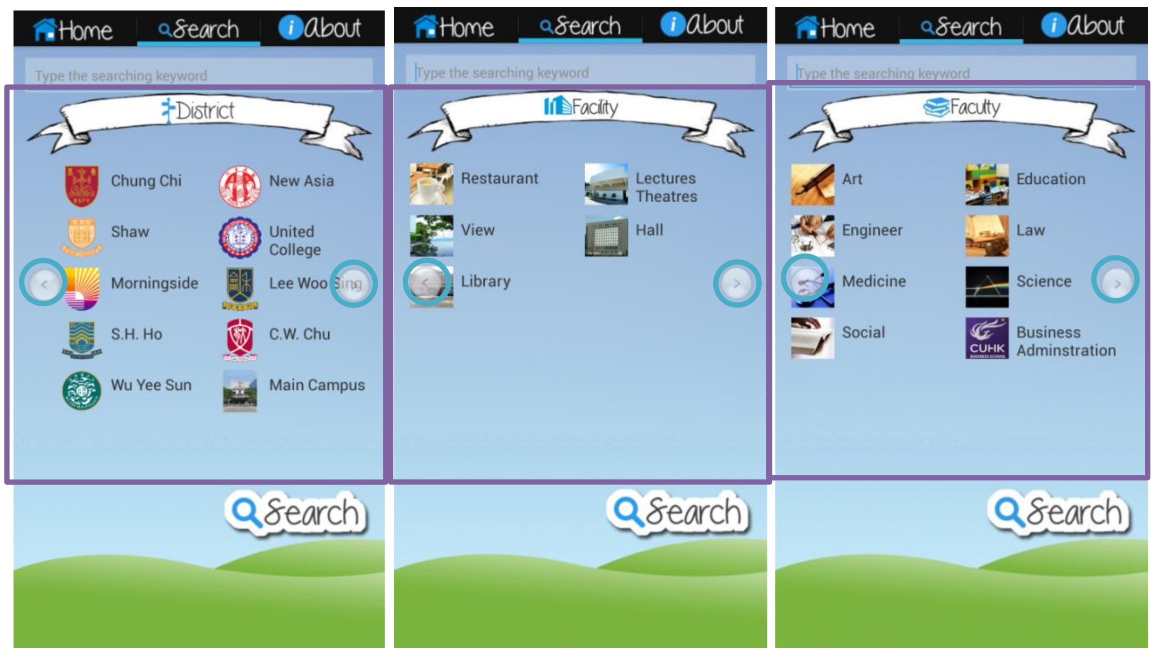
```
/** showPreviousView
 * Flip to the previous fragment
 */
public void showPreviousView() {
 if(view_flipper!=null){
        view_flipper.setOutAnimation(exit_left_to_right);
        view_flipper.setInAnimation(enter_left_to_right);
        view_flipper.showPrevious();
        }
}

/** showNextView
 * Flip to the next fragment
 */
public void showNextView() {
 if(view_flipper!=null){
        view_flipper.setInAnimation(enter_right_to_left);
        view_flipper.setOutAnimation(exit_right_to_left);
        view_flipper.showNext();
         }
}
```

In the *showPreviousView* function, it sets the out Animation to *exit_left_to_right* and the in *animation* to *enter_left_to_right*. Then, it will call the *showPrevious* in *view_flipper* to show the previous *fragment*. In the *showNextView* function, it sets the in *animation* to *enter_right_to_left* and the out Animation to *exit_right_to_left*. Then, it will call the *showNext* in *view_flipper* to show the next *fragment*.

```
<LinearLayout android:orientation="vertical">
    <!-- CTag Title -->
    <TextView android:id="@+id/result_ctag_name"/>
    <TextView android:id="@+id/result_ctag_description"/>

    <!-- Image -->
    <ImageView android:id="@+id/result_ctag_photo"/>

    <!-- Likes and Dislikes -->
    <LinearLayout android:orientation="horizontal" >
        <TextView android:id="@+id/result_ctag_likes"/>
        <TextView android:text="Likes"/>
        <TextView android:id="@+id/result_ctag_dislikes"/>
        <TextView android:text="Dislikes"/>
    </LinearLayout>

    <!-- Comment title-->
    <LinearLayout android:orientation="horizontal">
       <ImageView android:src="@drawable/comments"/>
    </LinearLayout>

    <!-- Comment list-->
    <ListView android:id="@+id/result_ctag_listView">
    </ListView>

    <!-- Submit comment-->
    <LinearLayout android:orientation="horizontal">
        <EditText android:id="@+id/result_ctag_editText"/>
        <ImageButton android:id="@+id/result_ctag_submit_button"/>
    </LinearLayout>

</LinearLayout>
```

The above is xml for the description of CTag. The top of the layout is the *TextView* that shows the name and description of CTag. The center is the *ImageView* that shows the photo CTag. Below the *ImageView*, there are *TextViews* to show the likes and dislikes. They are embraced by the *LinearLayout* which the orientation is horizontal. At the bottom, there are *EditText* and *ImageButton* for user.

