STUDENT ASSISTANCE SYSTEM BY CU LINK CARD

LYU1501 Final Year Project Report

Ho Pak Lam (1155034644)

Supervised by Prof. LYU Rung Tsong Michael

Abstract

Android application development together with the use of the NFC technology is the focus of this project to enrich course related activities. Homework coupon system is developed and deployed in a real case situation, which is course CSCI3100.

Homework coupon system with CU Link card allows teachers giving homework coupon to students during class. After class, students could view, use and exchange their coupons. Tutor could check coupons when marking student's homework. All of the above functions are done in smartphone. Moreover, teachers may trace each coupon transaction and view coupon detail information at any time.

Android applications are developed for teachers and tutors. For student, both Android and iOS application are developed. They tap their CU Link card (NFC card) on the Android device. The card is used as a token to recognize a user. A kiosk is also produced for students who do not have a NFC-equipped smartphone. Users can use the app to scan a QR code, which could be get at the kiosk by tapping the student card. Apart from the smartphone applications, there is a server which contains a database to store the information.

Application development, database design, schema, and implementation are discussed in this project. Deployment details and related statistics are also discussed.

Contents

1. Introduction

1.1 Motivation

Smart card makes daily life easier. Smart cards that using NFC/ RFID technology like Octopus cards, student cards are indispensable products which could optimize user experience by just tapping the card on a card reader.

The Chinese University also used NFC card called CU Link card. However, the card is not well utilized. Many activities still rely on traditional pattern. For example, when we enroll in some open seminars, we still need to input personal information on every activity. It is believed that the procedures could be simplified by tapping the student card to recognize a user, which could save time and effort. Increasing the usage rate of a NFC card, introducing more applications on NFC, could optimize the usage of NFC.

1.2 Objective

Our objective is to create an Android application which the CU Link card could be used as a token to enrich course related activities.

To visualize our system effectively, we decided to create a "virtual homework coupon" system, in order to show the functions which could be used in the app. The system will be discussed in the following section.

1.3 Background

In our campus, CU link card are commonly used in functions including attendance taking, book borrowing in libraries and account top-up. Yet, a NFC card could be used more effectively. Simplifying procedures and functions by tapping the student card rather than inputting personal information every time could save time and effort. Increasing the usage rate of a NFC card, introducing more applications on NFC, could optimize the usage of NFC.

Specifically, there exist some courses like CSCI3100 which gives out “homework coupon” to motivate the interaction between student and teacher. Students can get a coupon if they answered some questions well during lesson. By using the “coupons”, student could be exempted in answering some questions in the homework assignment. Students are also allowed to give the coupons they own to other students like friends if they want. Physical coupons are given in the current situation. Our system could give out virtual coupons, which introduces more functions, for example, teacher can track who have given the coupon to other students.

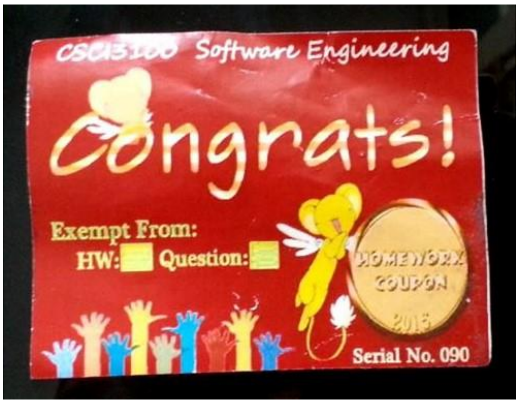


Figure 1 A sample of “homework coupons” used in course CSCI3100 last year.

2. Technical Support and Preliminary study

2.1 Development Environment

As an Android phone application with NFC function is going to be build, we use Android studio with Java. In second semester, we deployed our project on a real scenario, which is the CSCI3100 course. Therefore, we applied a method so that student without NFC supported smartphone could also use our app to check the coupons. Student could tap the card on a kiosk, and then get a QR code. They can then scan the QR code and get the result. By deploying the QR code kiosk, Android users and iOS users can also use this system to view and use their homework coupon. For iOS development, we use Xcode with Swift.

2.2 Programming Software Tool

Android Studio

Android studio is chosen instead of Eclipse ADT. The major reason is that Android Studio is the official IDE. Comparing with Eclipse, Android Studio has a better integration with apps development. Android Studio supports Gradle, while Eclipse require more plugin for that. There are a lot of steps from coding until an APK is produced. Gradle helps developer build the project in one click in Android Studio. Moreover, developer only need to "edit the build the files at the module level" [1], according to Android Studio. Suppose we want to change the compiling version, editing the plain text in the Gradle build files works.

Figure 2 Icon of Android Studio

During Android development, we have set the minimum SDK to API 15: Android 4.0.4 (IceCreamSandwich). According to Android studio, 94% of smartphone use this API or above, which could strike a balance between the number of smartphones could be used and technology advancement. Although setting the minimum SDK at a lower level allows even more device to use, but there may be differences in API calls and hence increase development difficulties. However, if we set the minimum SDK too high, most of the devices could not use our application.

We are using Android Studio 1.4.0, which is the latest version available at the time we start to develop our application.

Android Studio comes along with its simulator, but instead we directly connect our Android device to do debugging and testing. The major reason is that NFC function could not be used with the simulator.

Xcode

Xcode is the only choice for developing iOS application. Users can use Objective-C or Swift to develop the app. Swift is a programming language which is easy to catch up. Pointers are not available in Swift and thus eliminate many referencing errors. The app is developed on XCode 7.2.1 on OS X El Capitan, which is the latest version. The deployment target is iOS 9.0.

Figure 3 Logo of Swift

2.3 Database Management Software Tool: phpMyAdmin

This homework coupon system stores the coupons in a database rather than saving it into the student CU Link card or other NFC cards, as memory in the NFC card is very limited and we need to keep the states of each coupon up-to-date. As a result, a MySQL database with phpMyAdmin is used.

Figure 4 Icon of phpMyAdmin

We have created tables inside the database to store information of classes, tutors of different classes, coupons and coupon transaction logs.

phpMyAdmin is used for creating new table, managing database structure and making modification on database record during the testing and debugging stage.

We have chosen to use phpMyAdmin as it has a GUI which is easy to understand and use. Moreover, its web interface makes it work well in different platforms, even for a browser of a smartphone. Moreover, there are “query-by-example” [2] which could be used to create complex queries.

We are using phpMyAdmin version 4.0.8. The environment is provided by the VIEW Lab.

2.4 NFC Technologies

2.4.1 NFC Operation Mode

According to NFC forum, NFC has 3 operation mode:

The first one is reader/writer mode (ISO 14443, FeliCa standard) [3]. The NFC device, smartphone in our case, could read the data of NFC tag, like the ID of the tag The NFC tag is passive, the active device which is the smartphone initiate the action to read or write data from/ to the tag. Our project uses this mode the get the tag ID as a token for identifying the person.

The remaining two modes are Peer-to-Peer mode, and Card Emulation mode. For the former one, a peer is a smartphone device. In this mode, two smartphones can touch each other for transferring data, just like in Wi-Fi or Bluetooth. For the latter one, the NFC reader creates EM field to get the information of a NFC device. In another words, the NFC device acts like a smart card. This operation mode used in mobile payment.

2.4.2 NFC technologies

From the final year project from year 2013, they have summarized some differences between different types of NFC technologies and tag types. No matter the MIFARE system which used by the CU Link card, or the FeliCa which used by Octopus card in Hong Kong, they have similar usage with different memory size and data rate. [4]

In our project, the UID of the NFC card (CU Link card) is used as a token to recognize a person. The UID is a string attached to the card which are different among CU Link cards.

Worth to note that, there is a protocol called “NFC-A (ISO/IEC 14443 type A) protocol anti-collision and activation” [5]. The major difference of this type of protocol with others is that on every time the card tap the reader, a randomly generated UID is shown on the reader. In this case, the UID could not be used as a token for identifying a user. Both CU Link card and Octopus card is not using this strategy.

3. System Architecture

3.1 Overall Architecture

The system is divided into three parts: server, client (application), and kiosk.

The server is responsible for storing data in a database. The data in database is always retrieved by the client or kiosk.

A client can use the mobile app to do tasks like reading the NFC card. All the NFC manipulation are done in the app. The UID of the NFC tag is then sent to the server to get related data.

Moreover, since iOS (client) do not have NFC function, we need to rely on the kiosk. The UID of the NFC card is read in the kiosk and sent to server. A QR code is then generated. QR code is scanned by the iOS device so that the iOS device can communicate with the server.

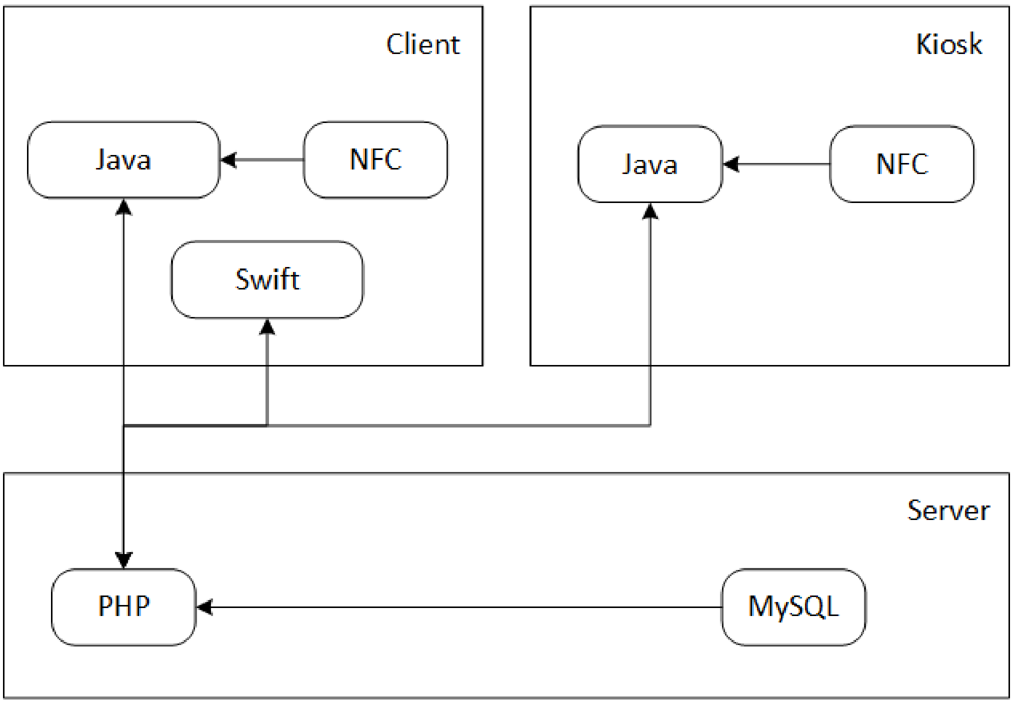


Figure 5 Simple graph showing the overall architecture of the system

3.2 System Overview

As mentioned in the first section, we use a “virtual homework coupon” application as an example to explain the function available in the application. For the sake of communication, we will name the system that currently using, that means giving out “physical homework coupon”, as the *traditional model*.

This system is divided in to three parts, namely “Teacher side”, “Tutor side” and “Student side”.

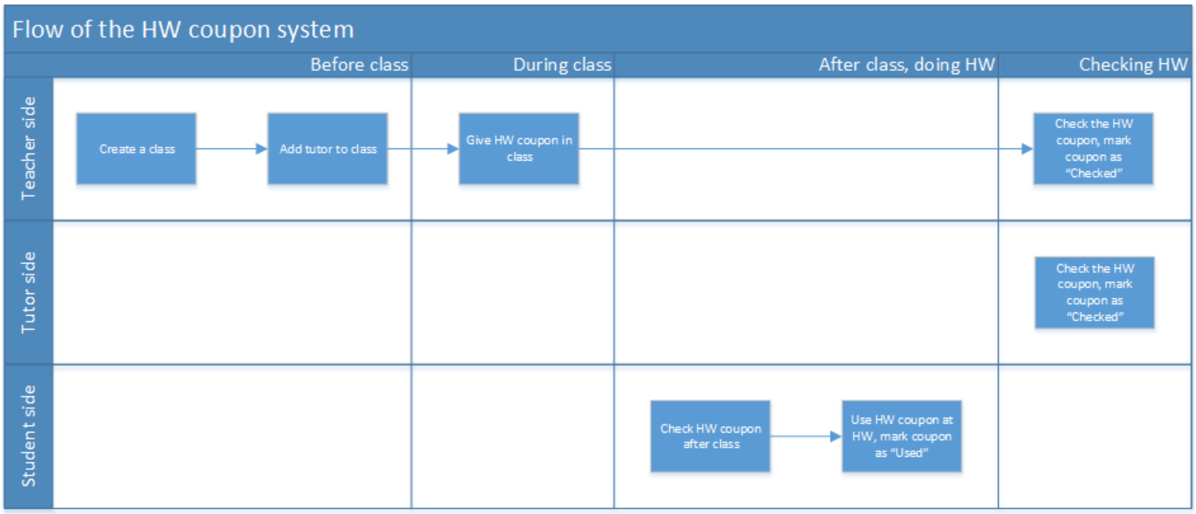

Flow of the HW coupon system
Before class
During class
After class, doing HW
Checking HW
Teacher side
Tutor side
Student side
Create a class
Add tutor to class
Give HW coupon in class
Check the HW coupon, mark coupon as “Checked”
Check the HW coupon, mark coupon as “Checked”
Check HW coupon after class
Use HW coupon at HW, mark coupon as “Used”

Figure 6 The flow of homework coupon system

1. First of all, the “Teacher” (Instructor) need to “Create a class”. For example, there is a class called “CSCI3100 Software Development”. By inputting data to the app, the app

will submit the information to the server. Teacher's NFC card is scanned as a token to identify a particular teacher created this class.

2. Tutor corrects homework. Teacher can "Add tutor to class", this step is optional. By scanning teacher's card and tutor's card, we can know that a particular tutor belongs to a particular course/ class (say CSCI3100).
3. During lesson, if a student answered a question well, teacher may consider giving out a "homework coupon". In the *traditional model*, a physical homework coupon is given out. But using this Android application, Teacher taps his/ her NFC card, and then student taps his/ her own card on the same device. The transaction is finished.
4. The student can check his/ her own coupons, by tapping his/ her own card onto the smartphone, e.g. how many coupons he/ she owns, at what time and date he/ she got the coupon, etc. On the other hand, teacher could also view how many coupons he/ she has given out, whether the coupon is used or not, etc.
5. The student can also give coupon to other people.
6. The student can mark the coupon as "used", indicating that the student wants to use this coupon in the homework for exempting answering some questions. The uniqueness of a coupon is indicated by a string. In the *traditional model*, student clips the coupon into the homework, but now, student need to write the code (string) of the coupon.
7. Tutor or teacher could mark the coupon as "checked", that mean it is consumed, this action could not be undo. Tutor/ teacher input the string given by the student, and then scan teacher/ tutor's NFC card that had previously registered in step 1 & 2.

There are some advantages.

1. The instructor can check who has transferred the coupon to other users.
2. The student can check whether the coupon is successfully consumed. In other words, if the tutor/ teacher has received the coupon.

We could generalize the above event. It means that those functions could also be applied on different event. For example, as course attendance taking. Similar with the above example, the teacher need to create a class. The process of "giving homework coupons" now becomes taking attendance. Teacher's CU Link card and student's CU Link card is tapped. The "coupon" generated is an indicator that the student has attended the lesson. On the student side, the student can tap his/ her own CU Link card to check the "coupons", that mean if the attendance is recorded successfully.

Some students do not have an Android phone with NFC function, one of the works done in semester two is to cater the needs of these users so that they can check the coupons on their own smartphone.

1. Instead tapping their own card to the smartphone, a kiosk is produced and put at SHB 1/F. Students can go to the kiosk and tap their card to the kiosk. The kiosk then produces a QR code.

2. Student then scan the QR code using our app “CU Homework Koupon”. The details of coupons are listed out.

Every QR code could be used for once only. The code will be invalid after used or 15 minutes after generated.

In short, student with NFC enabled Android phone could directly tap their student card to their own smartphone, while students without NFC enabled Android phone could tap the card to the kiosk and scan QR code with the app installed in their smartphone.

3.3 Activity Diagram

Diagrams are used to shows the activity flow of different functions in brief.

Create class/ event

To create class/ event, user need to type the class information into the application first, and then tap teacher’s NFC card. After scanning the UID of the card, the information of the class will be sent to database and saved.

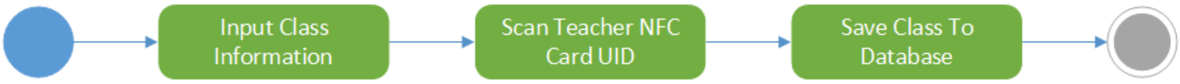


Figure 7 Activity diagram showing the flow of adding a class/ event

Add tutor

To add tutor, user need to scan the teacher NFC card first and the application will read the UID of the card. If the UID is not found in the database, the application will guide the user to “create new class” screen. If the UID is found in the database, the application will display all the classes created by the teacher NFC card in a list. After selecting the class, the application will ask user to tap the tutor NFC card. After scanning the UID of the NFC card, the information about the tutor and class is updated to the database.

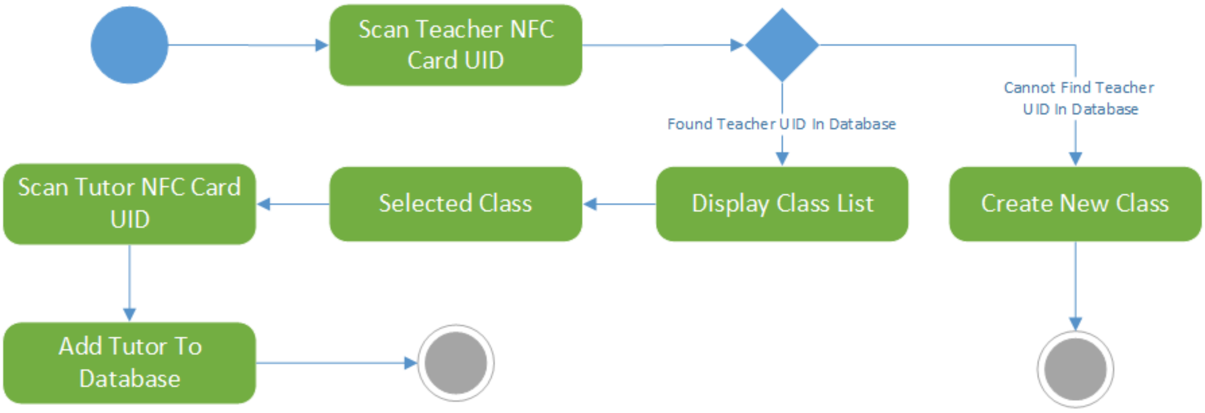


Figure 8 Activity diagram showing the flow of adding a tutor

Give out coupons

To give coupon, teachers need to tap their teacher NFC card. After scanning the UID of the card, the UID will be sent to server. If the UID is not found in the database, the application will guide user to “create new class” screen. If the UID is found in the database, the application will display all the classes created by the teacher NFC card in a list. After selecting the class, the application will ask user to tap the student NFC card. After scanning the UID of the NFC card, coupon will be created and stored in database.

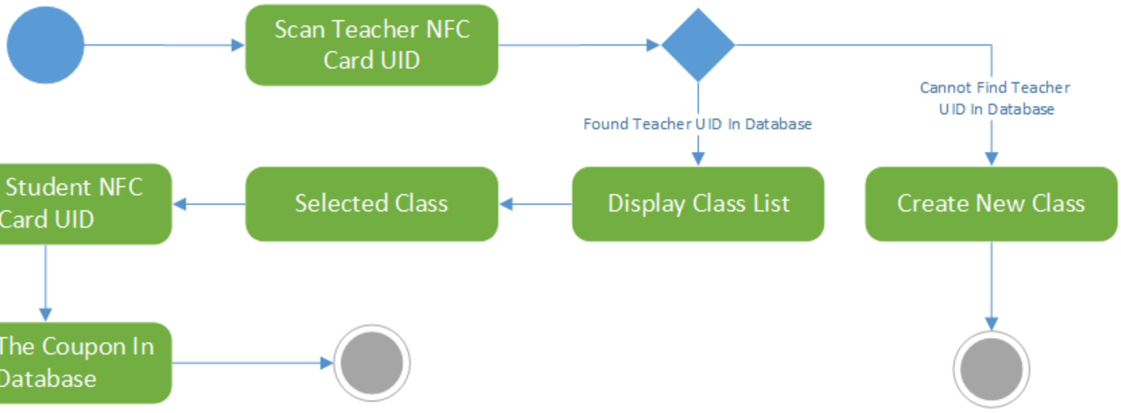

Scan Teacher NFC Card UID
Found Teacher UID In Database
Cannot Find Teacher UID In Database
Scan Student NFC Card UID
Selected Class
Display Class List
Create New Class
Save The Coupon In Database

Figure 9 Activity diagram showing the flow of giving out coupons

Transfer coupons between users

To make transaction on coupons, student need to scan their student NFC card first. After scanning the UID of the card, the application will display all the coupons that the student got in a list. Student need to select which coupon he/ she want to transact. The application will display the detail information of the coupon. Student needs to press “exchange coupon” and scan the student NFC card. Then, the receiver’s student NFC card needs to be scanned. If the coupon belongs to the sender and the coupon is not mark as used or checked, the new coupon owner and coupon ID will be saved in the database, else, error message will be shown.