Search Tab

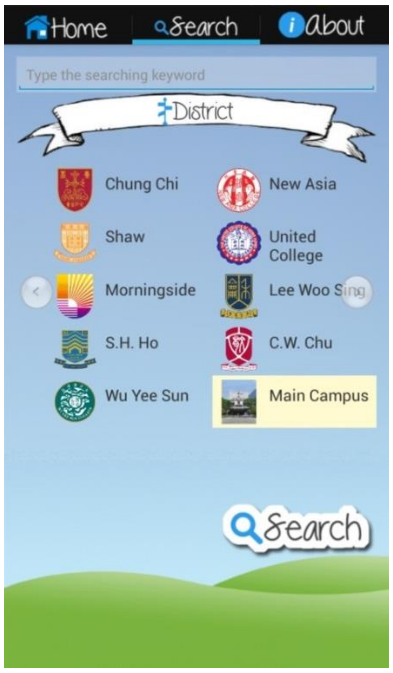


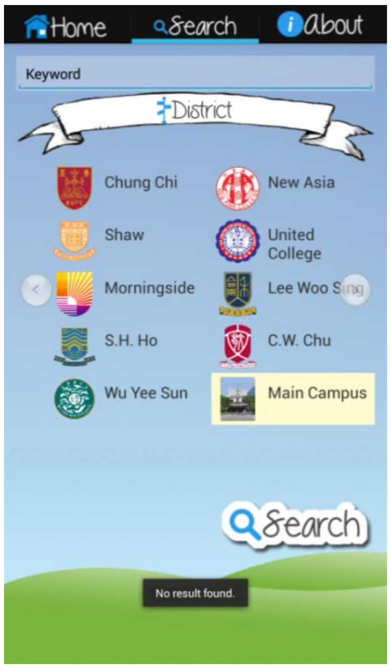
At the Search tab, we created three subpages in the form of *fragments*. Then, we use the *ViewFlipper* to be the container for District, Facility, and Faculty *fragments*.

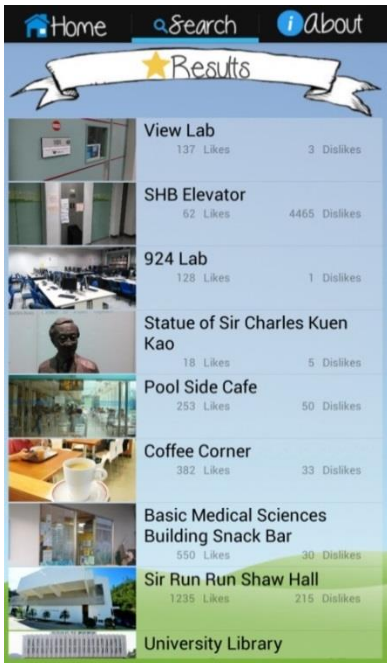
Each *fragment* will contain a *GridView* with two columns. On top of *ViewFlipper*, there are two buttons: previous and next. The user can either swipe or tab the buttons to shift pages. At the bottom, there are the *ImageButton* for the search.

```
<LinearLayout android:orientation="horizontal" >
    <!-- Image -->
    <ImageView android:id="@+id/category_image"/>
    <!—Category name -->
    <TextView android:id="@+id/category_name"/>
</LinearLayout>
```

The above is the xml of each items in *GridView*. There is a *ImageView* on the left hand side of the category. The right hand side is the *TextViews* of category name.







```
//District
   private GridView district_gridView;

   private static final String[] DISTRICT_CATEGORY =
     new String[]{"Chung Chi","New Asia","Shaw","United
     College","Morningside","Lee Woo Sing", "S.H. Ho", "C.W. Chu","Wu
     Yee Sun", "Main Campus"};

   private int district_focus[] =
     new int[DISTRICT_CATEGORY.length];
```

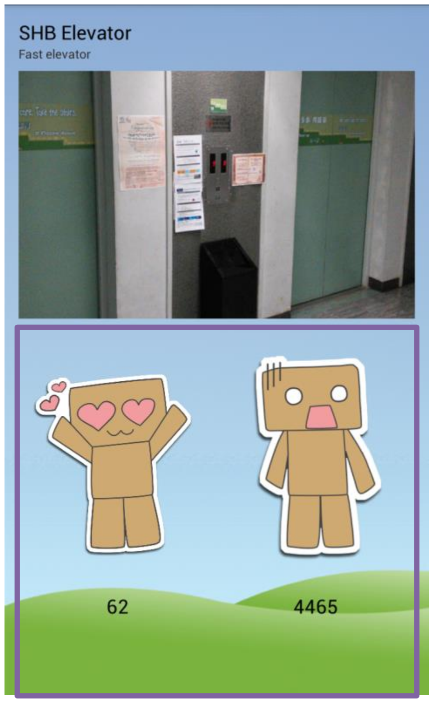
```
//Facility
  private GridView facility_gridView;

  private static final String[] FACILITY_CATEGORY =
     new String[]{"Restaurant","Lectures Theatres","View","Hall","Library"};

  private int facility_focus[] = new int[FACILITY_CATEGORY.length];

//Faculty
   private GridView faculty_gridView;

   private static final String[] FACULTY_CATEGORY =
      new String[]{"Art","Education","Engineer","Law",
      "Medicine","Science","Social","Business Adminstration"};

   private int faculty_focus[] = new int[FACULTY_CATEGORY.length];
```

The *district_focus[]*, *facility_focus[]* and *faculty_focus[]* are used to record which of the items are selected. The selected items will be highlighted. When the search result is null, there will be pop up a *Toast* message to notice users no result found.

```
<LinearLayout android:orientation="horizontal">
    <!-- Image -->
    <ImageView android:id="@+id/search_result_detail_imageView"/>
    <!-- Description -->
    <LinearLayout android:orientation="vertical">
        <TextView android:id="@+id/search_result_detail_name"/>
        <!-- Likes and Dislikes -->
        <LinearLayout android:orientation="horizontal">
                <TextView android:id="@+id/search_result_detail_likes"/>
                <TextView android:text="Likes"/>
                <TextView android:id="@+id/search_result_detail_dislikes"/>
                <TextView android:text="Dislikes"/>
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

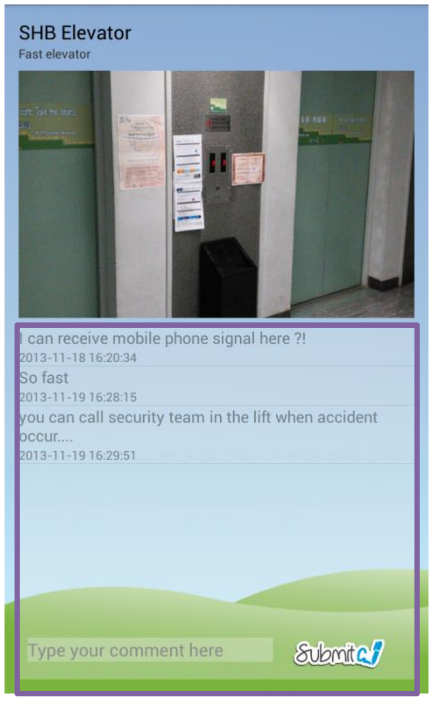
The above is the xml of each row in result list. The left hand side is the *ImageView* to show the photo of search result. The right hand side are the *TextViews* that show the name, likes, and dislikes of location represented by CTag. When the users click the row, the app will open the description of CTag that same as that in top likes.

Scanning CTag



SHB Elevator
Fast elevator

62
4465



SHB Elevator
Fast elevator

I can receive mobile phone signal here ?!
2013-11-18 16:20:34
So fast
2013-11-19 16:28:15
you can call security team in the lift when accident occur....
2013-11-19 16:29:51

```
<LinearLayout android:orientation="vertical">
    <!-- CTag Title -->
    <LinearLayout android:orientation="horizontal" >
      <!-- CTag name and description -->
        <LinearLayout android:orientation="vertical" >
            <TextView android:id="@+id/ctag_name"/>
            <TextView android:id="@+id/ctag_description/>
        </LinearLayout>
    </LinearLayout>
    <!-- Image -->
    <ImageView android:id="@+id/ctag_image"/>
   <!-- fragment container -->
    <FrameLayout android:id="@+id/fragment_container"/>
```

The above layouts are opened when users scan the CTag using NFC. The top of layout are the *TextViews* that show the name and description. The center of layout is the *ImageView* that show the photo of the location represented by CTag. The below is the *FrameLayout* that contains two *fragments*: *fragment_like* and *fragment_comment*.

```
<LinearLayout android:orientation="vertical">

     <!-- Like/Dislike button -->
    <LinearLayout android:orientation="horizontal" >

        <ImageButton android:id="@+id/button_like"/>
        <ImageButton android:id="@+id/button_dislike"/>

    </LinearLayout>

    <!-- Like/Dislike counts -->
    <LinearLayout android:orientation="horizontal" >

        <TextView android:id="@+id/ctag_likes"/>
        <TextView android:id="@+id/ctag_dislikes"/>

    </LinearLayout>

</LinearLayout>
```

The *fragment_like* contains *button_like* and *button_dislike* for voting. The two *TextView* at the bottom shows the number of likes and dislikes.

```
<RelativeLayout>
   <!-- Comment list -->
    <ListView android:id="@+id/comment_listview"/>
   <!–comment submit -->
    <LinearLayout android:orientation="horizontal">
        <EditText android:id="@+id/comment_edittext"/>
        <ImageButton android:id="@+id/submit_comment_button"/>
    </LinearLayout>
</RelativeLayout>
```

The *fragment_comment* shows the comments in *comment_ListView*. There are the *comment_edittext* and the *submit_comment_button* for submitting the comment.

7.2.2 Modules

Search Engine

The search engine uses the keywords and set of filters to generate the result list. At the beginning, we use the filters to give the subset CTag locations. Within this subset, we use the keyword to further reduce the results. This is more efficient then doing the reverse (i.e. using the keywords to generate a temporary results, then use filters to further filter out the result), because according to the experience of OpenRice, user usually won't type any keywords. Instead, they select desired filters, then search. If we use keywords to generate result in the beginning, then it will give all the result in the database, hence it is not efficient.

In our implementation, we use bitwise operations to gather the selection information and do searching. A 32 bits string is used to record the selection of tags. Each bit represents a selection of the corresponding tag. If the bit set to 1 meaning that user has selected the tag, otherwise the bit remains at 0.

Category	Bit position	Code	Tag
District	0	0x000001	Chung Chi College
	1	0x000002	New Asia
	2	0x000004	Shaw
	3	0x000008	United College

	4	0x000010	Morining Side College
	5	0x000020	Lee Wo Sing College
	6	0x000040	Sin Heng Ho College
	7	0x000080	C. W. Chu
	8	0x000100	Wu Yee Sun
Facility	9	0x000200	Main Campus
	10	0x000400	Restaurant
	11	0x000800	Lectures Theaters
	12	0x001000	View
	13	0x002000	Hall
Faculty	14	0x004000	Library
	15	0x008000	Art
	16	0x010000	Education
	17	0x020000	Engineering
	18	0x040000	Law
	19	0x080000	Medicine
	20	0x100000	Science
	21	0x200000	Social
	22	0x400000	Business Administration

When a tag is pressed, it will update a binary string which representing the whole selection. In our implementation, we defined a variable, *tag_code* representing the string. It is initialized to zero, meaning that no category is selected.