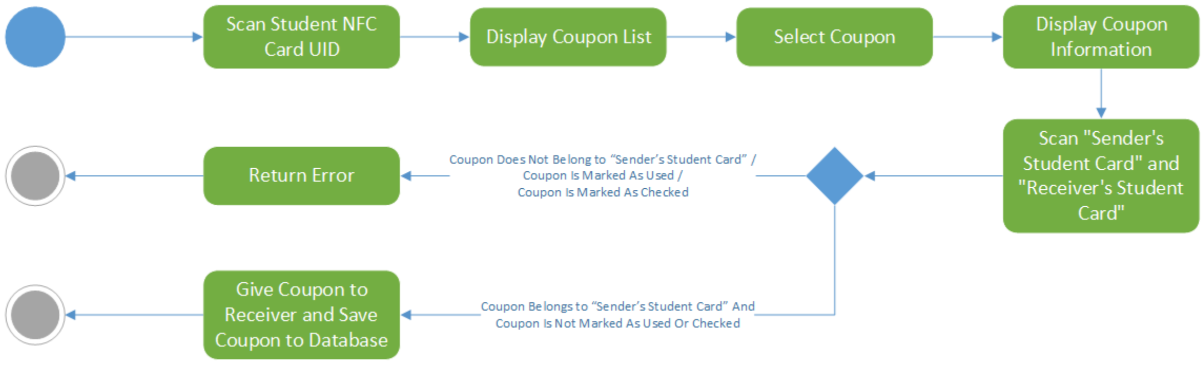

Scan Student NFC Card UID
Display Coupon List
Select Coupon
Display Coupon Information
Scan "Sender's Student Card" and "Receiver's Student Card"
Coupon Does Not Belong to “Sender’s Student Card” / Coupon Is Marked As Used / Coupon Is Marked As Checked
Return Error
Coupon Belongs to “Sender’s Student Card” And Coupon Is Not Marked As Used Or Checked
Give Coupon to Receiver and Save Coupon to Database

Figure 10 Activity diagram showing the flow of transferring coupon between users

View and Check Coupons with NFC enabled smartphone

First, type the coupon ID to the application. The detail of that coupon will be shown. Then tap the teacher / tutor NFC card. If the UID of the card is found in the database and the card is in the teacher / tutor list of the class, the coupon will be marked as check in the database, else, error message will be shown.

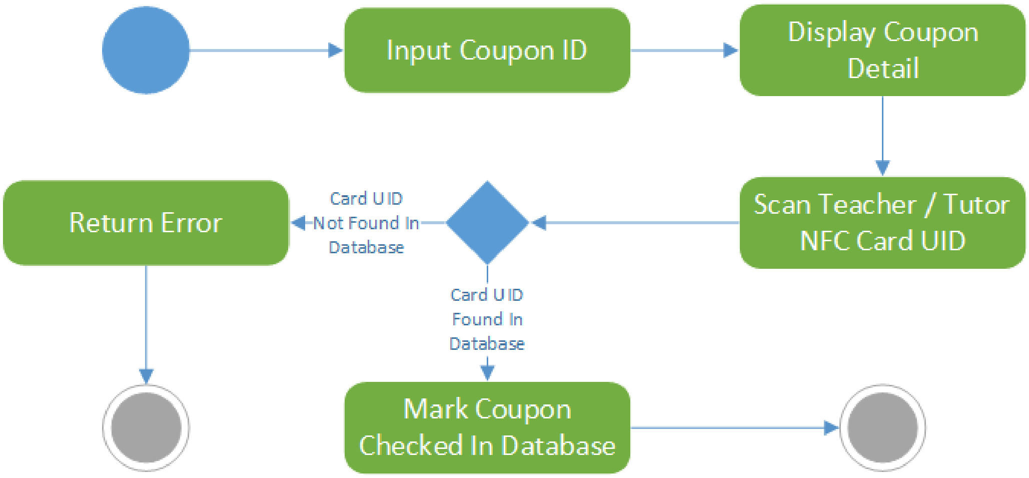

Input Coupon ID
Display Coupon Detail
Scan Teacher / Tutor NFC Card UID
Card UID Not Found In Database
Return Error
Card UID Found In Database
Mark Coupon Checked In Database

Figure 11 Activity diagram showing the flow viewing and checking coupons.

Generate and Use QR code to view coupons

To generate the QR code, first, the student card UID is scan at the kiosk. Then a temporary ID is generated at server. Such information in stored in the database. The temporary ID is then converted in to a QR code and displayed to the user at the kiosk.

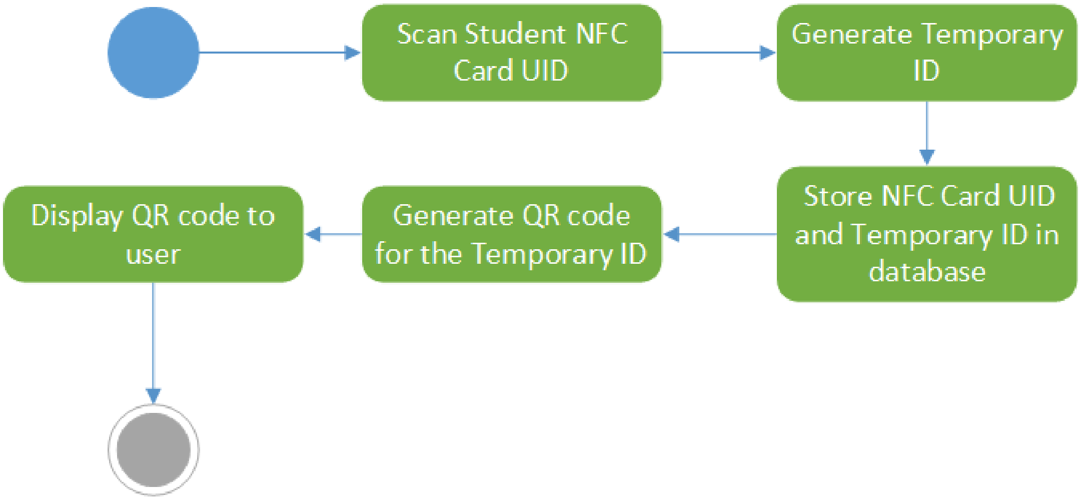

Scan Student NFC Card UID
Generate Temporary ID
Store NFC Card UID and Temporary ID in database
Generate QR code for the Temporary ID
Display QR code to user

Figure 12 Activity diagram showing the flow of generating QR code

The user can scan the QR code using his/ her smartphone. The app then sends the temporary ID to server. If it is found, the temporary ID will be deleted from the database, then the student card UID is returned to the smartphone. Else, an error will be shown.

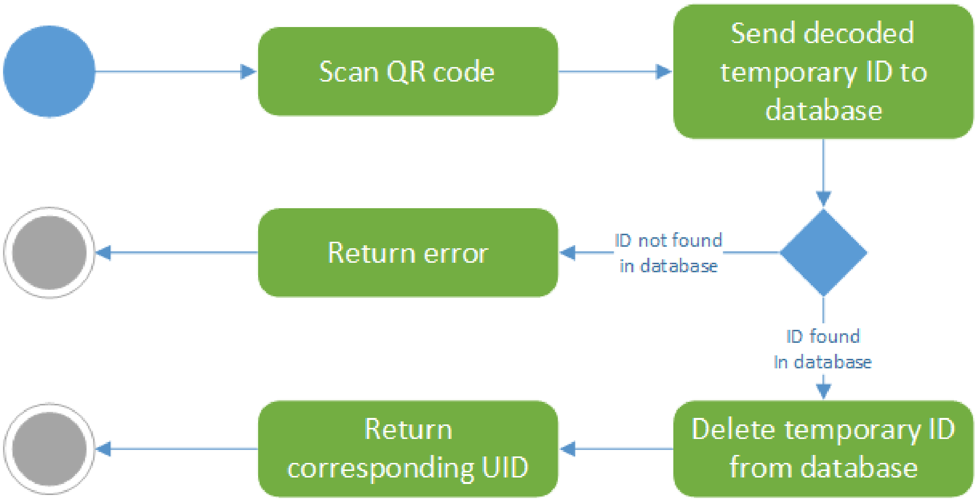


Figure 13 Activity diagram showing the flow of using QR code

4. Design and Implementation

4.1 Server Side

4.1.1 Database: ER Diagram

The graph below shows the structure of our database.

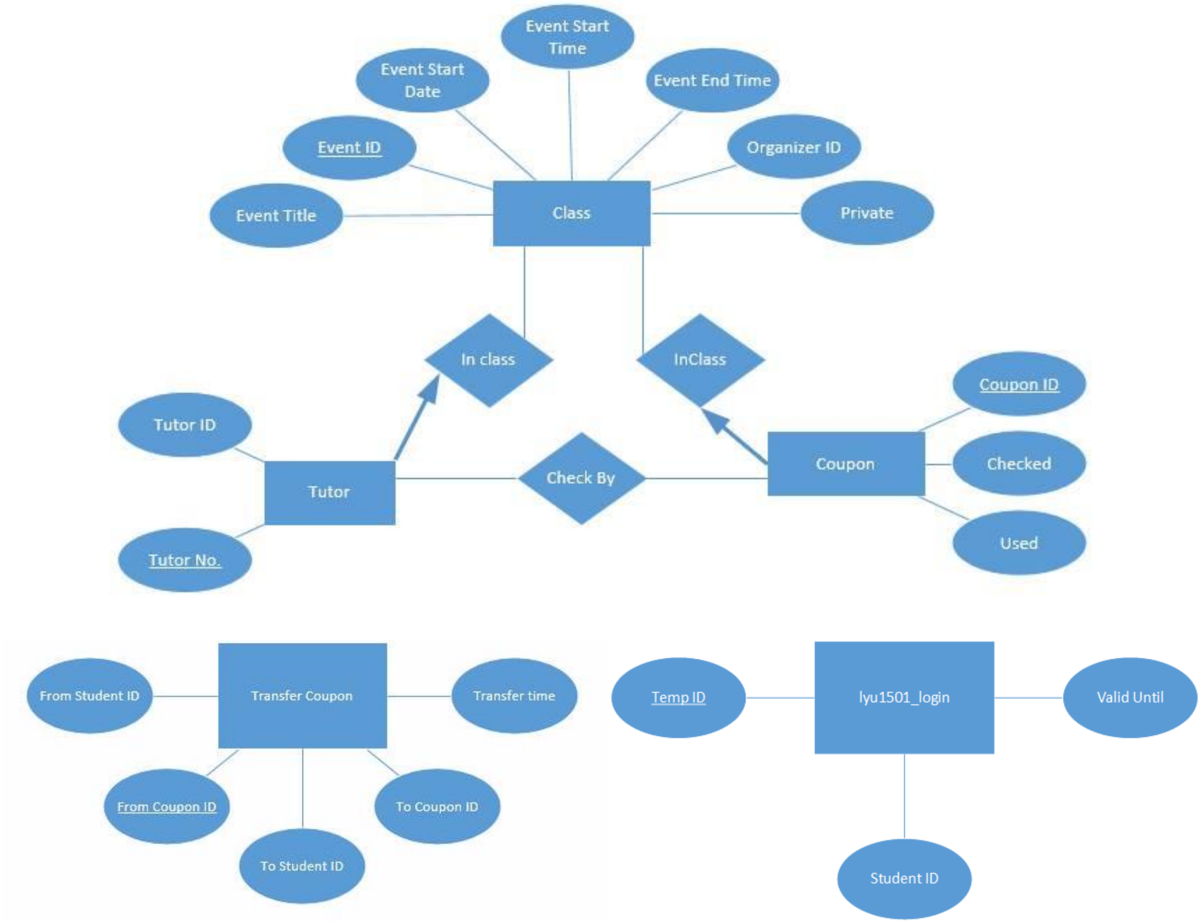


Figure 14 ER diagram showing the structure of our database.

The database is used to store data including the class information, coupon information, etc., as mentioned in the previous section. The server offers a large memory size, comparing to the Android smartphone, so the server is capable to handle an increasing number of records. The server is also used to synchronize information with the Android device, user can view the most updated information once the device is connected to the internet. Related information will be retrieved from the database. Another reason is security issue. By putting the data in server, user could only retrieve the part that is related to the user.

4.1.2 Database: Schema

The schema of the database is shown below:

```
class(eventID: integer, eventTitle: text, eventDate: date,
       eventStartTime: time, eventEndTime: time, eventDetail: text,
       organizerID: varchar: private: int)

coupon(couponID: varchar, eventID: integer,
       remarks: text, used: int, checked: int, checkBy: varchar)

trancou(fromCouponID: varchar, toCouponID: varchar, tranTime: datetime)

belong(couponID: varchar, studentID: varchar)

tutor(tutorNo: int, eventID: int, tutorID: varchar)

login(studentID: varchar, tempID: varchar, validUntil: timestamp)
```

Class Table

Each tuple represents a uniquely identified class/ event. `CouponID` is the primary key. When a user creates a new class/ event, the user can choose whether the event is a private one or not. “Private event” means the event is not searchable. “Open event” means it is a searchable class. For example, “homework coupon” system is a private event, because we do not want student beyond the specific class to gain access to this event and earn coupons freely. Furthermore, as mentioned above, this system could be applied on various occasions, suppose there is an open seminar hold by the university, everyone can join if they are interested, this will be an “open event”.

Attribute	**Description**	**Data Type**
eventID	The ID of the class/ event, unique	Integer (Auto increment)
eventTitle	The title of the class/ event	String
eventDate	Specify the date of the class/ event, optional field	Date
eventStartTime	Specify the starting time of the class/ event, optional field	Time
eventEndTime	Specify the ending time of the class/ event, optional field	Time
eventDetail	Specify the detail of the class/ event, optional field	String
organizerID	The card UID of the person who create this event	String
private	Declare if this event/ class is a private one. “1” if private, “0” if not private (that means open event)	Integer

Coupon Table

Each tuple represents a uniquely identified coupon. `CouponID` is the primary key. `eventID` is the foreign key (primary key of table `class`).

Attribute	**Description**	**Data Type**
couponID	The ID of the coupon, unique	String
eventID	The ID of the event	Integer (Auto increment)
remarks	Remarks added to the coupon	String
used	Mark '1'(True) if the student want to use this coupon, '0' (False) else.	Integer
checkBy	The card UID of the tutor/ teacher is marked if this coupon is consumed.	String

Trancou (Transfer coupon) table

What means by transfer coupon is that one user can give a coupon to another user. `fromCouponID` is the primary key. Each tuple represents one transaction. To ensure the original owner of coupon cannot use the coupon after the transaction, the coupon ID is regenerated.

Attribute	**Description**	**Data Type**
fromCouponID	The original coupon ID, unique	String
toCouponID	The newly generated coupon ID	String
tranTime	The timestamp of the occurrence time of this transaction.	Date Time

Belong Table

Each tuple represents the relationship between a coupon and a student. `couponID` is the primary key. By joining this table with the previous table "Trancou", we can know the coupon is transferred from who to whom.

Attribute	**Description**	**Data Type**
couponID	The ID of the coupon, unique	String
studentID	The student card UID	String

Tutor table

Each tuple links a tutor with an event/ class. `eventID` is the foreign key (primary key of table `class`). Tutor-Class is a many-to-many relationship, which means one class can have many tutors, one tutor can join many classes.

Attribute	**Description**	**Data Type**
tutorNo	Uniquely identify a tutor in a class.	Integer (Auto increment)
eventID	The ID of the class/ event, uniquely identify a event.	Integer
tutorID	The card UID of the tutor.	String

Login table

Each tuple links a student card number with a temporary ID, with a timestamp. `tempID` is the primary key. If the temporary ID has not been used after the timestamp, usually 15 minutes after the temporary ID generated, it becomes invalid.

Attribute	Description	Data Type
studentID	The student card UID	String
tempID	A QR code is generated from this temporary ID. Smartphones without NFC function can scan the QR code and retrieve their own student card UID.	String
validUntil	The temporary ID is valid on or before this time, usually 15 minutes after the ID is created.	Timestamp

4.1.3 Implementation: PHP, JSON, SQL queries

In our application, PHP act as a communication channel for the Android application to submit queries to the server, and get back the related records.

There are few reasons of using PHP. PHP is a server-side scripting language, the content is hided from the viewpoint of a user, all the database configuration is separated from the Android app. The database is more prone to leakage of private information if the configuration is put inside the app. Moreover, Java and PHP got libraries on JSON. Therefore, using JSON is a relatively straightforward method to retrieve the data.

The PHP contains of a number of modules, namely:

Major group of modules	**Module names**
Configuration	`db_config`
Retrieve record from database	`get_coupon_details, get_coupon_log, get_student_coupon, get_teacher_coupon, view_teacher_class`
Insert record to database	`add_class, add_tutor, give_coupon, exchange_coupon, gen_tempID`
Edit record in database	`mark_used_coupon, mark_checked_coupon, tempID_to_studentID`

Configuration module

This module includes 2 files, `db_config` and `db_connect` are used to connect the database. Here, the part of code for connecting the database in the file `db_connect.php` is shown.

```
function connect() {
    // import database connection variables
    require_once __DIR__ . '/db_config.php';

    // Connecting to mysql database
    $link = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    /* check connection */
    if (mysqli_connect_errno()) {
        printf("Connect failed: %s\n", mysqli_connect_error());
        exit();
    }

    // returing connection cursor
    return $link;
}
```

Retrieve record from database

A lot of data is needed to retrieve from the database, including the coupon details (`get_coupon_details`), coupon transaction log (`get_coupon_log`), number of coupons that a student got (`get_student_coupon`), number of coupons that a teacher given (`get_teacher_coupon`), and number of courses that a teacher created (`view_teacher_class`).

The following code shows an example of getting the records of all the coupons got by a student, which is the code in `get_student_coupon.php`

```php
<?php
// create array for JSON response
$response = array();

if(!empty($_POST['studentID'])){
    $studentID = $_POST['studentID'];

    // include db info
    require_once __DIR__ . '/db_config.php';

    // connecting to db
    $db = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    //get all coupon hold by student in coupon table
    $result = mysqli_prepare($db, "SELECT couponID, eventTitle, used FROM
lyu1501_coupon NATURAL JOIN lyu1501_class WHERE studentID = ?");
    mysqli_stmt_bind_param($result, 's', $studentID);
    mysqli_stmt_execute($result);
    mysqli_stmt_bind_result($result, $couponID, $eventTitle, $used);
    mysqli_stmt_store_result($result);

    if(mysqli_stmt_num_rows($result) > 0){
        //coupon found
        $response["coupon"] = array();

        while (mysqli_stmt_fetch($result)) {
            // temp user array
            $coupon = array();
            $coupon["couponID"] = $couponID;
            $coupon["eventTitle"] = $eventTitle;

            if(strcmp($used , "0") == 0){
                $coupon["used"] = "Unused";
            } else {
                $coupon["used"] = "Used";
            }

            // push single coupon into final response array
            array_push($response["coupon"], $coupon);
        }
        // success
        $response["success"] = 1;

        // echoing JSON response
        echo json_encode($response);
    } else {
        // coupon not found
        $response["success"] = 0;
        $response["message"] = "No coupon found";
        echo json_encode($response);
    }

} else {
    // missing field studentID
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    echo json_encode($response);
}
?>
```

For other parts of the code, they are similar with different SQL queries. Only SQL queries is shown for clarity.

Purpose **(Corresponding PHP file)**	**SQL query**

View the general information of the coupons given by a teacher.
(get_teacher_coupon.php)
	SELECT *
	FROM lyu1501_coupon
	NATURAL JOIN lyu1501_class
	WHERE organizerID = '$teacherID'

View the details of a particular coupon. (get_coupon_details.php)	SELECT *
	FROM lyu1501_coupon
	NATURAL JOIN lyu1501_class
	WHERE couponID = ‘$pid’

View which user gave the coupon to which user (get_coupon_log.php)	SELECT * FROM lyu1501_trancou AS trancou JOIN lyu1501_belong AS from ON from.couponID=trancou.fromCouponID JOIN lyu1501_belong AS to ON to.couponID=trancou.toCouponID WHERE trancou.fromCouponID LIKE '$searchID%' OR trancou.toCouponID LIKE '$searchID%' ORDER BY trancou.toCouponID

View the classes/ events that teacher/ instructor created. (view_teacher_class.php)	SELECT *
	FROM lyu1501_class
	WHERE organizerID LIKE '$teacherID'

Insert Record from Database

We have 4 modules for inserting records. Add a new class/ event (`add_class`), add a tutor into a class/ event (`add_tutor`), giving out coupon by a teacher to a student (`give_coupon`), and transferring coupon between students (`exchange_coupon`).