```
// Initialize query bit string
tab_code = 0x00000000;
```

When the button is selected the listener will add a one on the corresponding bit. If the button is being deselected, the listener will clear the corresponding bit to zero.

```
/** onClickTagListener
 * listener to tagging buttons
 *
 * @param view the view object representing the selected button
 */
void onClickTagListener(View view)
{
    // Get the current selected button
    Category category = view.getCategory();

    // If the button is being selected
    if(view.isSelected())
        tag_code |= encode(category);

    // If the button is being deselected
    else
        tag_code &= ~encode(category);
}
```

The search operation can be done by looping each bit in the query bit string. In the beginning, we initialize an *Arraylist* for storing the list of selected tags. Using a pointer, we can track the current position in bit-string. It also allows us to decode back to tagging format. After all the bits are checked, we send the resulting list and keyword to the database for query.

```
/** search
 * Search operation
 * It will be invoked when the user click the search button
 */
void search()
{
     int cur_tag = 0x01; // Code declaring the current tag
     ArrayList<TAG> selected_tag_list = new ArrayList<TAG>(); // The result list of tags

     // Loop for each bit on tag_code
     while(tag_code != 0)
     {
          // If the current bit equals to 1, append the corresponding tag to result list
          if((tag_code & 0x01) == 1)
               selected_tag_list.add(decode(cur_tag));

          // update to next bit by shifting
          cur_tag  <<= 1;
          tag_code >>= 1;
     }

     // Use the result list and keyword to do searching
     cucom_web_db.search(selected_tag_list, keyword);
}
```

Voting System

The voting system involves two buttons; i.e. like and dislike. After the NFC activity is intent from NFC adapter, user can vote by clicking one of the buttons. If user press the like button, the button then invokes the method *like()*.In the *like()* method, it sends an intent to WriteCTagActivity. Along with the intent, it also attached a CTag object and a flag indicating the action, like.

```
/** like
 * like action
 */
void like()
{
    // Prepare intent
    Intent write_intent = new Intent(this, WriteCTagActivity.class);

    // Add data into intent
    write_intent.putExtra("CTAG_ID", ctag);
    write_intent.putExtra("IS_LIKE", Boolean.TRUE);

    // Fire intent
    startActivityForResult(write_intent, 2);
}
```

```
/** dislike
 * dislike action
 */
void dislike()
{
    // Prepare intent
    Intent write_intent = new Intent(this, WriteCTagActivity.class);

    // Add data into intent
    write_intent.putExtra("CTAG_ID", ctag);
    write_intent.putExtra("IS_DISLIKE", Boolean.TRUE);

    // Fire intent
    startActivityForResult(write_intent, 2);
}
```

When the updated voting result has been successfully written to the CTag, then the application will try to update the web database as well. To ensure the writing process is successful, we have implemented a callback method to receive error message. If no error message is found, then we can assume that the writing operation is successful.

```
@Override
protected void onActivityResult(int request_code, int result_code, Intent data) {

    // Check which request we're responding to
    if (request_code == NFC_ERROR_REQUEST)

        // Update likes to webserver
        if (result_code == LIKE_SUCCESS)
           cucom_web_db.updateLikes(ctag.getId(), ctag.getLikes());

        // Update Dislikes to webserver
        else if(result_code == DISLIKE_SUCCESS)
           cucom_web_db.updateDislikes(ctag.getId(), ctag.getDislikes());

        // Failed in writing process
        else
           Toast.makeText(context, "Failed to write on CTag", Toast.LENGTH_SHORT).show();
}
```

Commenting System

Commenting system is a simple mechanism to read and write text. Since NFC tag does not have enough memory space to store user comments, all the comments have to store on the internet. For each comment, we attach a timestamp for sorting.

When user browses to see the location details, the activity will get the comments from web database through *CucomWebDatabase* object. Then all comments will put into a list view that shows all the comments. To form a list, we have implemented an array adapter

```
/** StableArrayAdapter
 * @see ArrayAdapter
 * Stores lists of comments and timestamp used for preparing list view
 */
private class StableArrayAdapter extends ArrayAdapter<String> {

    /** Variable declarations
     * context The current application context
     * contents The list of comment contents
     * times The list of timestamp
     */
    private Context context;
    private ArrayList<String> contents;
    private ArrayList<String> times;
```

```
  /** Constructor
   * @param context The application context
   * @param contents List of comments
   * @param times List of timestamps
   */
   public StableArrayAdapter(Context context, ArrayList<String> contents, ArrayList<String> times)
    {
       // Call constructor of super class
       super(context, R.layout.list_item_comment, contents);

       // Set the values
       this.context = context;
       this.contents = contents;
       this.times = times;
    }
  @Override
   public View getView(int position, View convertView, ViewGroup parent)
   {
    // Get the view of each item row
   LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View row_view = inflater.inflate(R.layout.list_item_comment, parent, false);

    // Set values on the view
    ((TextView)row_view.findViewById(R.id.comment_content)).setText(contents.get(position));
    ((TextView)row_view.findViewById(R.id.comment_time)).setText(times.get(position));

    // Return the equipped view
    return row_view;

   }

   /** setComments
    * Used to update the list
    * @param comment_list The new comment list
    * @param time_list The new timestamp list
    */
   public void setComments(ArrayList<String>comment_list, ArrayList<String>time_list)
   {
    // Clear the lists
    contents.clear();
    times.clear();

    // Add each item to the lists
    for(int i = 0; i < comment_list.toArray(new String[0]).length; i++)
    {
         contents.add(comment_list.get(i));
         times.add(time_list.get(i));
    }

    // Update view
    notifyDataSetChanged();
   }
}
```

NFC Dispatch System

There are two Dispatch systems in this module. NFC tag intent dispatch system is easier to implement while NFC foreground dispatch system requires some callback methods. In order to use the NFC service of the mobile device, NFC permission is needed to be enabled.