Purpose **(Corresponding PHP file)**	**SQL query**

Create a new class/ event.
1. Insert a new event into the class table
2. The card UID of teacher himself/ herself is also added to the tutor table, because both teacher and tutors can validate the coupon (mark coupon as "checked")
(`add_class.php`)
	`INSERT INTO lyu1501_class(eventTitle, eventDate, eventStartTime, eventEndTime, eventDetail, organizerID, private)`
	`VALUES ('$eventTitle', '$eventDate', '$eventStartTime', '$eventEndTime', '$eventDetail', '$organizerID', '$isPrivate')`
	
	`INSERT INTO lyu1501_tutor(eventID, tutorID)`
	`VALUES ('$pastEvent', '$organizerID')`

Add a tutor to a class.
1. Search the corresponding eventID from class table
2. Insert the tutor card UID into the tutor
(`add_tutor.php`)
	`SELECT eventID`
	`FROM  lyu1501_class`
	`WHERE  organizerID = '$teacherID'`
	
	`INSERT INTO lyu1501_tutor`
	`(eventID ,tutorID)`
	`VALUES ('$eventID', '$tutorID')`

Giving out coupon by a teacher to a student
1. Add a new coupon
2. The teacher given a coupon to student, so a record is added in the “trancou” table, indicating a coupon transfer action has performed.
(give_coupon.php)
	INSERT INTO lyu1501_coupon(couponID, eventID)
	VALUES('$nextCoupon', '$eventID')
	
	INSERT INTO lyu1501_belong(studentID, couponID)
	VALUES('$studentID', '$nextCoupon')
	
	INSERT INTO lyu1501_trancou (fromCouponID , toCouponID , tranTime)
	VALUES ($logString', '$studentID', '$logString')

Giving a coupon from one student to another.
1. Check if the coupon is not consumed (checked = ‘0’), nor the student is going to use the coupon (used = ‘0’). If the above conditions are satisfied, go to step 2 & 3
2. Update the new coupon owner, and new coupon ID
3. Log the transaction
(exchange_coupon.php)
	SELECT *
	FROM lyu1501_coupon
	NATURAL JOIN lyu1501_belong
	WHERE studentID = '$fromStudentID'
	AND couponID = '$fromCouponID'
	AND checked = '0'
	AND used = '0'
	
	INSERT INTO lyu1501_belong (toStudentID , toCouponID)
	VALUES ('$toStudentID', '$nextCoupon')
	
	INSERT INTO lyu1501_trancou (fromCouponID, toCouponID, tranTime)
	VALUES ('$fromCouponID', '$nextCoupon', '$sqlNowString')

When a student ID is presented to the kiosk, a temporary ID, which will be converted to QR code later, is generated. (`gen_tempID.php`)	`INSERT INTO lyu1501_login('studentID', 'tempID', 'validUntil')`
	`VALUES (‘$studentID’, ‘$fullID’, ‘$validTimeString’)`

Edit Record in database

When the student decided to use the coupon, he/ she can click the button available in the Android app. It composes of 2 modules.

- When the tutor checks the coupon, the tutor can know if the student wants to use that particular coupon. (`mark_used_coupon`)
- The tutor checks the coupon and mark as checked. The coupon is consumed and cannot be undo. (`mark_checked_coupon`)

Moreover, there is a module which gets the student card UID from the QR code. (`tempID_to_studentID`)

Purpose **(Corresponding PHP file)**	**SQL query**

A student indicate that he/ she want to use the coupon.
1. Search the coupon ID.
	`SELECT checked`
	`FROM lyu1501_coupon`
	`WHERE couponID = ‘$couponid’`

2. Toggle the “used” flag. That means, if the student was not going use the coupon, mark that he/ she want to use, and vice versa.
(mark_used_coupon.php)
	(student wants to use the coupon)
	UPDATE lyu1501_coupon
	SET used = '1'
	WHERE couponID = '$couponid'
	
	(student do not want to use the coupon)
	UPDATE lyu1501_coupon
	SET used = '0'
	WHERE couponID = '$couponid'

Mark a coupon as checked/ consumed.
1. Check if the tutor found in the tutor list.
2. If found, consume the coupon.
(mark_checked_coupon.php)
	SELECT *
	FROM lyu1501_tutor
	WHERE eventID = $eventID
	AND tutorID = \"$tid\"
	
	UPDATE lyu1501_coupon
	SET used = '1', checked = '1', checkBy = '$tid'
	WHERE couponID = '$couponid'

Given the QR code (i.e. temporary ID), match it with the student card UID.
1. Search if the temporary ID exists in the database and if it is within the valid time.
2. If found, set the valid time to now, means it cannot be used again anymore.
	SELECT studentID
	FROM lyu1501_login
	WHERE tempID LIKE '$tempID'
	AND validUntil > CURRENT_TIMESTAMP
	
	UPDATE lyu1501_login
	SET validUntil = CURRENT_DATE( )
	WHERE tempID = '$tempID'

(tempID_to_studentID .php)	

Database Normalization

The main purpose of doing normalization is to reduce data redundancy, and inconsistence dependency. [6] If the data are redundant, and update requires multiple queries, the process will be less efficient. If either one record is not updated, the record becomes inconsistent. By performing normalization, data consistency is kept and maintainability is increased. There are different rules regarding normalization of the database. We will analyze our table with different normal forms.

First Normal Form (1NF):

Our database contains atomic values in each attribute. No repeating groups of data is allowed, and each table has a primary key. For example, the schema for the tutor table is:

```
tutor(tutorNo: int, eventID: int, tutorID: varchar)
```

One tutor may join many classes. One class may have many tutors. Every tutor has a “tutor ID” and every class/ event has an “event ID”. The primary key is a tutor number, which uniquely defines the class-tutor pair. It is not allowed that one tutor ID has many event ID in one tuple. For example

```
tutor(tutorNo: int, eventID1: int, eventID2: int, ..., tutorID: varchar)
```

violates 1NF.

Second Normal Form (2NF):

It is in 2NF because it is in 1NF and every non-key attribute depends on all the entire primary key. For every table in our database, there is only one primary key. All the attributes within the table depends on the primary key.

- For the “class” table: eventID is the primary key, and it is a representation of a class/ event. Fields like title, date, time, details, organizerID, depend on the eventID.
- For the “coupon” table: couponID is the primary key, a coupon belongs to a class/event, belongs to one student. Each coupon could have fields like its own remarks, whether it is consumed. All attributes depend on the primary key.
- For the “trancou” table: all data depends on the original coupon ID (primary key), including the newly generated coupon ID, time of having that transaction.
- Finally, for “tutor” table: tutor number, which is the primary key, indicates the event ID and tutor ID.

Potential problem of 2NF: Update anomaly problem. It is known that the set of information “studentID” and “couponID” in the “coupon” table also exists in “toStudentID” and “toCouponID” in the “trancou” table, update anomaly may occur if update are not carried out consistently. [7] Reviewing our SQL queries, such problem should not occur even though our database is not in 3NF (mentioned in the next part) yet, as both table are updated on every transactions.

Third Normal Form (3NF):

The table is in 3NF if it is in 2NF and there are no transitive dependencies. In the first semester, the tables structure is as below.

```
coupon(studentID: varchar, couponID: varchar, eventID: integer,
       remarks: text, used: int, checked: int, checkBy: varchar)

trancou(fromStudentID: varchar, fromCouponID: varchar,
       toStudentID: varchar, toCouponID: varchar, tranTime: datetime)
```

It is discussed that the above table is not in 3NF, because it is possible to record that coupon A belongs to student A, say in the table “coupon”, but in table “trancou” the same coupon A belongs to another person. To comply with 3NF, we need to made the adjustment to remove the attribute “studentID” in table “coupon”, attribute “fromStudentID” and “toStudentID” in table “trancou”. A “belong” table is added which extracts the “couponID” and “studentID” relationship.

```
belong(couponID: varchar, StudentID: varchar);
```

We will show that every non-prime attribute only depends on the candidate key. Candidate key in a table refers the minimal super key.

- For the “class” table: the only candidate key is the “eventID”, that is the primary key we are using. Even the attribute “eventTitle” may clash (i.e. different events may have the same event title). All other attributes only rely on the “eventID” but not on other attributes.
- For the “coupon table”: It is similar with the “class” table, attribute “couponID” is the only candidate key, and all other attributes only rely on that but not others.
- For all other tables: The only candidate key is the primary key, and no other non-prime attribute depends on another non-prime attribute.

Advantage of 3NF: 3NF ensures that the database is free of update, insertion, and deletion anomalies. It is not needed to edit multiple tables regarding the same set of data, and hence eliminate data inconsistencies.

There is trade-off between achieving a higher normalization form and readability. At the time the database is created, in order to comprehend the structure of database effectively, it is decided to use one table called “trancou” to log the basic information of coupon transaction. If the database is in 3NF, multiple joins are required for getting same set of useful result.

Since the table structure will be changed, the corresponding SQL queries should be edited. The following table shows how the tables from semester one reconstructed using the schema in semester two.

Schema in semester one	Schema in semester two
coupon(studentID: varchar, couponID: varchar, eventID: integer, remarks: text, used: int, checked: int, checkBy: varchar) trancou(fromStudentID: varchar, fromCouponID: varchar, toStudentID: varchar, toCouponID: varchar, tranTime: datetime)	coupon(couponID: varchar, eventID: integer, remarks: text, used: int, checked: int, checkBy: varchar) trancou(fromCouponID: varchar, toCouponID: varchar, tranTime: datetime) belong(couponID: varchar, studentID: varchar)

Original SQL Query	New SQL Query

SELECT *
FROM lyu1501_coupon
WHERE *(some conditions)*
	SELECT *
	FROM lyu1501_coupon
	NATURAL JOIN lyu1501_belong
	WHERE *(some conditions)*

SELECT *
FROM lyu1501_trancou
WHERE *(some conditions)*
	SELECT *
	FROM lyu1501_trancou AS trancou
	JOIN lyu1501_belong AS from
	ON
	from.couponID=trancou.fromCouponID
	JOIN lyu1501_belong AS to
	ON to.couponID=trancou.toCouponID
	WHERE *(some conditions)*

Query optimization

Manipulating table joins and selections generate useful results. The speed of executing a SQL query depends on the strategies used in joins and selections. Select useful attributes before perform table join is generally more efficient then performing selection after joins. Such difference in speed is not noticeable when there are a little number of tuples, but become more obvious when the amount of tuple grows.

For example, the following query appeared previously gets the detailed information of a particular coupon a student got. For simplicity, `SELECT *` (select all attributes) is written here, but in actual application, only one attribute is not needed, which is "organizerID".

```
SELECT *

FROM lyu1501_coupon

NATURAL JOIN lyu1501_class

WHERE couponID = '$pid'
```

By looking at the following schema, the only attribute for joining two table is "eventID", which is the primary key for "class" table and foreign key for "coupon" table.

```
class(eventID: integer, eventTitle: text, eventDate: date,
        eventStartTime: time, eventEndTime: time, eventDetail: text,
        organizerID: varchar: private: int)
coupon(couponID: varchar, eventID: integer,
         remarks: text, used: int, checked: int, checkBy: varchar)
```

When consider choosing a suitable join, natural join is the most “natural” choice because only useful result is produced. Inner join also produces same set of useful result but the join attribute is duplicated. If we choose inner join, “eventID” will appear 2 times in every tuple, which is redundant. Not specifying type of joins is also allowed in MySQL, i.e.:

```
FROM lyu1501_coupon, lyu1501_class
```

Which we sometimes adapted when we start to construct the SQL query. Not specifying the type of join is equivalent to Cartesian product, also known as cross join. [8] No matter natural join or Cartesian product both produce the correct result, but of course, in real practice, natural join is used, as Cartesian product produces a huge table before we filter the useful one by the `WHERE` clause, which is time-wasting, and undesirable.

The process of executing the query is like this:

1. Join “coupon” table and “class” table.
2. For this new table, select a particular tuple, which means a particular coupon the user requested.
3. Do “project” operation, eliminates the attribute “organizerID”.

The logical query plan is as the following diagram.

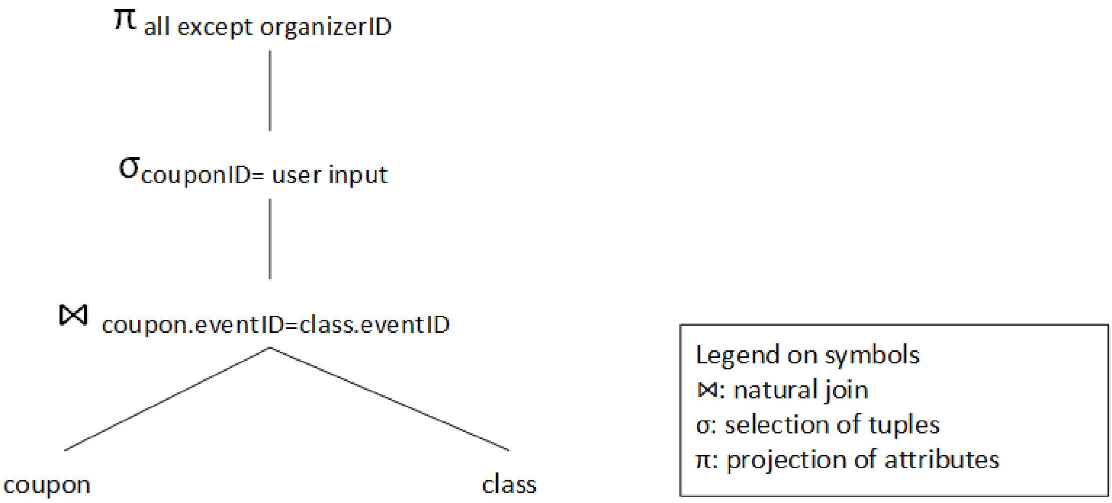


Figure 15 Logical query plan of "join, select, project"

If there are many rows in the database. Do selection and projection before join may help increasing the speed for getting the result. The logical query plan becomes the follow form:

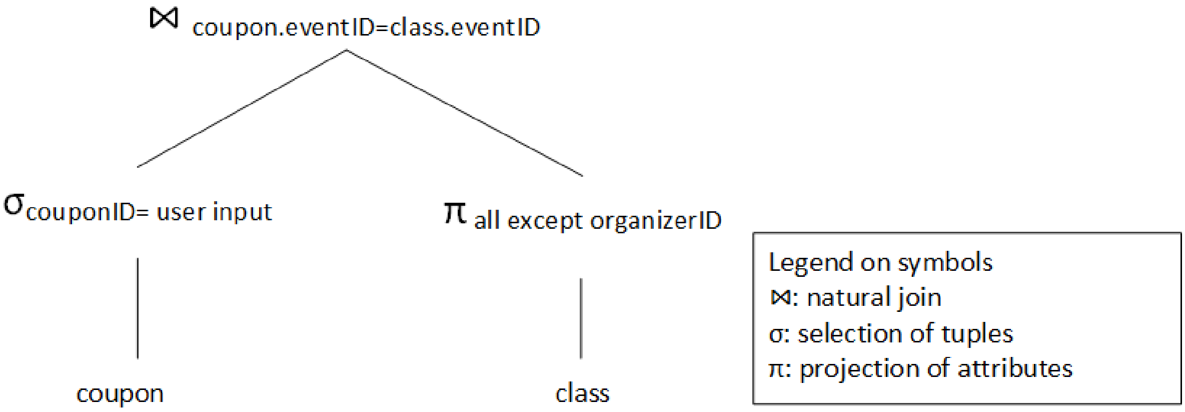


Figure 16 Logical query plan of doing "select, project, join"

At this moment, there are very few number of rows in the database. After consideration, it is not necessary to change our query plan as there are no difference on which plan is chosen. The readability of the SQL query is higher if we keep the SQL query unchanged.

4.1.4 Data security: SQL injection

Data security is important for any system. Although there is no bullet-proof system, but using strategies could reduce the chance of being invaded. SQL injection is one of the methods to attack the database. User can destroy the database by this method. In the first semester, the function "mysql_real_escape_string" is used in PHP code to avoid some SQL injection problem. This function escapes special characters such as single quotes, by prepending a backslash. [9] For example, the following code in "`add_class.php`" shows the usage of this function for the field event title and details where users can input any string.

```
$eventTitle = mysql_real_escape_string($eventTitle);
$eventDetail = mysql_real_escape_string($eventDetail);

$result = mysql_query("INSERT INTO  `viewtech`.`lyu1501_class`
(`eventID` ,`eventTitle` ,`eventDate` ,`eventStartTime` ,`eventEndTime` ,
`eventDetail` ,`organizerID` ,`private`)VALUES ('$pastEvent',
'$eventTitle',  '$eventDate',  '$eventStartTime',  '$eventEndTime',
'$eventDetail',  '$organizerID',  '$isPrivate');");
```

However, using the above solution is not the best one. The wildcards in LIKE operator are not escaped. Moreover, a large calls of "mysql_real_escape_string" slows down the database as it calls MySQL library every time [10]. In the second semester, MySQLi with prepare statement is used to fix this problem. The following example is extracted from the file `view_teacher_class.php`. Where a particular organizer ID is chosen.

```
$result = mysqli_prepare($db, "SELECT eventID, eventTitle, eventDate, 
private FROM `lyu1501_class` WHERE  `organizerID` LIKE ? ");

mysqli_stmt_bind_param($result, 's', $teacherID);
mysqli_stmt_execute($result);
mysqli_stmt_bind_result($result, $eventID, $eventTitle, $eventDate, 
$isPrivate);

mysqli_stmt_store_result($result);
```

In the above code, a prepare statement “$result” have been created, then, variable “$teacherID” is bind into the prepare statement before executing the query.

4.1.5 System failure

System failure may occur. All the records in the database are needed to back up regularly. In case the system is down or the database is corrupted, there still one more copy of database, so as to protect the data, preventing from loss.

4.1.6 ID Design and implementation

The structure of the identity numbers, including the coupon code and the temporary ID which is shown in the kiosk (in the form of QR code), is discussed in this sub-section.

4.1.6.1 Coupon Code

Each coupon has a unique code. Every coupon is assigned a 20-digit number, and the code is constructed as below, explained with an example.

0	9	9	2	1	1	2	0	1	6	0	4	0	5	1	2	1	9	3	7
I			II			Year				Month		Day		Hour		Minute		Second	

I. It is the coupon sequence number in this event. The above example means this is the 99th coupon issued in this event. It returns to ‘000’ after ‘999’.

II. It is the overall coupon sequence number. The above example means this is the 211th coupon issued. It returns to ‘000’ after ‘999’.

The following pseudocode is implemented in the server as PHP code.

```
if(coupon sequence in the event >= 1000){
    coupon sequence in the event = 0;
}

if(overall coupon sequence >= 1000){
    overall coupon sequence = 0;
}

$now = new DateTime()

$nowString = format $now to
year(4-digit) + month (2-digit) + day (2-digit) + hour (24 hour format,
2-digit) + minute (2-digit) + second (2-digit)

$couponCode = str_pad(coupon sequence in the event, 3, '0',
STR_PAD_LEFT). str_pad(overall coupon sequence, 3, '0', STR_PAD_LEFT) .
$nowString;
```

4.1.6.2 Temporary ID in kiosk

Each temporary ID has a unique code. It is displayed in the kiosk as a QR code. It is a 9-digit number, and the code is constructed as below:

0	3	1	4	1	7	1	9	5
I			Random number between (000000 to 999999)					

I. The overall sequence number. The above example means this is the 31st temporary ID generated. It returns to ‘000’ after ‘999’.

The following pseudocode is implemented in the server as PHP code.

```
if(sequence number >= 1000){
    sequence number = 0;
}

$fullID =
str_pad(sequence number, 3, '0', STR_PAD_LEFT) . str_pad(rand(0,999999),
6, '0');
```

4.2 Client Side

4.2.1 Modules design: Concerning different users

In the “homework coupon” system, we could divide functions available in the app into 8 major modules. The following UML use-case diagram shows the permission of different users in using those modules. We have 3 kinds of actors, namely “student”, “teacher”, and “tutor”. Moreover, the iOS app is also available for students but NFC function could not be used, a kiosk with NFC function is available for students who do not have NFC equipped smartphone to view their coupons.