```
<uses-permission android:name="android.permission.NFC"/>
```

NFC Tag Intent Dispatch System

In the manifest file, we have registered the intent filter of our external type.

```
<intent-filter>
   <action android:name="android.nfc.action.NDEF_DISCOVERED" />
   <category android:name="android.intent.category.DEFAULT" />
   <data
         android:host="ext"
         android:pathPrefix="/_____:___"
         android:scheme="vnd.android.nfc" />
</intent-filter>
```

Hence, whenever a CTag is scanned, the NFC activity in Cucom will be launched. The intent will attach an NFC Tag object. We have to construct the CTag using this object.

```
// Confirm intent type
if(NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction()))
{
    Log.d(NFC_DEBUG_TAG, "NFC Intent detected");

    // Get NDEF message
    NdefMessage[] ndef_messages = NfcHelper.getNdefMessages(getIntent());

    // Get CTAGDEF message
    byte[] ctagdef_message = NfcHelper.getPayLoadMessage(ndef_messages, 0, 0);

    // Construct a CTag object
    c_tag = new CTag(db, ctagdef_message);
    c_tag.retrieveInformation();
}
```

NFC Foreground Dispatch System

We enables the NFC foreground dispatch system using the *NfcAdapter* object.

```
//Get NFC adapter
nfc_adapter = NfcAdapter.getDefaultAdapter(getApplicationContext());

//Enable Foreground dispatch system
nfc_adapter.enableForegroundDispatch(this, pending_intent, iFilters, null);
```

When foreground dispatch system is enabled, whenever an NFC tag is found, it will send intent to this activity. We can get the tag from the *intent* object.

```java
//Get NFC tag
Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);

// Validate the intent attachment
if(tag == null){
    Log.d("NFC_Intent", "Not a tag");
    return;
}
Log.d("NFC_Intent", "Tag found");

// Read data from tag
NdefMessage[] messages;
messages = NfcHelper.getNdefMessages(intent);
byte[] ctagdef_message = NfcHelper.getPayloadMessage(messages, 0, 0);

// Construct a CTag object
CTag c_tag = new CTag(CTagDatabase.getReadOnlyDatabase(this), ctagdef_message);
```

NFC Helper

Besides the dispatch systems, we have implemented an NFC helper to perform read/write operations. The *getNdefMessages* method takes the intent as the parameter, and then gets the raw NFC message from the intent. After parsing, the raw message converts to NDEF record and NDEF message.

```java
        /** getNdefMessages
         * Get the NFC tag from intent then parse the data into array of NdefMessage
         * @param intent Activity intent of ACTION_NDEF_DISCOVERED
         * @return array of NDEF messages, null if unknown intent found
         */
        public static NdefMessage[] getNdefMessages(Intent intent)
        {
                // Prepare message array
                NdefMessage[] message = null;
           // Known intent found
           if(NfcAdapter.ACTION_NDEF_DISCOVERED.equals(intent.getAction()))
           {
                 // Get raw messages
                 Parcelable[] rawMessages = intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);

                 // Convert the raw messages to NdefMessage type
                 if(rawMessages != null)
                 {
                     message = new NdefMessage[rawMessages.length];
                     for(int i = 0; i < rawMessages.length; i++)
                         message[i] = (NdefMessage) rawMessages[i];
                 }
                 // Create empty message
                 else
                 {
                     byte[] empty = new byte[] {};
                     NdefRecord record = new NdefRecord ( NdefRecord.TNF_UNKNOWN, empty, empty, empty);
                     NdefMessage msg = new NdefMessage( new NdefRecord[] {record});
                     message = new NdefMessage[]{msg};
                 }
           }
           // Unknown intent
           else
           {
                 Log.d("Intent", "Unknown intent.");
           }
           return message;
       }
```

Another method, *writeNdefMessageToTag,* is used to write an NDEF message to an NFC tag. Since, the writing process relies on the communication to the NFC tag, thorough error checking is needed.

```
        /** writeNdefMessageToTag
         * Write an NDEF message into a tag
         * @param message NDEF message
         * @param detectedTag
         * @param activity Activity for showing error toast message and change image
         * @return true when success, false when failed
         */
        public static boolean writeNdefMessageToTag(NdefMessage message, Tag detected_tag, Activity
activity)
        {
                int size = message.toByteArray().length;

                try {
                        // Get the NDEF object
                        Ndef ndef = Ndef.get(detected_tag);

                        // If the tag is empty
                        if(ndef != null)
                        {
                                ndef.connect();

                                // Failed if the tag is read-only
                                if(!ndef.isWritable())
                                {
                           Toast.makeText(activity.getApplicationContext(), "Tag is read-only",
                           Toast.LENGTH_SHORT).show();
                                        return false;
                                }
                                // Failed if the capacity of tag is not enough
                                if(ndef.getMaxSize() < size)
                                {
                                        Toast.makeText(activity.getApplicationContext(),
                           "The data cannot written to tag, Tag capacity is " + ndef.getMaxSize() +
                           " bytes, message is " + size + " bytes.", Toast.LENGTH_SHORT).show();
                                        return false;
                                }

                                // Writes the message to the NFC tag
                                ndef.writeNdefMessage(message);
                                ndef.close();
                        }
```

```
            // If the tag is empty
            else
            {
                // Prepare an object for formating
                NdefFormatable ndef_format = NdefFormatable.get(detected_tag);
                if(ndef_format != null)
                {
                    try{
                        // Format the NFC tag and replace with the new message
                        ndef_format.connect();
                        ndef_format.format(message);
                        ndef_format.close();