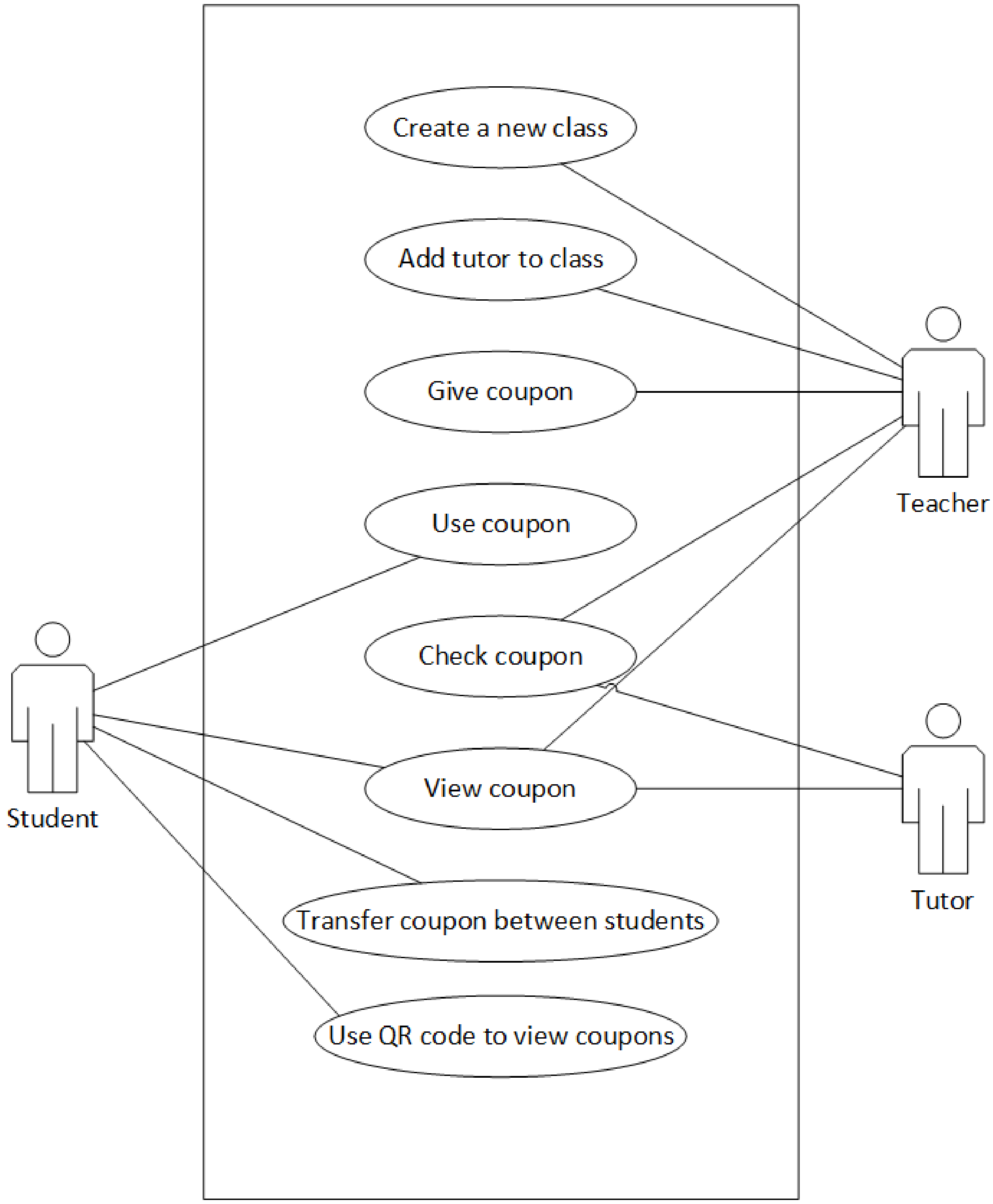


Figure 17 UML use-case diagram showing 8 major functions on basis of actors and actions

4.2.2 Module design: Data flow diagram

The following data flow diagram shows the 5 major functions that the teacher could use.

Additionally, tutor can use the function "check coupon".

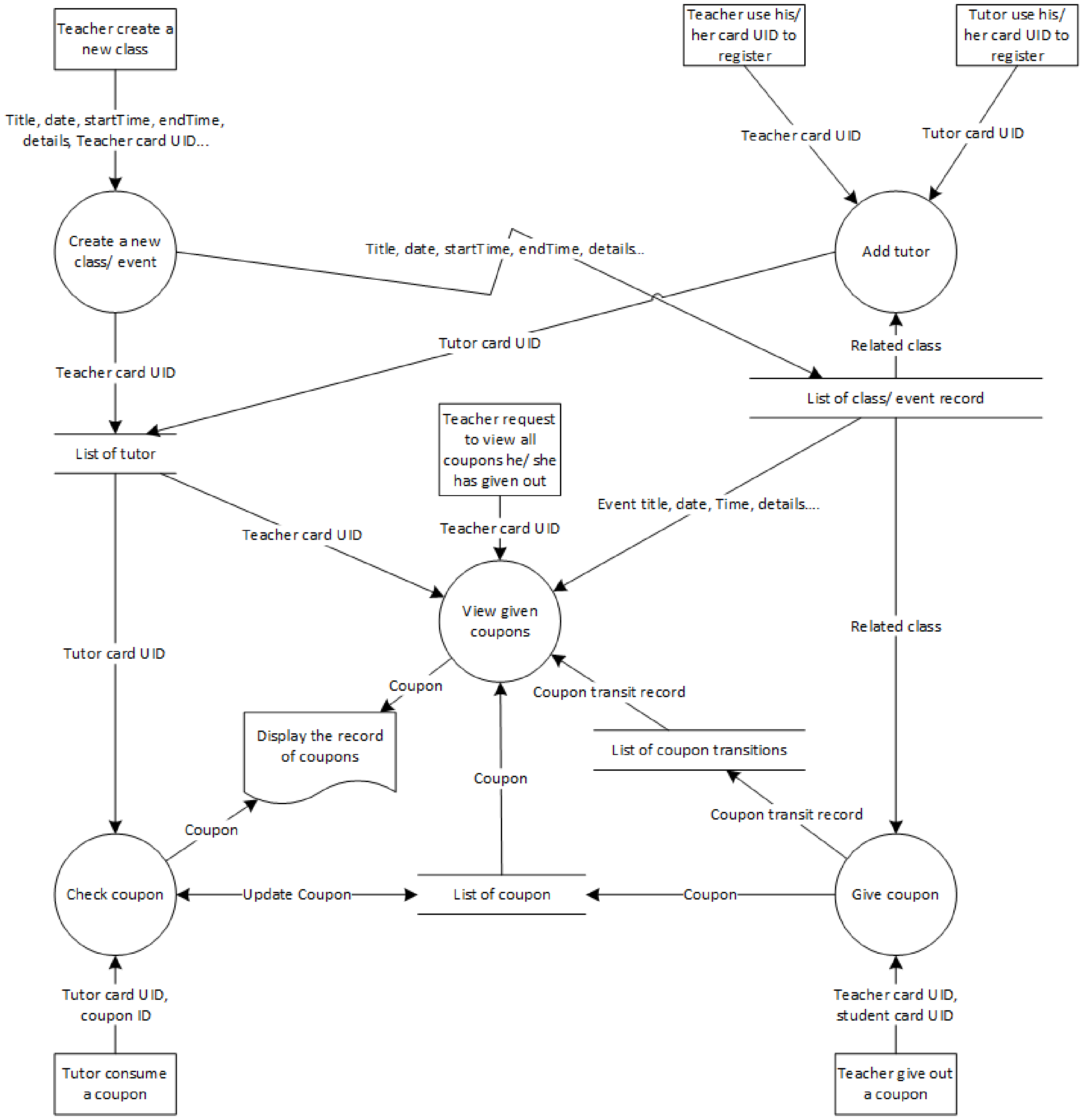

Teacher create a new class
Teacher use his/ her card UID to register
Tutor use his/ her card UID to register
Title, date, startTime, endTime, details, Teacher card UID...
Teacher card UID
Tutor card UID
Create a new class/ event
Title, date, startTime, endTime, details...
Add tutor
Tutor card UID
Related class
Teacher card UID
List of class/ event record
List of tutor
Teacher request to view all coupons he/ she has given out
Teacher card UID
Event title, date, Time, details....
Teacher card UID
View given coupons
Tutor card UID
Related class
Coupon
Coupon transit record
Display the record of coupons
List of coupon transitions
Coupon
Coupon transit record
Coupon
Check coupon
Update Coupon
List of coupon
Coupon
Give coupon
Tutor card UID, coupon ID
Teacher card UID, student card UID
Tutor consume a coupon
Teacher give out a coupon

Figure 18 Data flow diagram showing the major functions related to teacher & tutor

The UML use-case diagram in the previous section shown that a student can use 4 major functions. One of the functions “View given coupons” is described in the previous diagram. The remaining functions are described below.

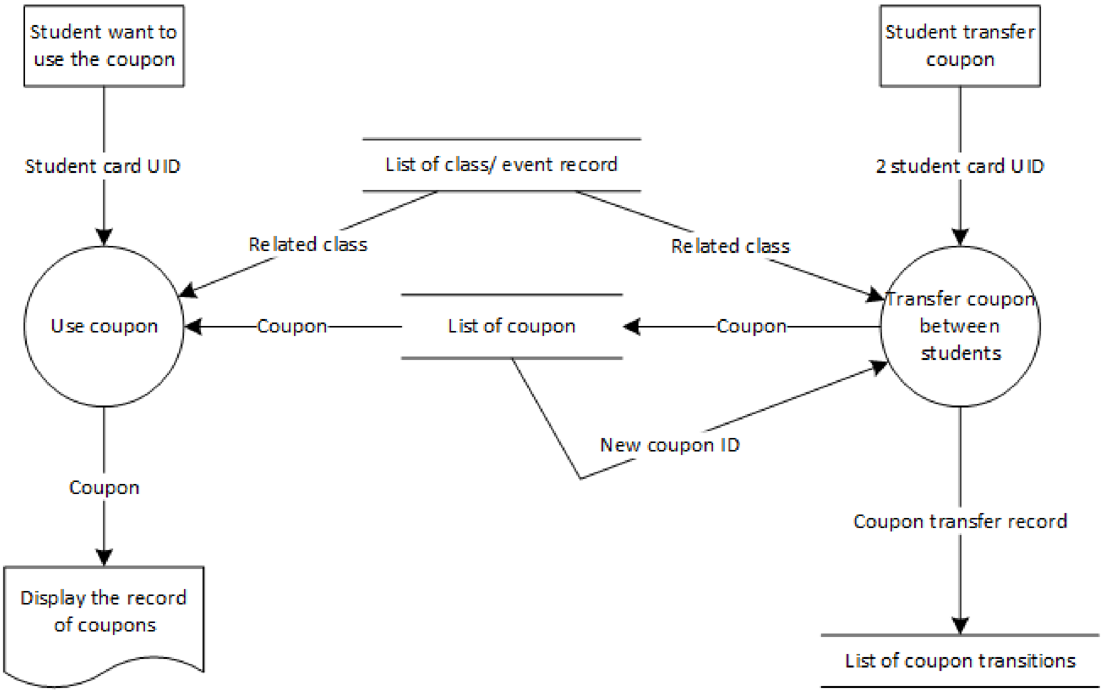


Figure 19 Data flow diagram showing the major functions related to student, except the part using QR code.

The last function is “Use QR code to view coupons”. For students who do not have a NFC equipped smartphone. They could use an alternate method to view their coupons on their smartphone. They could tap their card to the kiosk and scan the QR code generated by the kiosk to view the coupons.

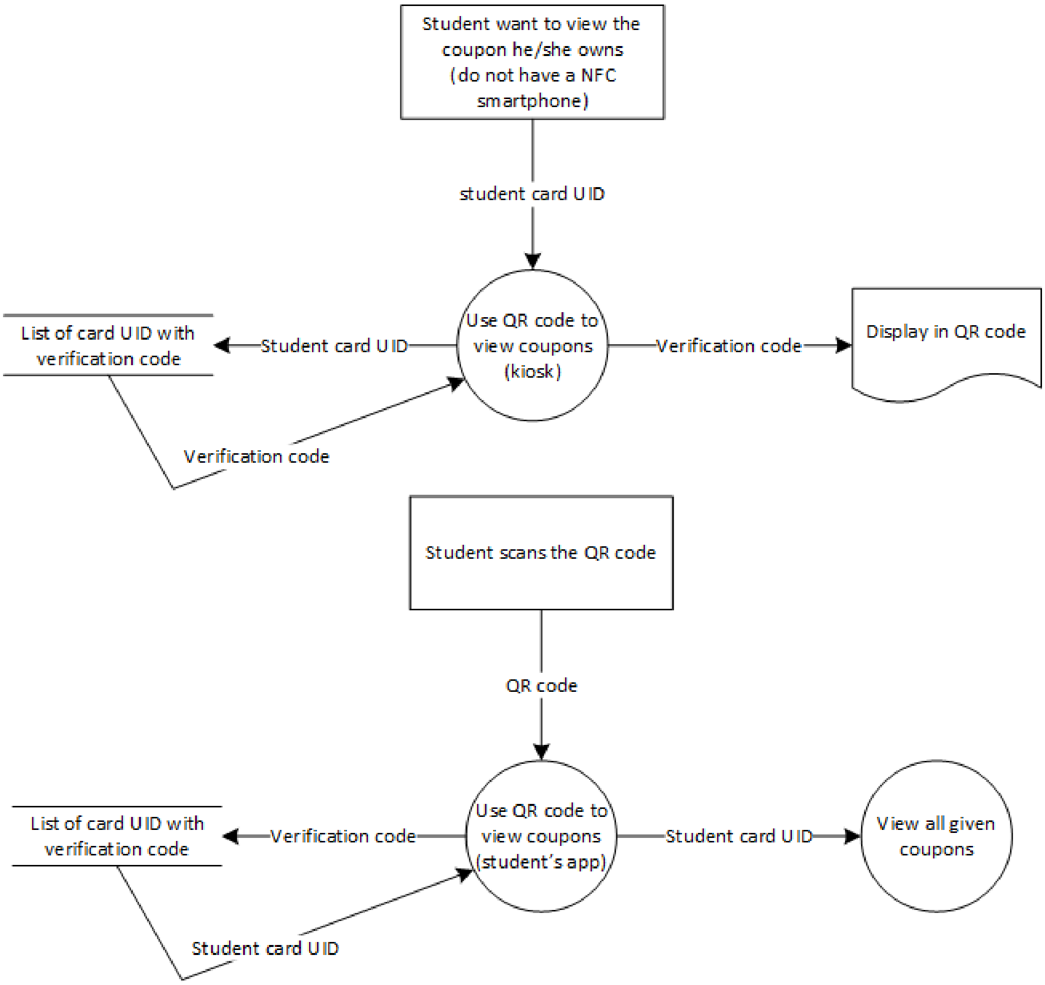

Student want to view the coupon he/she owns (do not have a NFC smartphone)
student card UID
Use QR code to view coupons (kiosk)
Student card UID
List of card UID with verification code
Verification code
Verification code
Display in QR code
Student scans the QR code
QR code
Use QR code to view coupons (student's app)
Verification code
List of card UID with verification code
Student card UID
Student card UID
View all given coupons

Figure 20 Data flow diagram showing the how to use the QR code to view coupons

4.2.3 User interface design

4.2.3.1 Android application

A simple and intuitive design of a system is important in terms of user experience. A good user experience can attract user in using the application. We try to reduce the number of buttons need to be click when using the app. As our application have 3 types of users, teacher, tutor and students, we separate into 3 apps, each app designed for the particular type of user. Moreover, to cater students without a NFC enabled smartphone, we have constructed a kiosk. The following diagrams shows the wireframe of our Android application.

Tutor App

The tutor app is the simplest one. The main page is for searching coupons by coupon ID. User can click on the “view coupon transition log” button to go to next page to the transfer history. The record is shown on the screen.

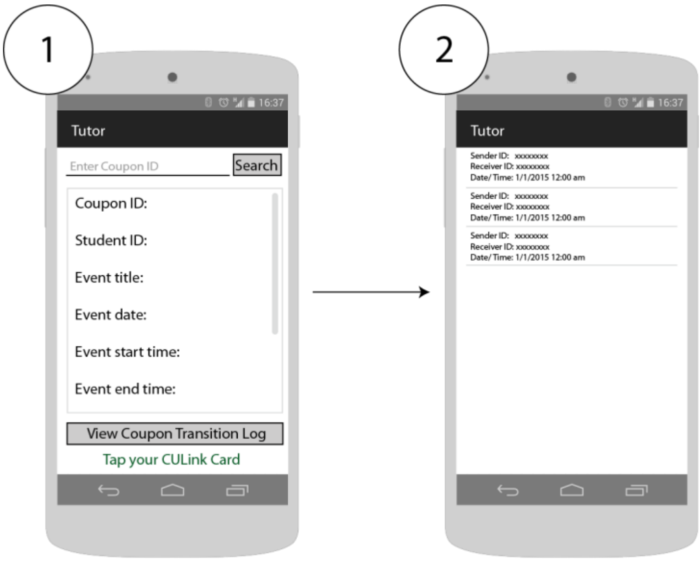


Figure 21 Graph showing the flow of screen for tutor

1. Main Page (search coupon ID)
2. View coupon transition log

Student App

The main function of student part is to view the coupons he/ she got. The coupons consumed are also shown on the screen. When the student chooses a record, the details are shown. On pressing the “Decide to use this coupon” button, the coupon will be marked as “used”. So when he/ she submit the homework with this coupon, the tutor who corrects the homework knows if the student intended to use that coupon. Moreover, the student can give his/ her coupon to other users. The design has considered smartphones with or without NFC function.

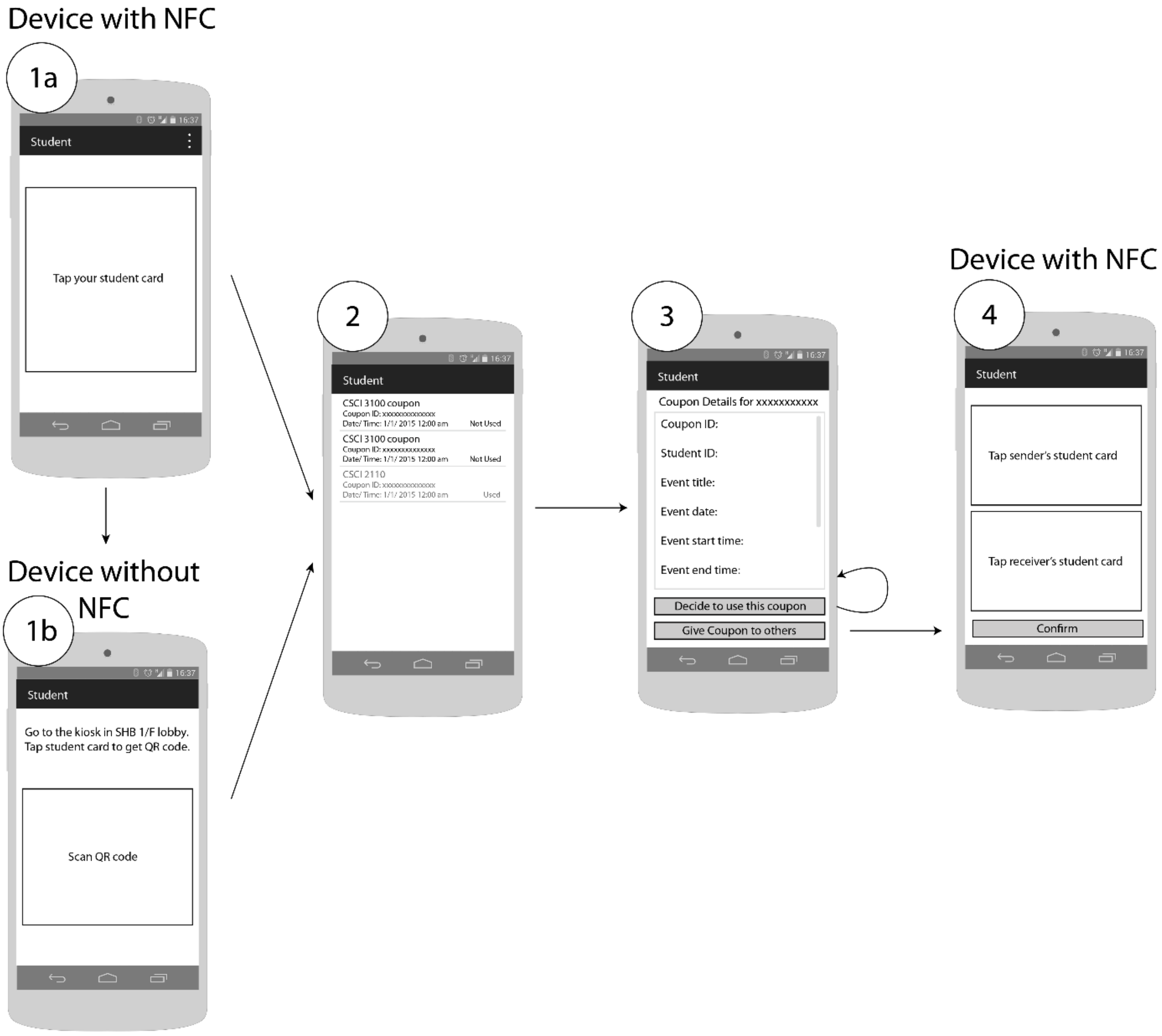

Device with NFC
1a
16:37
Student
Tap your student card
Device without NFC
1b
16:37
Student
Go to the kiosk in SHB 1/F lobby. Tap student card to get QR code.
Scan QR code
2
16:37
Student
CSCI 3100 coupon
Coupon ID: xxxxxxxxxxxxxx
Date/ Time: 1/1/ 2015 12:00 am
Not Used
CSCI 3100 coupon
Coupon ID: xxxxxxxxxxxxxx
Date/ Time: 1/1/ 2015 12:00 am
Not Used
CSCI 2110
Coupon ID: xxxxxxxxxxxxxx
Date/ Time: 1/1/ 2015 12:00 am
Used
3
16:37
Student
Coupon Details for xxxxxxxxxxx
Coupon ID:
Student ID:
Event title:
Event date:
Event start time:
Event end time:
Decide to use this coupon
Give Coupon to others
Device with NFC
4
16:37
Student
Tap sender's student card
Tap receiver's student card
Confirm

Figure 22 Graph showing the flow of screen for student

1. Home screen

    a. Tap student card. Devices with NFC function will show this page automatically. Users can also go to page 1b by selecting “Scan QR Code” in the menu.

    b. Scan QR code. Devices without NFC function will show this page automatically.