                    }catch(IOException e)
                    {
                        Toast.makeText(activity.getApplicationContext(),
                        "Failed to format tag", Toast.LENGTH_SHORT).show();
                        return false;
                    }
                }
                else
                {
                    Toast.makeText(activity.getApplicationContext(),
                    "NDEF is not supported", Toast.LENGTH_SHORT).show();
                    return false;
                }
            }
        }catch(Exception e)
        {
            Toast.makeText(activity.getApplicationContext(),
            "Write operation is failed", Toast.LENGTH_SHORT).show();
            return false;
        }
        return true;
    }
```

8 LIMITATIONS AND DIFFICULTIES

8.1 NFC Tag Security

Cucom not only requires user to read data from NFC tag, it also requires user to write data back to the NFC tag. Hence, it is difficult to ensure the security of the NFC tag. In our available tags (Type 2), there are only two modes; read/write mode and read-only mode. Read-only mode can guarantee the data inside the tag cannot be changed, however, user cannot write back on the tag. Read/write mode allows user to write data on the tag freely without authentication, it is easy to attack the NFC tag by replacing the data inside the tag. In Google Market, it already exist free applications allowing user to write data on NFC tag.

Another solution is using signature to encrypt the data inside the tag. Although attacker can still mess up with the data inside the tag, but attacker cannot affect the data in CTag format nor clone a new CTag.

There are two feasible tag types which equipped with good security and allow read/write operations; Mifare Classic and Mifare Ultralight C. Since new smartphones gradually knock out the support towards Mifare Classic, it does not have good compatibility. Mifare Ultralight C seems to be a good replacement of CTag, but coding with the authentication service requires complete datasheet, which is not easily available. Hence, this is our limitation.

8.2 NFC Tag Lifetime

The NFC tag in our application requires frequent write operations. Hence, the write endurance affects the lifetime of CTags. Typical Write endurance (cycles) is given below

Tag	**Write Endurance (Cycles)**
Mifare Ultralight / Mifare Ultralight C	10,000
Mifare Classic	100,000

Using the write endurances, we can estimate the lifetime of CTag by using the following formula

$$L = \frac{E}{N_{avg}}$$

Where L is the lifetime (days) of CTag, E is the write endurance, N_{avg} is the average like / dislike rate per CTag per day. The more popular the CTag, the lesser the lifetime. By giving the popularity, we can estimate the lifetime of the CTag.

Popularity	Mifare Ultralight Lifetime (days)	Mifare Classic (days)
5	2,000	20,000
10	1,000	10,000
25	400	4,000
50	200	2,000
100	100	1,000
150	67	667
200	50	500
500	20	200

Since CTags are distributed around the campus, short lifetime may cause frequent replacement. To illustrate this scenario, we can calculate the failure rate and the mean time between failures (MTBF).

$$\mathrm{MTBF} = \frac{1}{\sum_{i=1}^{n} \lambda_i}$$

Where,

$$\lambda_i = \frac{1}{L}$$

For $i = 1,2,3,\ldots,n$

Where n is the number of deployed CTags and λ_i is the failure rate.

Popularity	Deployed CTags	MTBF of Mifare Ultralight (days)	MTBF of Mifare Classic (days)
10	20	50	500
10	50	20	200
50	20	10	100
50	50	4	40
100	20	5	50
100	50	2	2

Replacement of CTag is frequent if we use NFC tag with write endurance 10,000 cycles. For NFC tag such as Mifare Classic and Mifare Plus, they have reasonable replacement time.

9 Future Works

9.1 Comment in Offline

Cucom allows user to like and dislike without accessing the internet. However, if user wants to leave a comment to the location, they still have to connect to the internet. To make use of the interactivity of NFC, Cucom can be further improved by letting user to comment offline.

Up to now, CTag only stores the metadata and voting results of the CTag location. We can make use of the remaining space, to create a buffer. When a user cannot access the internet but willing to leave comment, he can submit his comment and add it to the buffer. In the next time, when another user scan that CTag, and the user can access the internet, then the user will help uploading the buffer to the webserver.

9.2 Enhance User Interface

To make the application attractive, we have focused on the user experience. But Cucom is lack of customization. User cannot customize their own theme and their own gestures. In future, we can add set of themes and characters for selection. Also, we can add motion gesture to do some critical action, such as like and dislike. For instance, User can give like by shaking or rotating their phone.

9.3 Customizable Features

Giving likes and writing comments may sound fun but not beneficial to user. CTag uses metadata to control operations. After deployment, different departments in The Chinese University can add features to the CTag depending on the usage, location and time. For example, user can book a classroom by scanning the CTag of the classroom. It is also possible to add more than one application on the CTag.

Group game like treasure hunt, can make use of CTag to create checkpoint. Since all the CTags are already placed around the campus, organizer does not need to prepare checkpoints with lots of gears. Besides, CTags are securely embedded on buildings or objects, checkpoints are not easy to affect.

9.4 Profiling

It could be easy if people who intended to give bias comments since no tracking module is in the application. Making profiles can solve this problem. We can also restrict the liking action on CTag using profiles.

On the other hand, after we add customizable features (as discussed above). Service like booking career talk can be more personalized. User do not need to login to CWEM every time. By combining the profile with secure element, it is possible to use CTag to mark attendants.

10 Conclusion

NFC is a new technology that will gradually grow to the market. It is our honored to have the chance to develop application using NFC. In this project, we have done intensive research regarding NFC. During the research, we not only learned the usage of NFC but also the fundamentals of electronics.

This is our first time to develop a mobile app in Android and we learned a lot through the process of development in our FYP. We have self-studied and put a lot of effort on studying the NFC and android SDK. From brainstorming and designing to implementation on the idea, we have experienced the convenience brought by NFC Technology. Although we encountered several problems during the development, we have found the solution for it.

In future, we will add more interesting features on Cucom to make the app more attractive. We hope that finally Cucom can be deployed in the Chinese University of Hong Kong.

11 Reference

[1] L. A. Perlow, Sleeping with Your Smartphone: How to Break the 24/7 Habit and Change the Way You Work, Harvard Business Review Press, 2012.

[2] RapidNFC Ltd., "NFC Tags vs QR Codes - How to Make the Right Choice," 3 May 2013. [Online]. Available: http://rapidnfc.com/blog/73/nfc_tags_vs_qr_codes_how_to_make_right_choice.

[3] Diego López-de-Ipiña,Juan Ignacio Vazquez,Iker Jamardo, "Touch Computing: Simplifying Human to Environment Interaction through NFC Technology," 2007.

[4] M. Egan, "What is NFC? How does NFC work? For what might you use NFC? - a quick guide to NFC," 9 October 2013. [Online]. Available: http://www.pcadvisor.co.uk/how-to/mobile-phone/3472879/what-is-nfc-how-nfc-works-what-it-does/.