2. List of coupon a student got
3. Details of a particular coupon
4. Give coupon to others. This function is only available for devices with NFC function.

Teacher App

The teacher has the highest permission; he/ she has 5 major functions. The flow is shown in the following diagrams.

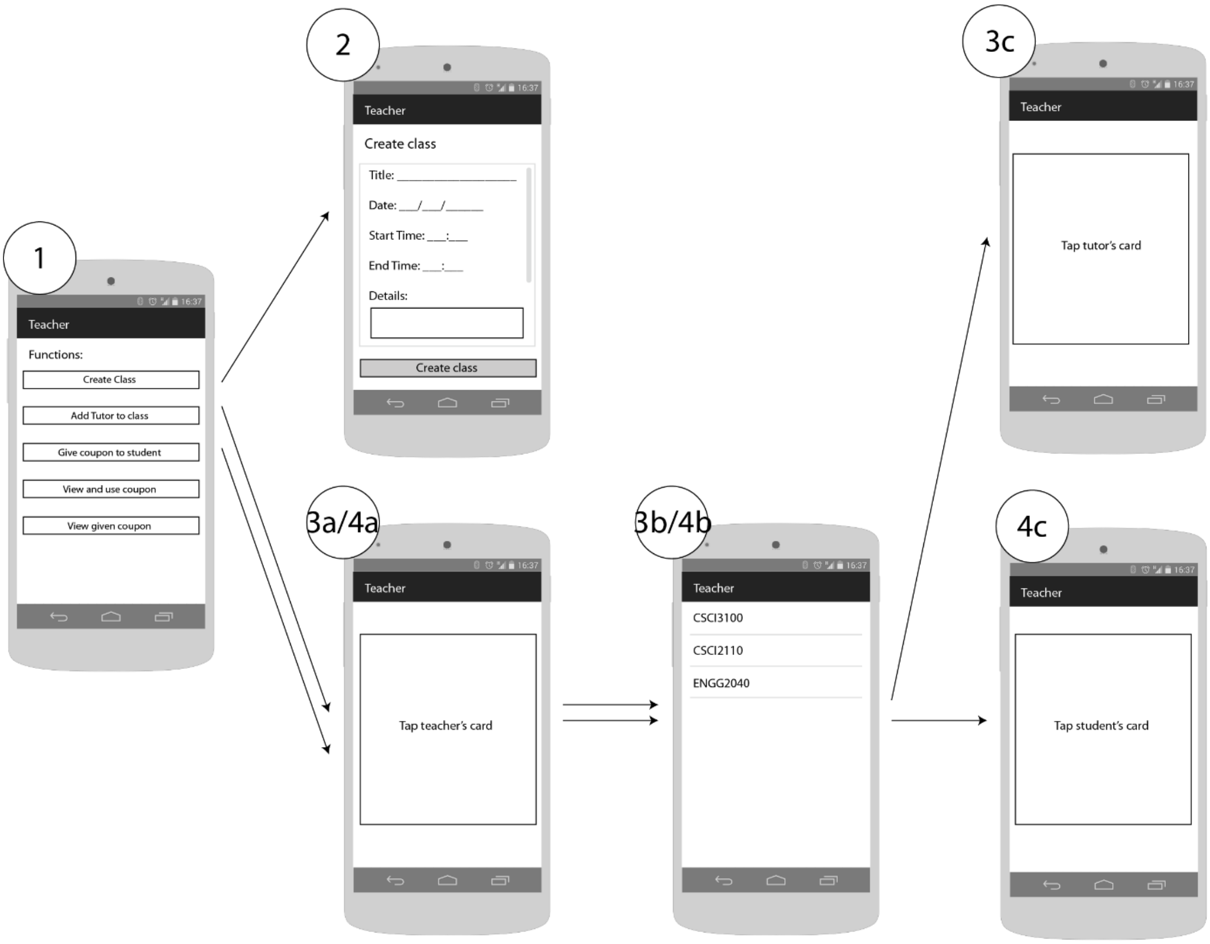


Figure 23 Graph showing the flow of screen for teacher.

1. Main page
2. Create class
3. Add tutor to a class (step 3a request teacher's NFC card, 3b: teacher choose a class, 3c: request tutor's NFC card)

4. Give coupon to student (step 4a request teacher's NFC card, 4b: teacher choose a class, 4c: request student's NFC card)

Screen 2 is the interface of creating class. When a class is successfully created, the teacher can go back to the main page.

When the teacher pressed the second button "add tutor to class" on the main page. First of all, screen 3a is shown, requesting teacher to tap his/ her CU Link card. After that, screen 3b is shown if the teacher has created more than 1 class. Teacher need to choose the correct class. If the teacher has created 1 class only, screen 3b will be skipped and directly go to screen 3c. Screen 3c request tutor to tap his/ her card, success or failure message will be shown after the card is tapped. The app remains at screen 3c if the user has not pressed the "back" button at the bottom of the screen. If there are more tutors in the same course, he/ she can continue to tap one by one.

For the third function, that is "giving coupons to student", belongs to screens 4a through 4c. Screen 4a and 4b are same as screen 3a and 3b respectively. Screen 4c is similar with 3c except that the system request student to tap the card. A coupon is given on every tap.

The following figure shows the remaining 2 functions of the app. Which are "check/ consume coupons" and "view coupons".

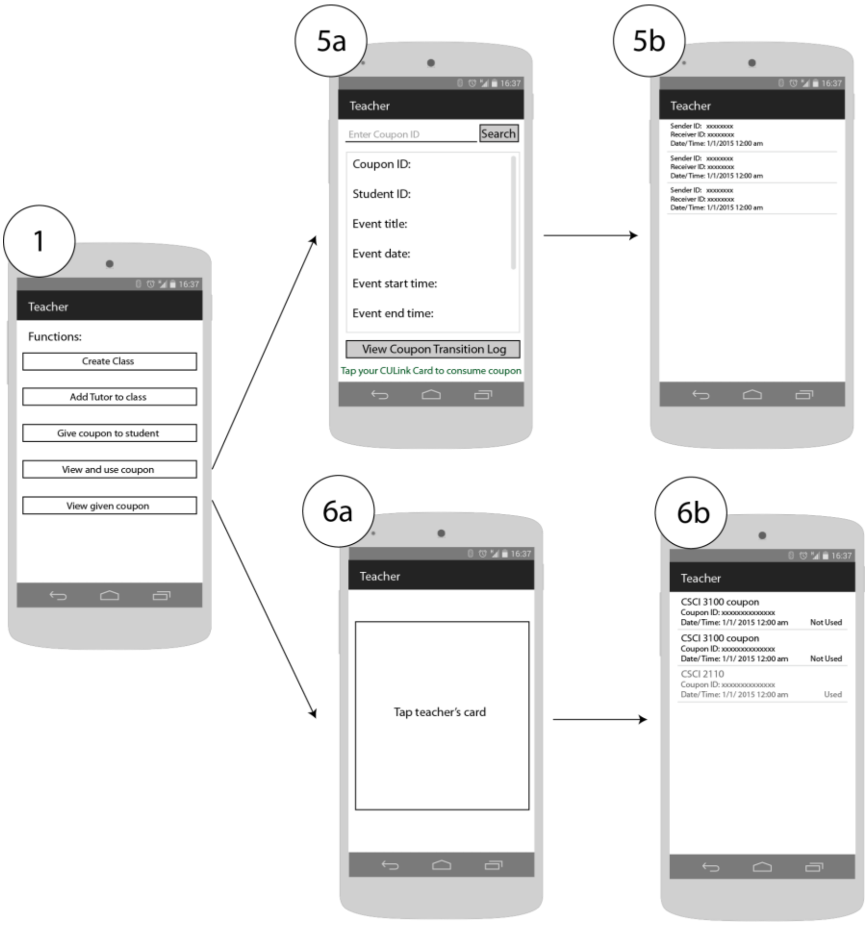


Figure 24 Graph showing the flow of screen for teacher.

1. Home page

5a. Search coupon ID

5b. View coupon transaction log

6a. Tap teacher’s card

6b. List of coupon a teacher give out

Screen 5a and 5b belongs to the function consume coupon. The usage is the same as the tutor app. Teacher got the permission to mark the coupon as checked, or say consumed. Same with tutor, teacher can view the coupon transaction log.

Screen set number 6 are for viewing the coupons given out. First of all, teacher’s card is read (screen 6a), then all the coupons are listed out (screen 6b).

Kiosk

The kiosk aims to generate a QR code after a student has presented his/ her student card. So that the student can use the “student app” which is mentioned previously to read his/ her own coupons. The home page prompt the user to tap the student card to the kiosk. After that, a QR code will be shown. It will disappear automatically after 15 seconds. Student could tap the student card again to exit this page. If the student wants QR code again, he/ she can tap the student card to the kiosk again to get another one. The wireframe is shown below.

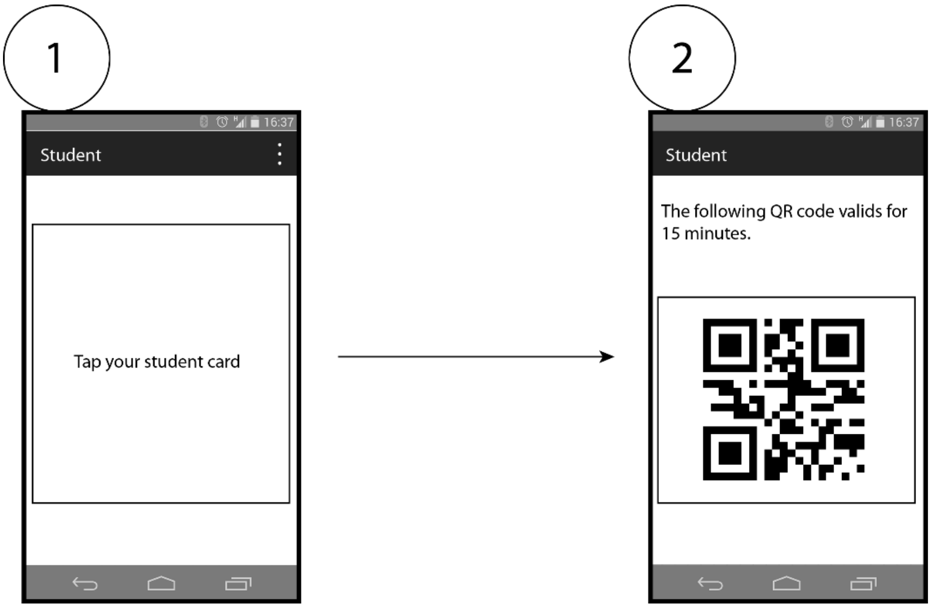


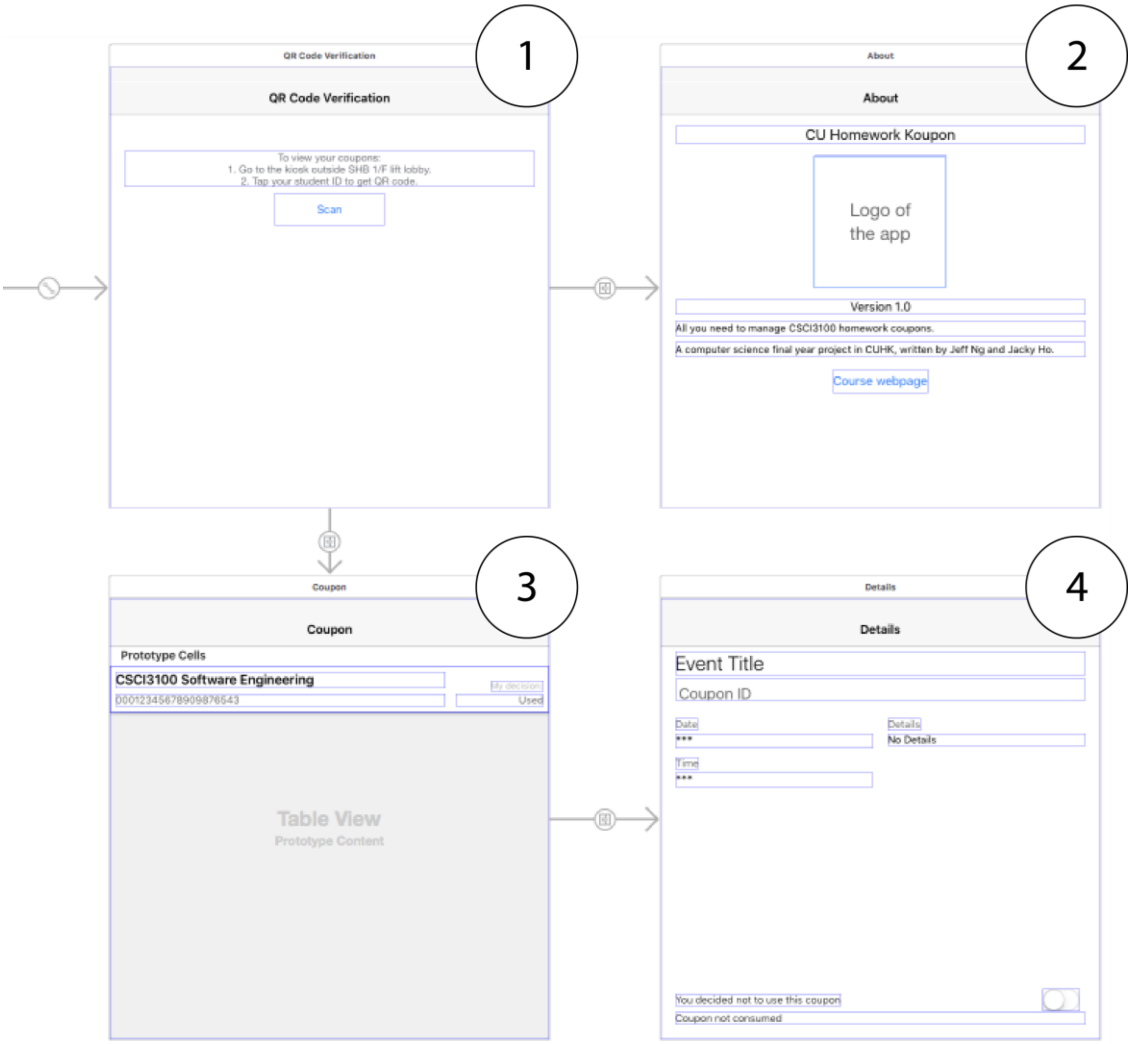
Figure 25 Graph showing the flow of screen for the kiosk.

1. Home page
2. QR code page

4.2.3.2 iOS application

We provide an iOS version for the student app as there are iOS users. The flow is similar to the student app mentioned in the previous section, except that the NFC function has been taken away because iOS does not support NFC.

The following figure shows the flow of the whole app. Note the following is diagram is the similar to the file “Main.storyboard” in Xcode.



To view your coupons:
1. Go to the kiosk outside SHB 1/F lift lobby.
2. Tap your student ID to get QR code.

All you need to manage CSCI3100 homework coupons.

A computer science final year project in CUHK, written by Jeff Ng and Jacky Ho.

You decided not to use this coupon

Coupon not consumed

1. Home Page
2. About Page
3. List of coupon a student got
4. Details of a particular coupon

When the “Scan” button in the home page is pressed, a camera pop-up window is shown so that the student can scan the QR code. Page 3 is shown if the QR code is identified. Choosing any one record will segue to page 4.

Pressing “information” button on the top-right hand corner in the home page will segue to the about page (page 2).

4.2.4 Modules implementation: Sequence Diagram

New Class/ Event

The client using the Android app is teacher. When a teacher clicks the button to add a new class/ event, he/ she need to input some basic information about the class/ event, including the class/ event title, which is a mandatory field, other optional fields including date, starting time, ending time, details. The UID of CU Link card of the teacher is also retrieved and send to the PHP (server). PHP then adds the following information to the database. After that, PHP sends an echo back to the client to indicate that the class is successfully added.

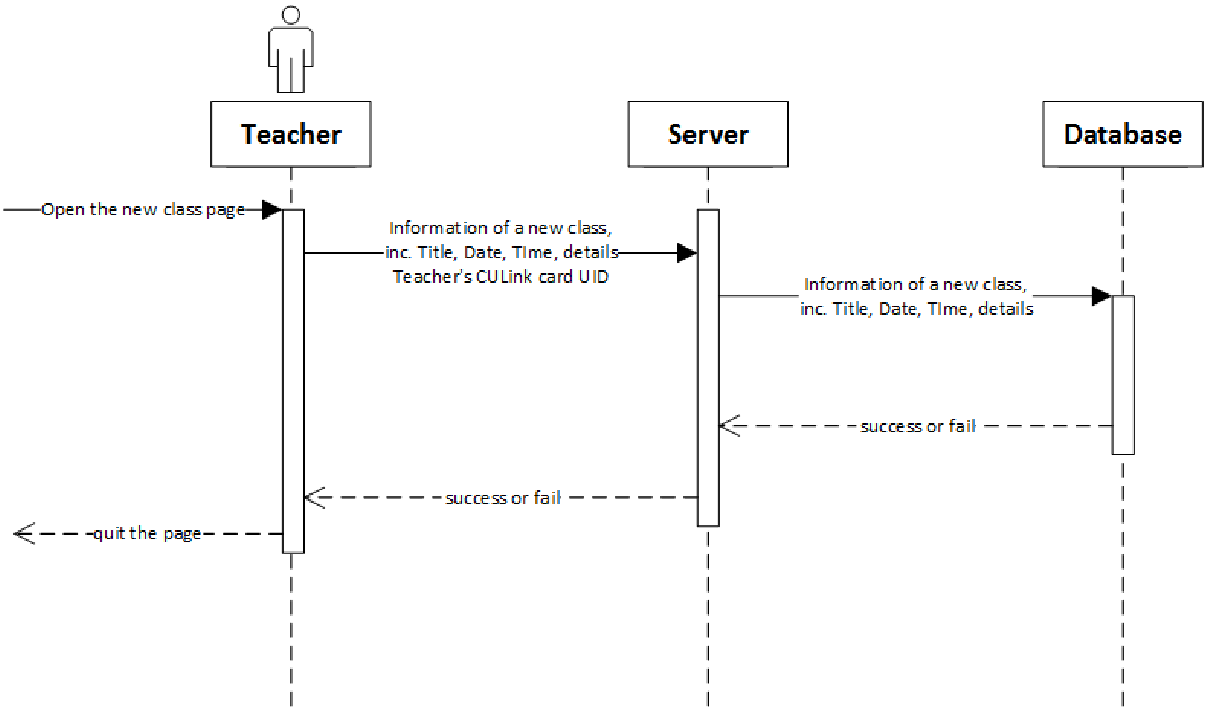

Teacher
Server
Database
Open the new class page
Information of a new class,
inc. Title, Date, TIme, details
Teacher's CULink card UID
Information of a new class,
inc. Title, Date, TIme, details
success or fail
success or fail
quit the page

Figure 26 UML sequence diagram showing the data exchange of adding a new class/ event.

Add tutor

A teacher can add tutors to the class. Teacher's CU Link card UID and Tutor's CU Link card UID are required. Those information is sent to the PHP server. PHP will add a record to the database. Normally, PHP will send a success message to the Android device. The following UML sequence diagram supposed the one using Android application is teacher. If the one using the app is tutor, the arrow in dotted lines indicating "success or fail" from server to teacher will become from server to tutor.

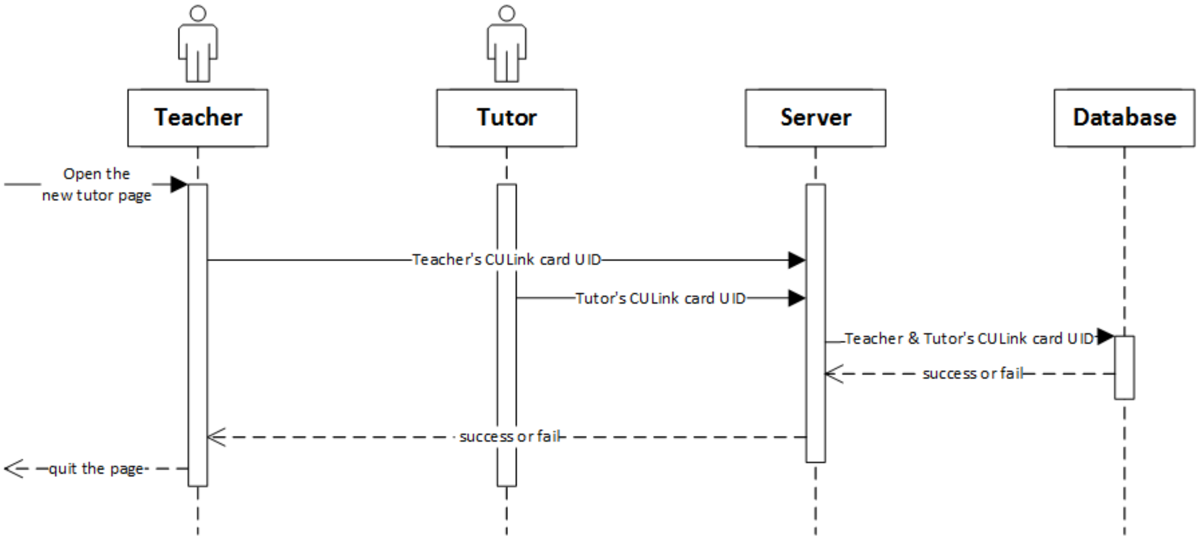


Figure 27 UML sequence diagram showing the data exchange of adding a tutor to a class.

Give out coupons

The following diagram shows how a teacher gives out a coupon to students. Similar with adding a tutor, Teacher's CU Link card UID and student's CU Link card UID is retrieved and sent to PHP. The server will check whether teacher's card is a legitimate one, which means the one who create the class/ event and giving out the coupon belong to the same person. If so, a coupon is created by PHP, inserted into the database. Finally, a success message is returned. If the teacher's card is not a legitimate one, the PHP echo a fail message back to the Android device.

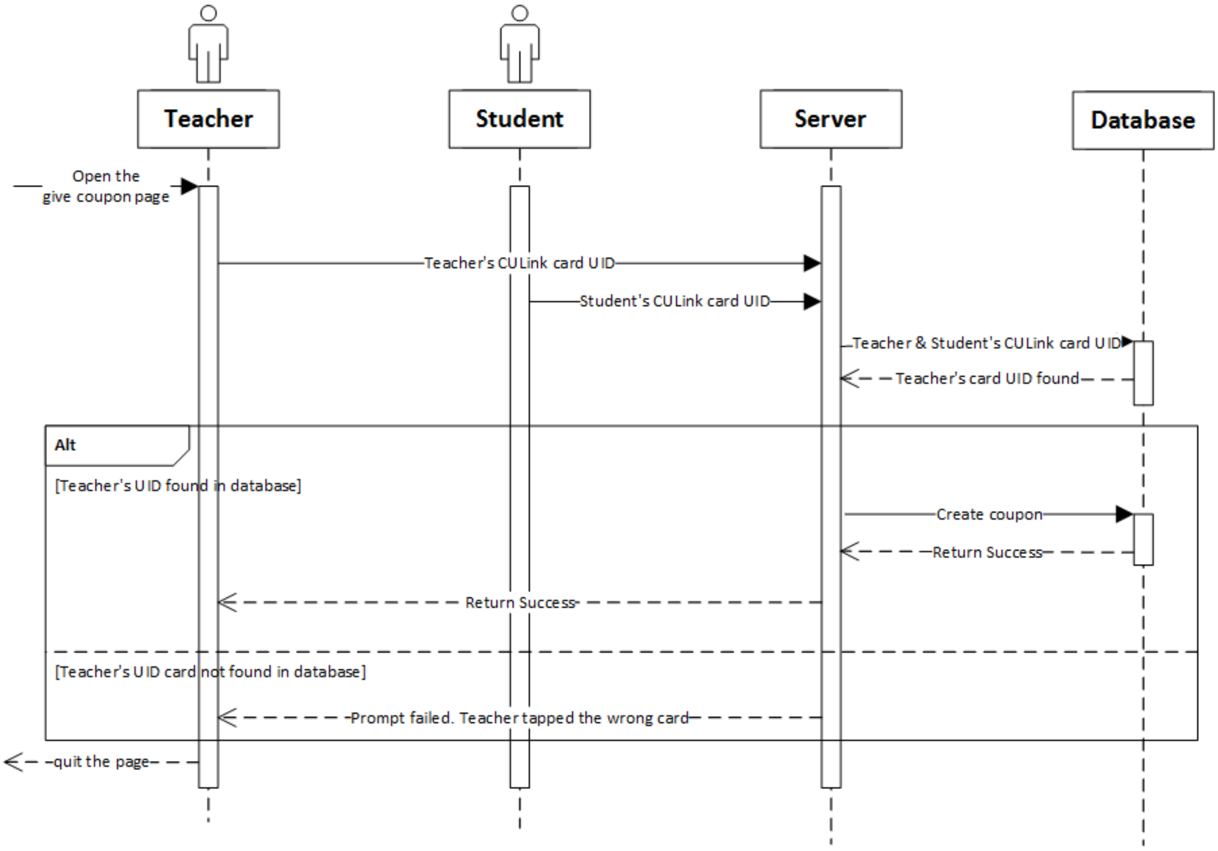

Teacher
Student
Server
Database
Open the give coupon page
Teacher's CULink card UID
Student's CULink card UID
Teacher & Student's CULink card UID
Teacher's card UID found
Alt
[Teacher's UID found in database]
Create coupon
Return Success
Return Success
[Teacher's UID card not found in database]
Prompt failed. Teacher tapped the wrong card
quit the page

Figure 28 UML sequence diagram showing the data exchange of giving out a coupon.

Use Coupon

To use a coupon, there are two parts of activities, the first part involving the student indicating he/ she wants to use a particular coupon. The following diagram describes this module. Another part is the tutor checks the coupon and mark as consumed, which is described after this part. The student who is using the app presses the “use coupon” button. A message is sent to PHP. Then the PHP will query the database if the student decides to use/ not to use coupon, also if the coupon has been consumed. We can summarize into 3 cases, as shown in the “Alt” part in the diagram.

1. Situation: The student was not going to use the coupon, and the coupon is not consumed.
   Action: Server send a query to update the database, mark that the student is decide to use this coupon. Success message is sent back to the client.
2. Situation: The student was going to use the coupon, and the coupon is not consumed.
   Action: Server send a query to update the database, mark that the student make up his/ her mind, not to use this coupon. Success message is sent back to the client.
3. Situation: The coupon is consumed.
   Action: As it is consumed already, the server sends a failure message to the app.

The diagram is shown on the next page.

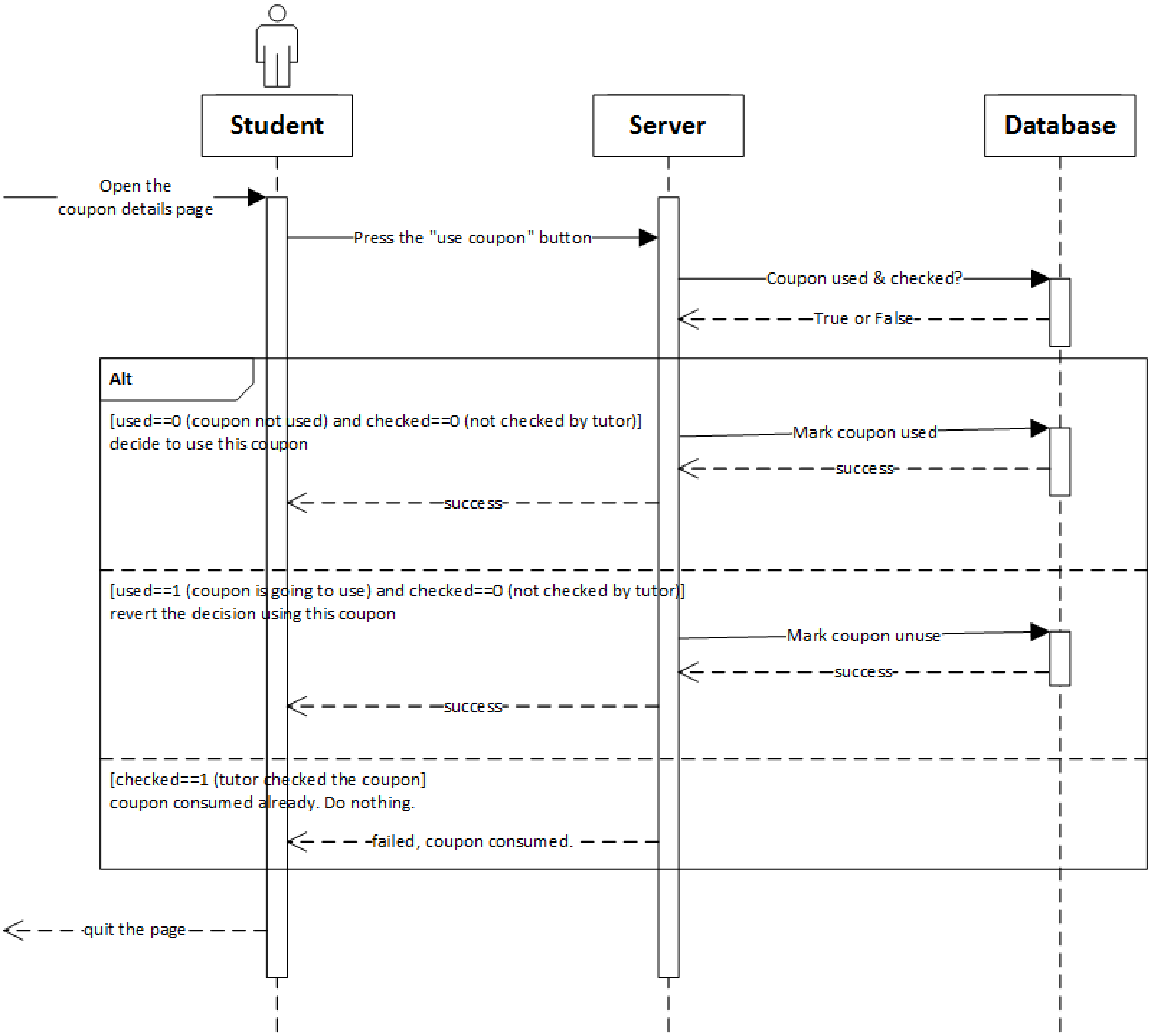

Student
Server
Database
Open the coupon details page
Press the "use coupon" button
Coupon used & checked?
True or False
Alt
[used==0 (coupon not used) and checked==0 (not checked by tutor)]
decide to use this coupon
Mark coupon used
success
success
[used==1 (coupon is going to use) and checked==0 (not checked by tutor)]
revert the decision using this coupon
Mark coupon unuse
success
success
[checked==1 (tutor checked the coupon]
coupon consumed already. Do nothing.
failed, coupon consumed.
quit the page

Figure 29 UML sequence diagram showing the data exchange of a student indicating whether he/ she is going to use a coupon.

Check Coupons

The tutor checks the coupon provided by the student and mark as consumed. The coupon ID is sent to PHP. PHP then finds relating records in the database and return the record. Similar with the situation, there are 3 situations:

1. Situation: The student was not going to use the coupon, and the coupon is not consumed.
   Action: Note that tutor knows the student is not going to use the coupon before he/ she press the button as the information is shown on the page. He/ she may consider follow up action. If he presses the button to consume the coupon, server sends a query to update the database, mark the coupon as consumed, and set that student want to use the coupon. Success message is sent back to the client (tutor).
2. Situation: The student was going to use the coupon, and the coupon is not consumed.
   Action: Server sends a query to update the database, mark that the coupon is checked and consumed. Success message is sent back to the client.
3. Situation: The coupon is consumed.
   Action: As it is consumed already, the server sends a failure message to the app.

The diagram is shown on the next page.

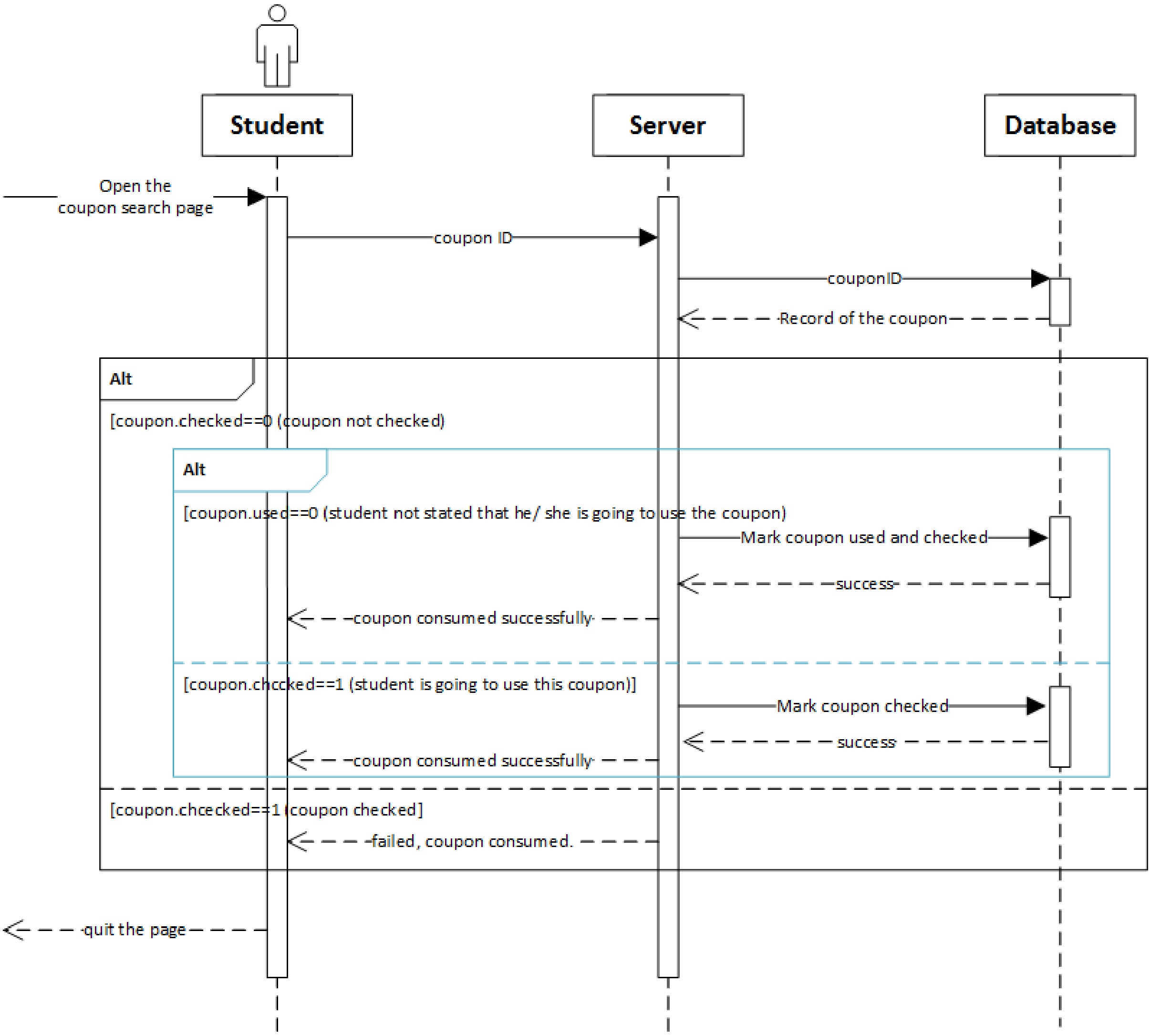

Student
Server
Database
Open the coupon search page
coupon ID
couponID
Record of the coupon
Alt
[coupon.checked==0 (coupon not checked)
Alt
[coupon.used==0 (student not stated that he/ she is going to use the coupon)
Mark coupon used and checked
success
coupon consumed successfully
[coupon.chccked==1 (student is going to use this coupon)]
Mark coupon checked
success
coupon consumed successfully
[coupon.chcecked==1 (coupon checked]
failed, coupon consumed.
quit the page

Figure 30 UML sequence diagram showing the data exchange of searching out a particular coupon to validate (mark the coupon as consumed).

View Coupon Details

The client taps the NFC card to the device, the UID is read and sent to the PHP server. PHP then search the database by the UID and return a list of coupons. The PHP returns an array of JSON records to the client. On the client side, the application read every entry of JSON and display in a list manner. The client can click on a particular entry, that particular coupon ID is sent to the server and the server queries the database again. Detailed information related to the coupon including the title, date, time, and details.

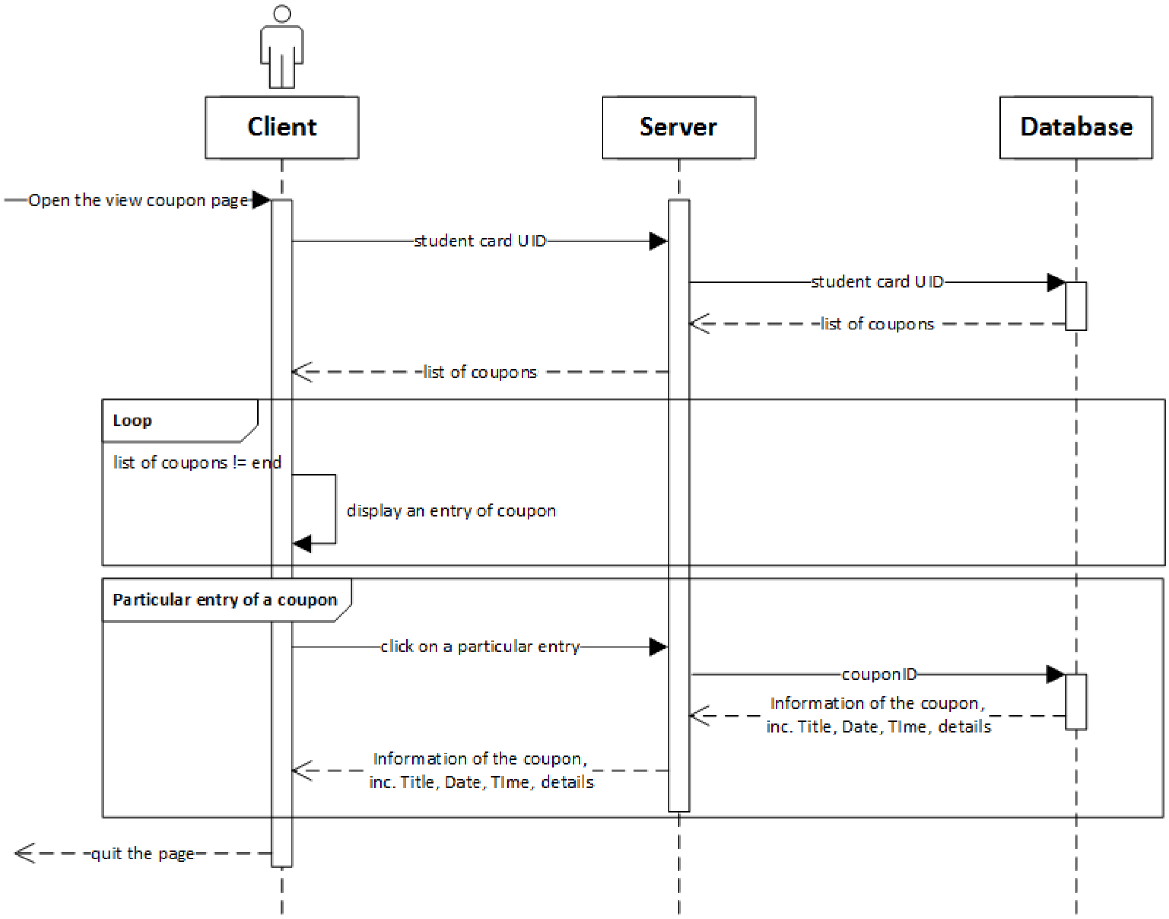

Client
Server
Database
Open the view coupon page
student card UID
student card UID
list of coupons
list of coupons
Loop
list of coupons != end
display an entry of coupon
Particular entry of a coupon
click on a particular entry
couponID
Information of the coupon, inc. Title, Date, TIme, details
Information of the coupon, inc. Title, Date, TIme, details
quit the page

Figure 31 UML sequence diagram showing the data exchange of viewing the detail of coupons the user has.