[5] Vedat Coskun, Kerem Ok, Busra Ozdenizci, Professional NFC Application Development for Android, Wrox, 2013.

[6] "Near field communication," 18 11 2013. [Online]. Available: http://en.wikipedia.org/wiki/Near_field_communication.

[7] J. Kim, "User-Generated Content (UGC) revolution?: Critique of the promise of YouTube," 2010.

[8] R. Want, "Near Field Communication," *Pervasive Computing, IEEE ,* pp. 4-7, 2011.

[9] M. P. Marie Reveilhac, "Promising Secure Element Alternatives for NFC," 2009.

[10] V. Coskun, K. Ok and B. Ozdenizci, "A Survey on Near Field Communication (NFC)," *Wireless Pers Commun,* pp. 2259-2294, 2012.

[11] G. Madlmayr, J. Langer, C. Kantner and J. Scharinger, "NFC Devices: Security and Privacy," *The Third International Conference on Availability, Reliability and Security,* pp. 642-647, 2008.

[12] ECMA International, "Near Field Communiction Wired Interface (NFC-WI)," *ECMA International,* pp. ECMA-373, 2012.

[13] European Telecommunications Standards Institute, "Host Controller

Interface (HCI)," *ETSI TS 102 622 Technical Specification,* 2008.

[14] T. Frederick O. R. Miesterfeld, N. John M. McCambridge, R. Ronald E. Fassnacht and W. a. o. M. Jerr M. Nasiadka.United State of America Patent 4739323, 1986.

[15] "Serial Peripheral Interface Bus," 9 9 2013. [Online]. Available: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

[16] NFC Forum, "Smart Posters," 2011.

[17] M. C. PhD and A. Nagy, "New Applications for NFC Devices," *Mobile and Wireless Communications Summit, 2007. 16th IST,* pp. 1-5, 2007.

[18] K. OK, V. COSKUN, M. N. AYDIN and B. OZDENIZCI, "Current Benefits and Future Directions of NFC Services," *International Conference on Education and Management Technology（ICEMT 2010）*, pp. 334-338, 2010.

[19] M. Roland, U. A. U. o. A. S. A. NFC Res. Lab. Hagenberg and J. Langer, "Digital Signature Records for the NFC Data Exchange Format," *Near Field Communication (NFC), 2010 Second International Workshop on,* pp. 71-76, 2010.

[20] "User experience," 8 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/User_experience.

[21] "ISO_9241," 15 October 2013. [Online]. Available: http://en.wikipedia.org/wiki/ISO_9241#ISO_9241-210.

[22] S. Borsci, M. Kurosu, S. Federici and M. L. Mele, "Steps of UX: From The Expectations Of The Users Before Product Purchase To The Final Impression Of The Product," in *Computer Systems Experiences of Users with and without Disabilities*, 2013, p. 52.

[23] S. Borsci, M. Kurosu, S. Federici and M. L. Mele, "User Experience," in *Computer Systems Experiences of Users with and without Disabilities*, pp. 49-52.

[24] A. I. Wasserman, "Software Engineering Issues for Mobile Application Development," [Online]. Available: http://works.bepress.com/cgi/viewcontent.cgi?article=1003&context=tony_wasserman.

[25] M. Negulescu, J. Ruiz, Y. Li and E. Lank, "Tap, Swipe, or Move: Attentional Demands for Distracted Smartphone Input," *AVI '12 Proceedings of the*

International Working Conference on Advanced Visual Interfaces, pp. 173-180, 2012.

[26] "Classification," 14 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/Classification.

[27] "Categorization," 2 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/Categorization.

[28] J. Melzer, "Classification, Tagging & Search," 28 August 2007. [Online]. Available: http://www.slideshare.net/jamesmelzer/search-taxonomy-tagging.

[29] "Taxonomy (general)," 25 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/Taxonomy_(general).

[30] S. Dumais and H. Chert, "Hierarchical classification of Web content," pp. 256-263, 2000.

[31] A. D. GORDON, "A Review of Hierarchical Classification," pp. 119-137, 1987.

[32] "A Survey of Hierarchical Classification," pp. 1-41, 2010.

[33] "Hierarchical classification," [Online]. Available: http://www.quantum3.co.za/CI%20Glossary.htm.

[34] K. JONES, "Tags vs. Hierarchy," 16 March 2009. [Online]. Available: http://engagedlearning.net/post/tags-vs-hierarchy/.

[35] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC)," *Workshop on RFID Security RFIDSec. 2006,* 2006.

[36] Atmel Corporation, "Atmel ATMEGA324A-PU Datasheet," Atmel Corporation, 2010.

[37] Olimex Ltd, "AVR-ISP-MK2 Programmer User Manual," Olimex Ltd, 2013.

[38] NXP Semiconductors, "MF1S503x Product data sheet," NXP Semiconductors, 2011.

[39] SHENZHEN TECHSTAR ELECTRONICS CO.,LTD, "TS1620-1 LCD Module Specification Ver1.0," SHENZHEN TECHSTAR ELECTRONICS CO.,LTD.

[40] International Organization for Standardization, "ISO/IEC JTC1/SC17 Cards and personal identification," International Organization for Standardization, 2007.

[41] NXP Semiconductors, "PN532 C106 Application Note," NXP Semiconductors, 2010.

[42] NXP Semiconductors, "PN532 User Manual," NXP Semiconductors, 2007.

[43] NXP Semiconductors, "PN532/C1 Near Field Communication (NFC) controller," NXP Semiconductors, 2012.

[44] NXP Semiconductors, "PN532/C1 Near Field Communication (NFC) Controller Product Data Sheet," NXP Semiconductors, 2007.

[45] EBVElektronik, "RFID Selection Guide," EBVElektronik, 2010.

[46] Motorola, Inc., "SPI Block Guide V03.06," Motorola, Inc., 2003.

[47] J. Lin and H. Shahnasser, "NFC Application Interface for Smart Phones and Appliances," *The Third International Conference on Digital Information Processing and Communications* , pp. 337-340, 2013.

[48] NFC Forum, "NFC Activity Specification," NFC Forum, 2010.

[49] NXP Semiconductors, "MIFARE Ultralight contactless single-ticket IC Product data sheet," NXP Semiconductors, 2010.

[50] NXP Semiconductors, "MF0ICU2 MIFARE Ultralight C Product short data sheet," NXP Semiconductors, 2009.

[51] NFC Forum, "Smart Poster Record Type Definition Technical Specification," NFC Forum, 2006.

[52] NFC Forum, "NFC Analog Specification," NFC Forum, 2012.

[53] NFC Forum, "Logical Link Control Protocol Technical Specification," NFC Forum, 2011.

[54] NFC Forum, "NFC Controller Interface(NCI) Specification," NFC Forum, 2012.

[55] NFC Forum, "NFC Data Exchange Format (NDEF) Technical Specification," NFC Forum, 2006.

[56] NFC Forum, "NFC Record Type Definition (RTD) Technical Specification," NFC Forum, 2006.

[57] NFC Forum, "Text Record Type Definition Technical Specification," NFC Forum, 2006.

[58] NFC Forum, "URI Record Type Definition Technical Specification," NFC

Forum, 2006.

[59] NFC Forum, "Signature Record Type Definition Technical Specification," NFC Forum, 2010.

[60] NFC Forum, "Simple NDEF Exchange Protocol Technical Specification," NFC Forum, 2013.

[61] NFC Forum, "Type 1 Tag Operation Specification Technical Specification," NFC Forum, 2011.

[62] NFC Forum, "Type 2 Tag Operation Specification Technical Specification," NFC Forum, 2011.

[63] C. Nabavi, "Product Reliability," © 2006 PCE Systems Ltd, 2006.

[64] AndroidHive, "How to connect Android with PHP, MYSQL," 2 May 2012. [Online]. Available: http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/.

[65] "Draw 9-patch," [Online]. Available: http://developer.android.com/tools/help/draw9patch.html.

[66] "How to Decide Between SQLite and MySQL," [Online]. Available: http://www.sobbayi.com/blog/software-development/decide-sqlite-mysql/.

12 ACKNOWLEDGEMENT

We would like to express the deepest appreciation to our supervisor, Professor Lyu Rung Tsong Michael, for his guidance and help. He persistantly gives gorgeous advice to our ideas. Without his continual help Cucom would not have been possible.

In addition, we would like to thank Mr. Edward Yau for his help. He provides us with reliable facilities and devices. Components of Cucom are based on these facilities.