Giving Coupons from one student to another

Coupons could be given from student to another. On pressing the button in the view coupon details page, the Android app will check if the coupon consumed based on the information got in previous queries. The system does not allow such function if the coupon is consumed or the student decided to use the coupon. If the client could give out coupons, his/ her CU Link card UID and the receiver's CU Link card UID is retrieved and sent to PHP. Similar with the previous module, the UIDs together with the coupon ID is used to search the database. Finally return success.

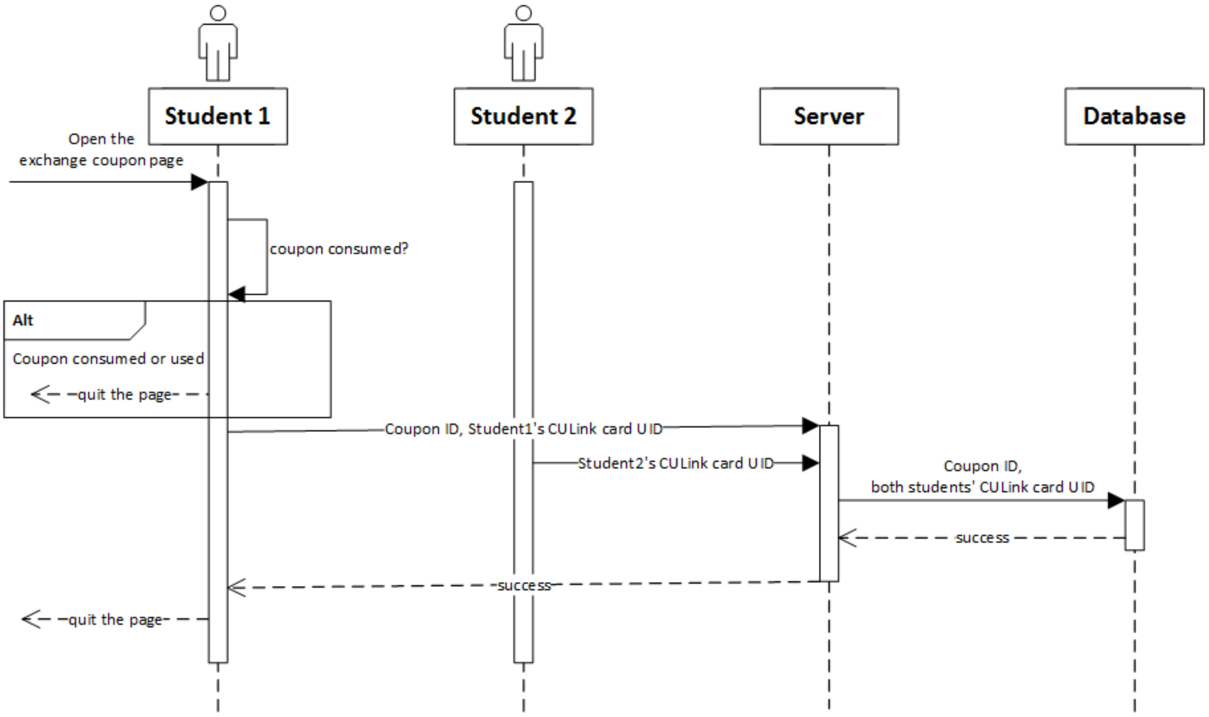

Student 1
Student 2
Server
Database
Open the exchange coupon page
coupon consumed?
Alt
Coupon consumed or used
quit the page
Coupon ID, Student1's CU Link card UID
Student2's CU Link card UID
Coupon ID, both students' CU Link card UID
success
success
quit the page

Figure 32 UML sequence diagram showing the data exchange of exchanging coupons between students.

Use QR code to view coupons

This step is only required for students who do not have a NFC equipped smartphone. A kiosk with NFC function equipped is available so that students can tap the student card to the kiosk. Then the kiosk gives a QR code to user immediately. The user can use their smartphone with our app installed to scan the QR code, and view the coupons.

When the user taps the student card, the corresponding card UID is sent to the database. The database records the card UID and generate a set of numbers. This number is converted to QR code by the kiosk. When the student scans the QR code, a reverse process is done, which means the QR code is converted back to a set of numbers. The database will match the record with the student card UID and send back this information to the app.

The following flowchart shows the validation process after scanning a QR code.

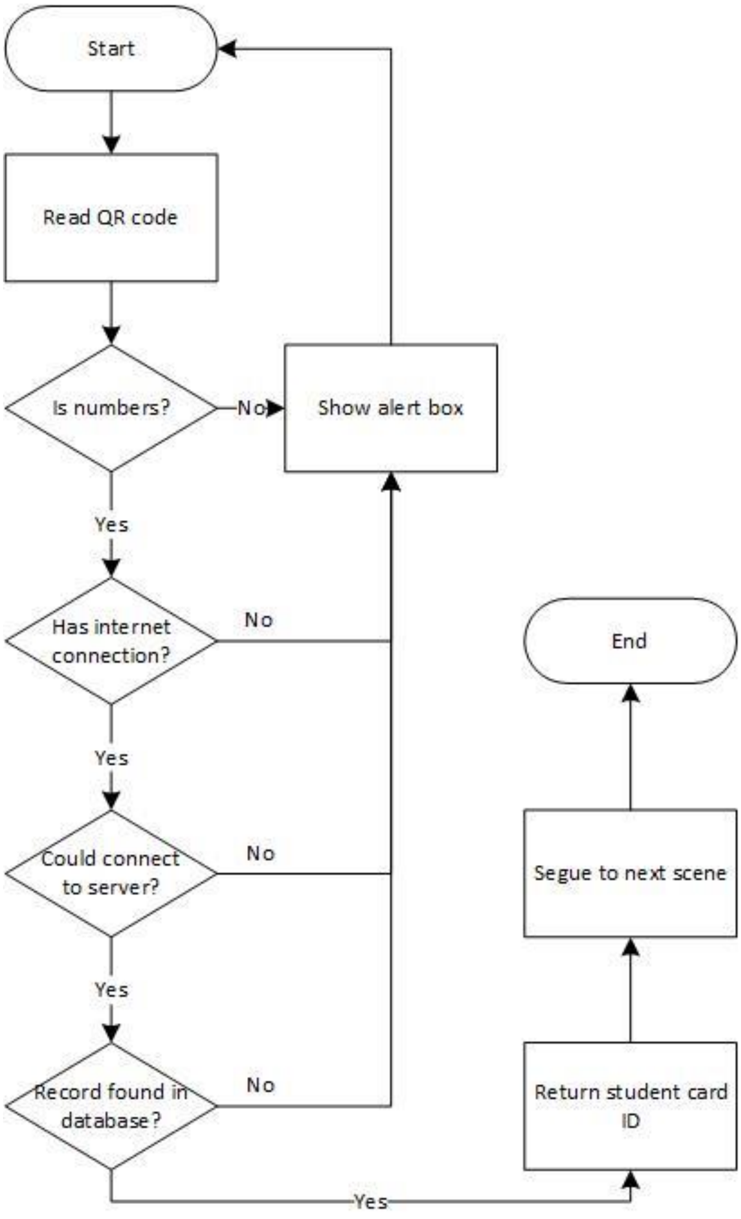


Figure 33 Flowchart showing validation process after scanning a QR code

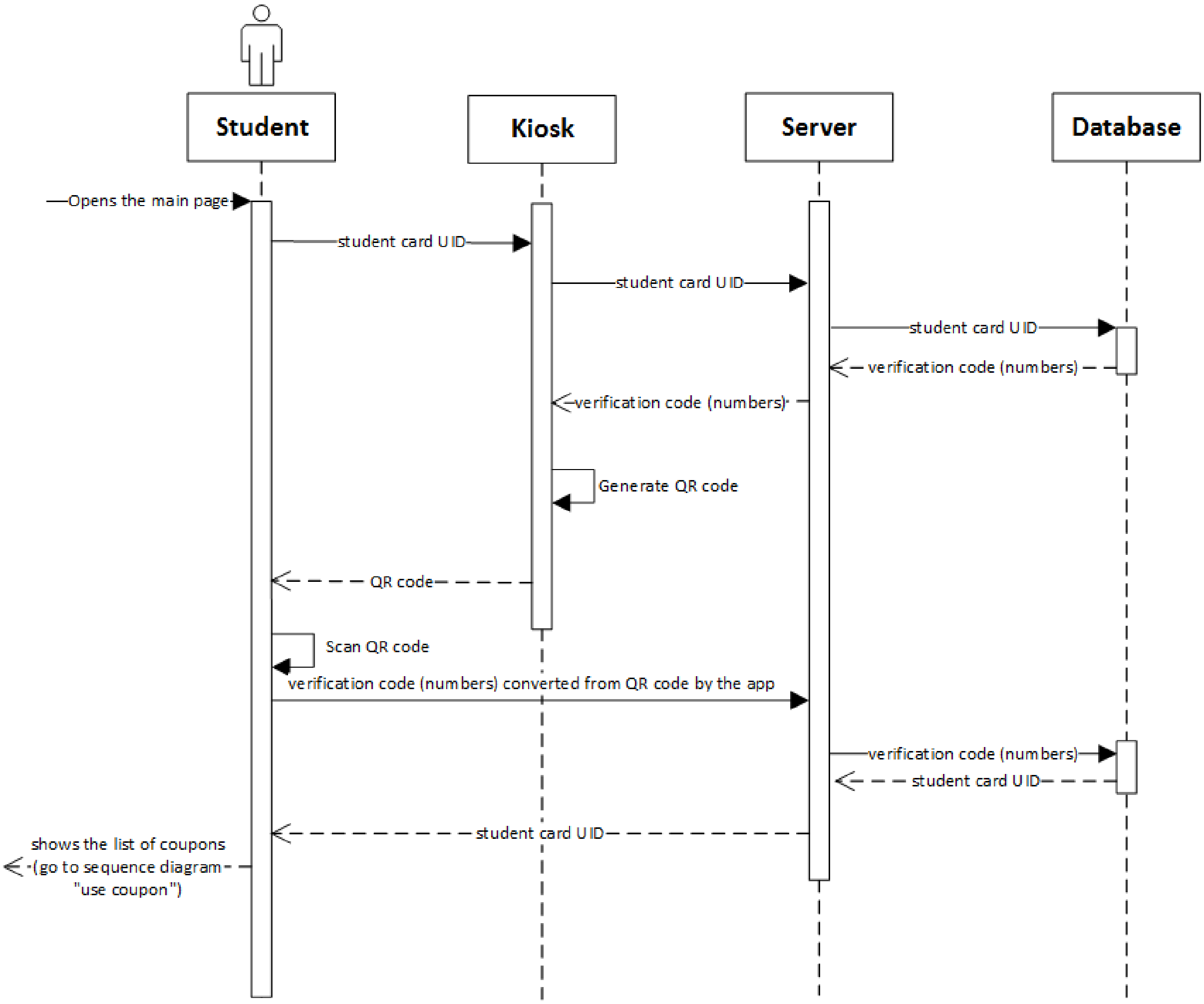

Student
Kiosk
Server
Database
Opens the main page
student card UID
student card UID
student card UID
verification code (numbers)
verification code (numbers)
Generate QR code
QR code
Scan QR code
verification code (numbers) converted from QR code by the app
verification code (numbers)
student card UID
student card UID
shows the list of coupons (go to sequence diagram "use coupon")

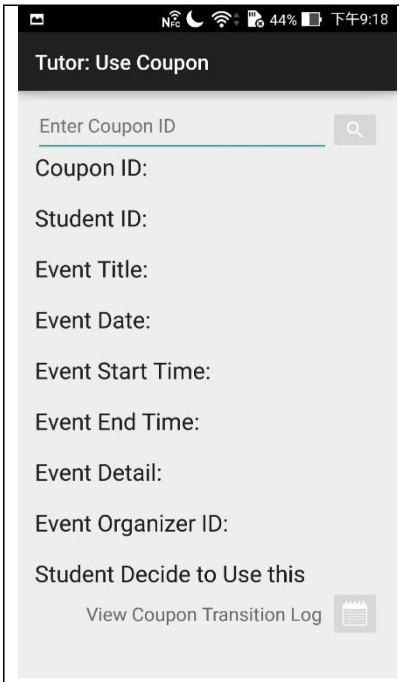
Figure 34 UML sequence diagram showing using QR code to check student's own coupons.

4.2.5 User Interface Implementation

4.2.5.1 Android application

The user interface aims for simplicity. Note that the image for reference only. The actual user interface may have slight differences based on the model and dimension of the device. Some sample interface is shown below.

The image on left hand side below shows the main page which the tutor can input coupon ID and search. If the search is success, related information is shown in the middle part. The middle screenshot shows a wrong input, such coupon does not exist. A toast is shown on the bottom. If the coupon code is correct, the image on the right hand side is shown. Details are available. A note in green color is displayed to prompt the user that they can tap the NFC card in order to consume the coupon.



Tutor: Use Coupon

Enter Coupon ID

Coupon ID:

Student ID:

Event Title:

Event Date:

Event Start Time:

Event End Time:

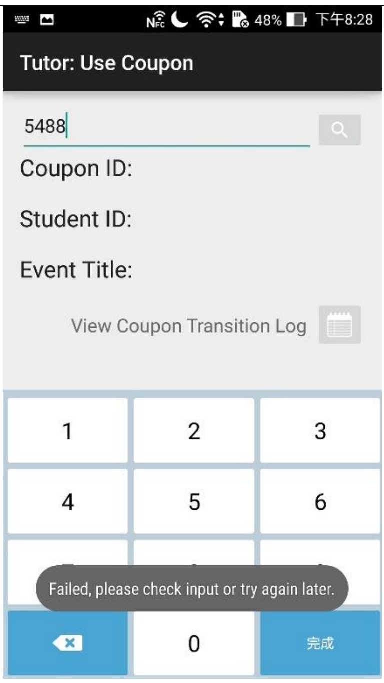
Event Detail:

Event Organizer ID:

Student Decide to Use this

View Coupon Transition Log

Figure 35 Main page of tutor's app



Tutor: Use Coupon

5488

Coupon ID:

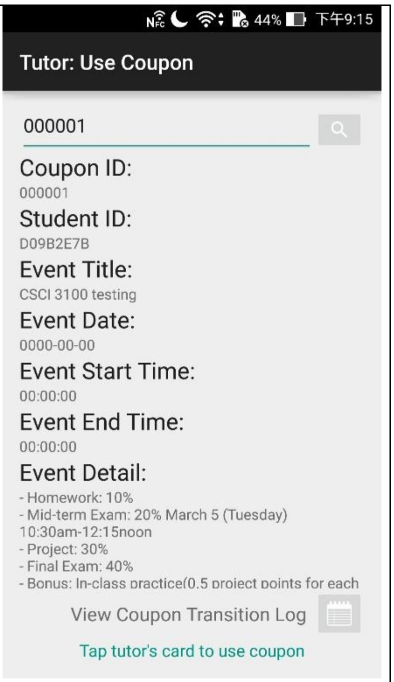
Student ID:

Event Title:

View Coupon Transition Log

Failed, please check input or try again later.

Figure 36 Simulating wrong input



Tutor: Use Coupon

000001

Coupon ID:
000001

Student ID:
D09B2E7B

Event Title:
CSCI 3100 testing

Event Date:
0000-00-00

Event Start Time:
00:00:00

Event End Time:
00:00:00

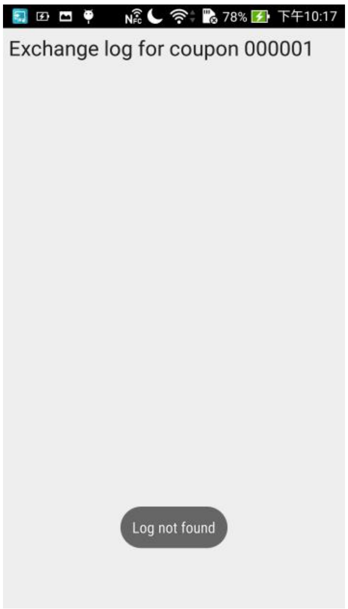
Event Detail:
- Homework: 10%
- Mid-term Exam: 20% March 5 (Tuesday) 10:30am-12:15noon
- Project: 30%
- Final Exam: 40%
- Bonus: In-class practice(0.5 project points for each

View Coupon Transition Log

Tap tutor's card to use coupon

Figure 37 Prompting the user to tap CU Link card

If the user press "view coupon transition log", a log showing coupon transitions is available. If the owner of coupon did not give the coupon to others nor received coupons from other students, "Log not found" is shown; otherwise, a log is available listing out the records.



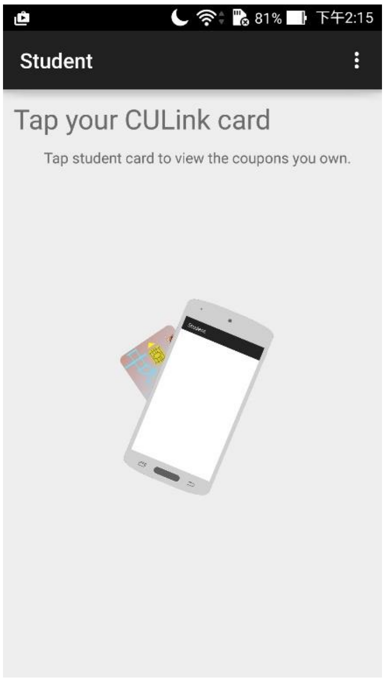
Exchange log for coupon 000001

Log not found

Figure 38 Sample screenshot showing "Log not found"

For other pages, both in student app and teacher app, on button pressed, the toast appears if it is failed.

The first page of the student app request user to tap the CU Link card. Also, the design in the teacher app for tapping CU Link card is similar, either appear as the design below, or as the image shown in Figure 37, a text in green is shown to prompt user to tap the card any time to perform a particular function.



Student

Tap your CULink card

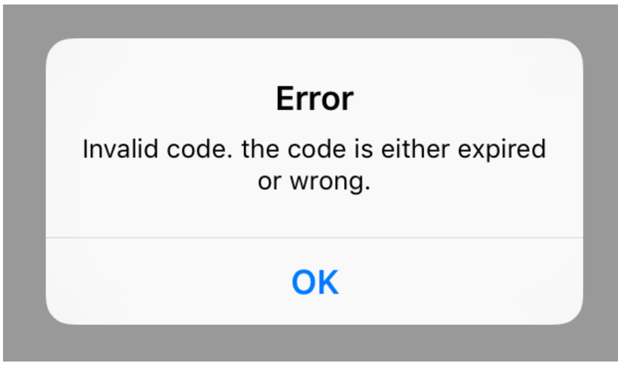
Tap student card to view the coupons you own.

Figure 39 Screenshot prompting the user to tap CU Link card to view the coupons.

Viewing the details of a particular coupon is similar with the previous screenshot that showing the main page of tutor app, except that a search box is omitted. The design of functions buttons is also similar, with an icon and text description next to it. For simplicity, not all of the screenshots showing the design are shown.

4.2.5.2 iOS application

One of the major difference between iOS and Android is that iOS do not have toast. We use dialog box instead. The following examples shows after scanning a QR code.

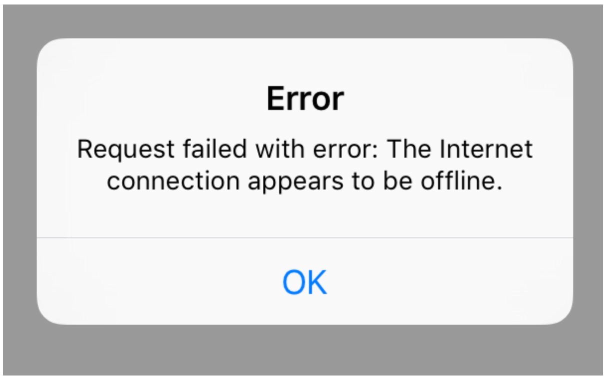


Error

Invalid code. the code is either expired or wrong.

Figure 40 Dialog box showing wrong QR code.

The above dialog box will be shown in the scanned QR code contains non numerical characters, or the code is not found in the databaase.

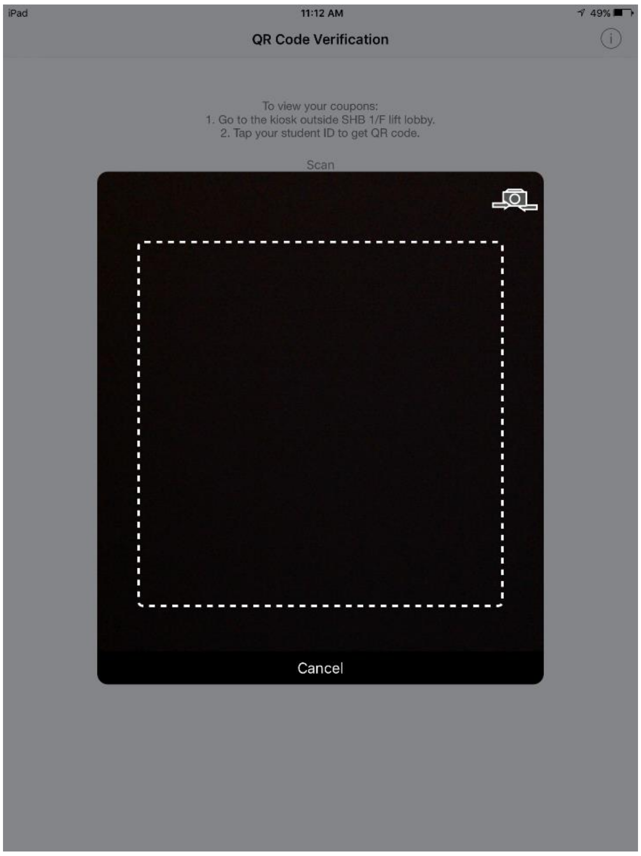


Error

Request failed with error: The Internet connection appears to be offline.

Figure 41 Dialog box showing no internet connection

When the user did not turn on the internet connection, the system cannot search the server for information. Therefore, a dialog box is shown telling the user that the request has failed with a reason.



To view your coupons:
1. Go to the kiosk outside SHB 1/F lift lobby.
2. Tap your student ID to get QR code.

Figure 42 Page for scanning QR code

A swift framework is used for implementing the QR code scanning function. The framework “QRCodeReader.swift” gives a GUI sample for scanning QR code. The pop-up camera box is as shown in the screenshot above is given by the framework. Moreover, there is a change camera button in the top right hand corner. Users can use either the front or back camera to scan the QR code. The programming details will be discussed in the next section, “Programming implementation”.

The design of the app has considered different size of monitors. The height between lines is different for different size of devices. The line spacing is narrower for smaller screens.

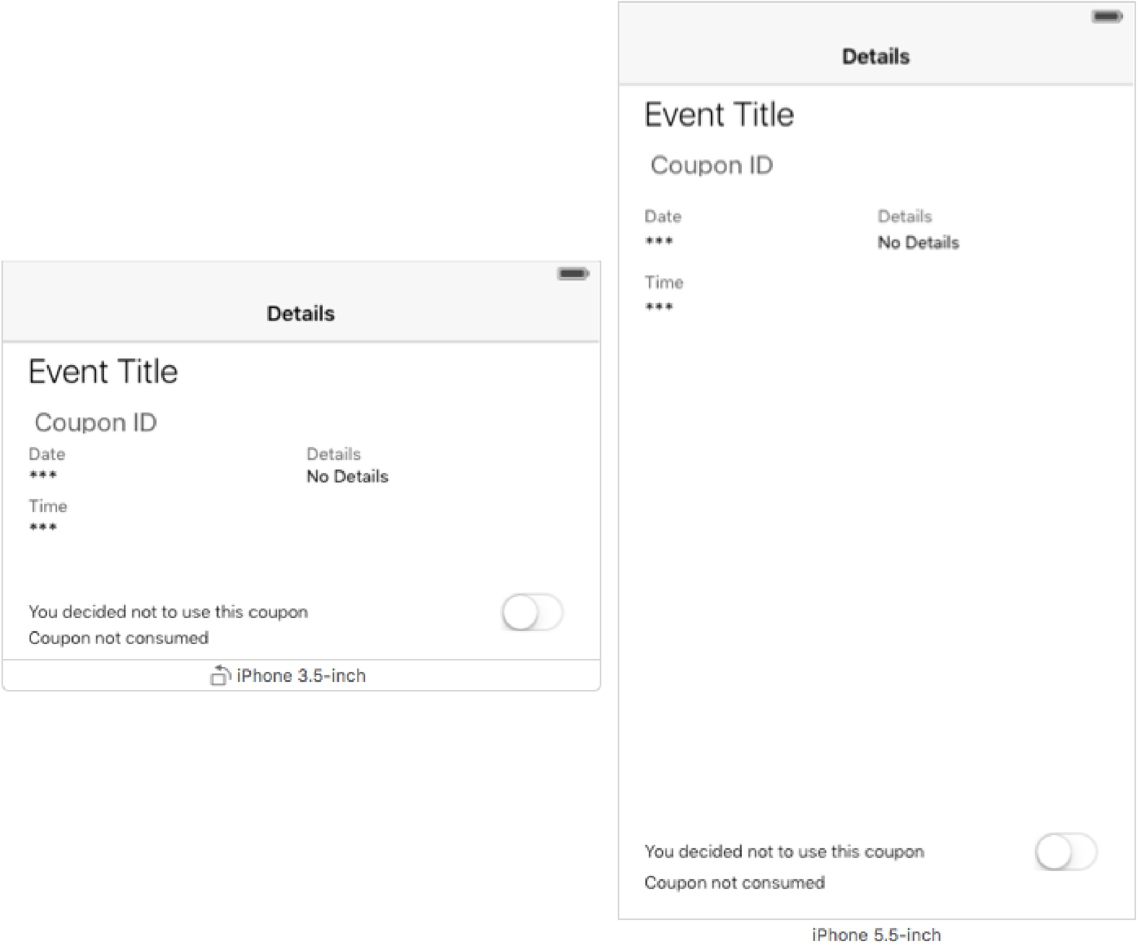


Figure 43 Sample page on different orientation

4.2.6 Programming Implementation

4.2.6.1 Android application

The system mentioned above is implemented by Java. There are some main points including reading UID from NFC card, doing internet connection, manipulating records with server and using toasts to show prompts.

Reading UID from NFC card

A part which is commonly used in our system is NFC scanning. The UID of the CU Link card is read when the user taps his/ her card on the Android phone. Enabling NFC permission is needed, by adding the following line to AndroidManifest.xml

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

The UID is get and stored at “scanNFC” when the CU Link card is tap on the device. The data got is originally a byte array, we then convert the array of bytes to hexadecimal string. As the back side of CU Link card contains the hexadecimal string, therefore converting to hexadecimal string enable us to verify whether the data read is correct. The following code in “Use_Coupon.java” in the tutor app shows the action performed after scanning the UID.

```
protected void onNewIntent(Intent intent) {
    if (intent.getAction().equals(NfcAdapter.ACTION_TAG_DISCOVERED)) {
        scanNFC = ByteArrayToHexString(intent.getByteArrayExtra
                  (NfcAdapter.EXTRA_ID));
    }
        tutorID = scanNFC;
        new UseCoupon().execute();
…
}
```

The function of the above part is to get the UID in the NFC, then consume the coupon. Actually not only CU Link card, the UID all NFC cards could be read.

Internet connection

Android application does not allow user to do internet related task in the main thread, so instead a new thread is created for doing these tasks. AsyncTask is simpler to manipulate, it allows user to “perform background operations and publish results on the UI thread” [11], according to Android studio. AsyncTask has 4 steps, the important one is “doInBackground()”, which is the task, usually uploading or downloading data. Another one is “onPostExecute(Result)”, the data downloaded is displayed after execution of the previous step.

Manipulating records

First of all, AsyncTask is created, then an http request is made to get the coupon transaction log, then followed by a loop to put the record into an array. The related codes are shown below. These tasks are done in the background. 3 examples are shown to describe the process of submitting and getting records from/ to the database.

The following code shows a view of coupon transition log of a particular coupon ID. The variable “success” is used for indicating whether there is successful return.

```java
JSONObject json = jParser.makeHttpRequest(url_coupons_log, "POST", params);

try {
    // Checking for SUCCESS TAG
    int success = json.getInt(TAG_SUCCESS);

    if (success == 1) {
        coupons = json.getJSONArray(TAG_COUPONS);

        for (int i = 0; i < coupons.length(); i++) {
            JSONObject c = coupons.getJSONObject(i);
            String fromSID = "From Student ID " + c.getString(TAG_FROMSID);
            String toSID = "To Student ID " + c.getString(TAG_TOSID);
            String tTime = "Time : " + c.getString(TAG_TRANTIME);

            // creating new HashMap
            HashMap<String, String> map = new HashMap<String, String>();

            // adding each child node to HashMap key => value
            map.put(TAG_FROMSID, fromSID);
            map.put(TAG_TOSID, toSID);
            map.put(TAG_TRANTIME, tTime);

            // adding HashList to ArrayList
            couponList.add(map);
        }
    } else {
        existLog = 0;
    }
} catch (JSONException e) {
    e.printStackTrace();
}
```

The following example search a particular coupon, which is available in the tutor app. For the string variable “deleteResult”. It is used for storing the fail message. The message is then shown using toast.

```java
JSONObject json;
json = jsonParser.makeHttpRequest(url_use_coupon, "POST", params);

try {
    int success = json.getInt(TAG_SUCCESS);
    if (success == 1) {
        deleteResult = "Coupon successfully used.";
        findResult = "";
        foundCouponID = "";
        foundStudentID = "";
        foundEventTitle = "";
        foundEventDate = "";
        foundEventStartTime = "";
        foundEventEndTime = "";
        foundEventDetail = "";
        foundEventOrg = "";
        foundUsed = "";
        foundChecked = "";
        deleteSuccess = 1;
    } else {
        deleteResult = "Failed, please check input or try again later. " +
json.getString(TAG_MESSAGE);

        deleteSuccess = 0;
        // failed to create coupon
    }
} catch (JSONException e) {
    e.printStackTrace();
}
```

The following example is the part of giving coupons from one student to another. The UID of 2 NFC cards (sender and receiver) and the coupon ID which the user selected to give out is sent to server. Again, the “success” variable is for indicating whether there is successful return from server.

```
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("fromStudentID", fromStudentID));
params.add(new BasicNameValuePair("fromCouponID", fromCouponID));
params.add(new BasicNameValuePair("toStudentID", toStudentID));

JSONObject json;
json = jsonParser.makeHttpRequest(url_exchange_coupon, "POST", params);

try {
    int success = json.getInt(TAG_SUCCESS);

    if (success == 1) {
         exchangeResult = "Coupon given successfully.";
         exchangeSuccess = 1;
    } else {
         exchangeResult = "Failed, please check input or try again later. " +
json.getString(TAG_MESSAGE);
            exchangeSuccess = 0;
        }
} catch (JSONException e) {
        e.printStackTrace();
}
```

For other http requests, their implementations are similar.

Generating QR Code

The QR code is generated using the QR code writer. Note that the “gotTempID” is the string used to convert to QR code. The following code is implemented in the app of the kiosk.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_gen_temp_login_id);

    // getting studnet ID from intent
    Intent i = getIntent();
    scanStudentID = i.getStringExtra("KEY_SID");

    new GenID().execute();

    while (findResult.isEmpty()) {
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    ((TextView) findViewById(R.id.displayTempID)).setText(gotTempID);

    //gen QR code
    QRCodeWriter writer = new QRCodeWriter();
    try {
        BitMatrix bitMatrix = writer.encode(gotTempID, BarcodeFormat.QR_CODE, 512,
512);
        int width = bitMatrix.getWidth();
        int height = bitMatrix.getHeight();
        Bitmap bmp = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                bmp.setPixel(x, y, bitMatrix.get(x, y) ? Color.BLACK : Color.WHITE);
            }
        }
        ((ImageView) findViewById(R.id.imageQRCode)).setImageBitmap(bmp);

    } catch (WriterException e) {
        e.printStackTrace();
    }

}
```

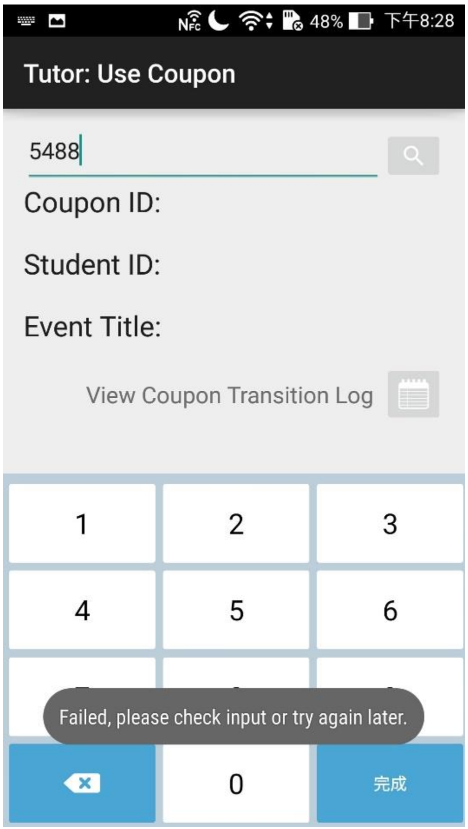
Scanning QR code

For devices without NFC function, the scanning QR code function is the only way for them to view the coupons. There is a “scan” button on the main page. When the user presses this button. QR code could be scanned and sent to the server. In order to speed up the development process, we rely on a tool to scan the QR code. The system will check if the user has the tool “ZXing Barcode Scanner”. If not, a dialog box is shown to prompt the students to download it from Google play, else, this software is triggered for scanning. After scanning the QR code. The code will return back to our system. The following code shows the function of pressing the “scan button”.

```
public void onClick(View v) {
    try {
        Intent intent = new Intent("com.google.zxing.client.android.SCAN");
        //Scan QR code
        intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
        intent.putExtra("SCAN_MODE", "SCAN_MODE");
        // call ZXing, and return 1 after scan
        startActivityForResult(intent, 1);
    } catch (ActivityNotFoundException ex) {
        new AlertDialog.Builder(tempCodeLogin.this)
                .setTitle("ZXing is required")
                .setMessage("Are you sure want to download 'ZXing Barcode 
Scanner' from Google Play?")
                .setPositiveButton(android.R.string.yes,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                 // if zxing is not installed, start Google Play and show zxing
                        Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("market://details?id=com.google.zxing.client.android"));
                        startActivity(intent);
                    }
                })
                .setNegativeButton(android.R.string.no,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        // do nothing
                    }
                })
                .setIcon(android.R.drawable.ic_dialog_alert)
                .show();
    }
}
```

Using Toasts

Using toasts is a simple and effective method to prompt user some information, for instance success message and fail message. Take the previous code as example. The toast will show “Failed, please check input or try again later.” if the coupon ID is not found on the system.


Tutor: Use Coupon

5488

Coupon ID:

Student ID:

Event Title:

Failed, please check input or try again later.

Figure 44 A sample screenshot showing the usage of toast

4.2.6.2 iOS application

The iOS developer library has stated the roles of Model-View-Controller which should be maintained in the whole app.

Model

Model "encapsulate the data and define the logic and computation that manipulate and process the data", according to iOS Developer Library [12]. "Homework coupon" is the model of this app. Therefore, a file "Coupon.swift" is created with the following structure.

Properties	`var eventTitle: String`
	`var couponID: String`
	`var use: String`
	`var eventDate: String`
	`var eventStartTime: String`
	`var eventEndTime: String`
	`var eventDetail: String`
	`var organizerID: String`
	`var remarks: String`
	`var checked: String`
	`var checkBy: String`

Initialization	`init?()`
	`init?(_ eventTitle1: String, couponID:String, use: String)`

Get Functions	`func getEventTitle() -> String`
	`func getCouponID() -> String`
	`func getUse() -> String`
	`func getEventDate() -> String`

	func getEventTimeWithFormat() -> String func getEventDetail() -> String func getOrganizerID() -> String func getRemarks() -> String func getChecked() -> String func getCheckBy() -> String
Set Functions	func setCoupon(eventTitle: String, couponID:String, use: String, studentID: String, eventDate: String, eventStartTime: String, eventEndTime: String, eventDetail: String, organizerID: String, remarks: String, checked: String, checkBy: String) -> Void typealias Handler = (status: AnyObject) -> Void func setUse(use: Int, handler: Handler) -> Void

View Controller

In the iOS app, each scene has one view controller. Most of the segues, delegates, outlets and actions are defined here. The following pseudo code shows how coupons are downloaded and listed in a table view.

```
override func viewWillAppear(animated: Bool) {
    coupons.removeAll()
    code.download(studentID){ (status) -> Void in
        let json = JSON(status)
        if(no json error){
            if (json["success"].intValue == 1){
                if let coupons = json["coupon"].array {
                    for coupon in coupons {
                        //insert coupons
                        Assign a json data to constant
                        let couponInsert = Coupon(eventTitle,
                                                  couponID: couponID,
                                                  use: use)!
                        self.coupons += [couponInsert]
                    }
                    self.couponTableView.reloadData()
                }
            }else{
                Display an alert box showing no coupons
                Unwind to last scene
            }
        }else{
            Display an alert box showing request failed
            Unwind to last scene
        }
    }
}
```

Frameworks

To speed up the development process, few frameworks that are available in GitHub are used, namely Alamofire, SwiftyJSON, and QRCodeReader.swift. Their functions are listed below:

Name of framework	Function
Alamofire	It provides a library to simplify the programming effort in doing HTTP requests and responses. The iOS app send POST request to the server and the server returns some data.
SwiftyJSON	With this framework, parsing JSON data become easier. Although there is library for parsing JSON in Swift but it is not user friendly. JSON is implicit about data types but Swift require explicit typing. Moreover, SwiftyJSON has done optional wrapping. [13]
QRCodeReader.swift	It provides a GUI for scanning QRCode using AVFoundation. The pop-up camera window for scanning QRCode is done by this framework.

The following example shows the usage of Alamofire and SwiftyJSON in the model "Coupon.swift". One of the functions in this model is to update the Boolean value in the database when the user clicks the button claim that he/ she want to use/ do not want to use the coupon.

Using "`Alamofire.request`" and specifying the HTTP method, URL and parameters, a HTTP request is done.

The function "`JSON(data)`" parse JSON data into array.

```
typealias Handler = (status: AnyObject) -> Void

func setUse(use: Int, handler: Handler) -> Void{
    Alamofire.request(.POST,
             "http://appsrv.cse.cuhk.edu.hk/~plho3/mark_used_coupon.php",
             parameters: ["couponID": self.couponID])
        .responseJSON { response in
            switch response.result {
            case .Success(let data):
                let json = JSON(data)
                if (json["success"].intValue == 1){
                    self.use = String(use)
                    handler(status: data)
                }else{
                    handler(status: data)
                }
            case .Failure(let error):
                handler(status: error)
            }
    }
}
```

The following pseudocode shows how the QR code reader is called. The camera could be called by setting the model presentation style, delegate and specifying the action after scanning is complete. Finally call the camera pop up window.

```
@IBAction func scanAction(sender: AnyObject) {
    if QRCodeReader.supportsMetadataObjectTypes() {
        reader.modalPresentationStyle = .FormSheet
        reader.delegate               = self

        reader.completionBlock = { (result: QRCodeReaderResult?) in
            if let result = result {
                if let _ = Int(result.value){
                    Alamofire.request(send HTTP post request to
                                     specified URL with the QR code content)
                        .responseJSON { response in
                            switch response.result {
                            case .Success(let data):
                                let json = JSON(data)
                                if (json["success"] is true){
                                    segue to next scene with json["studentID"]
                                }else{
                                    let message = json["message"].stringValue
                                    display an alert box with error message
                                }
                            case .Failure(let error):
                                display an alert box with error message
                            }
                    }
                }
            }
        }

        //Present the QR code reader camera pop up window
        presentViewController(reader, animated: true, completion: nil)
    }
    else {
        display an alert box with error message
    }
}
```

5. Software Deployment in Real Scenario

5.1 Method of Deployment

The smartphone app is using in course CSCI3100 Software Engineering, Spring, 2016. After discussing with Prof. Lyu, the course instructor, also our supervisor, it is decided that the physical coupon and the virtual coupon (i.e. our software system) should be carried out at the same time, in order to reduce the risk of unavailability in case of system failure. The details of this software system is announced to students during the first lesson. The leaflet is also posted on the course website. When a student receives a coupon, a leaflet is also given showing the download location and instruction of using the app. Also, the leaflet could be download from the course web site. The leaflet is attached in the appendix.

When the student wants to use the homework coupon, he/ she should clip the physical coupon and write down the code of the virtual coupon at the same time, also stating which question he/ she wants to exempt from doing.

If students have a NFC supported Android phone. He/ she can use the student app that is available in Google play since the first week of lesson. The kiosk is also available later to cater the students who do not have a NFC supported smartphone.

When the tutor receives the homework, he/ she will send the virtual coupon code to us through email. We have created an email account for this course handling all the enquiries and communicate with tutors regarding this project. We will check and validate the code. We will contact the student in case of any problems.

5.2 Deployment Progress

The course CSCI3100 starts at Monday. This project aims to start service since the first lesson starts. The following graph shows the deployment progress.

The Android app started development since semester one. The iOS app started development since we have confirmed using the kiosk. Since iOS do not support NFC function, the iOS app must exist together with the kiosk.

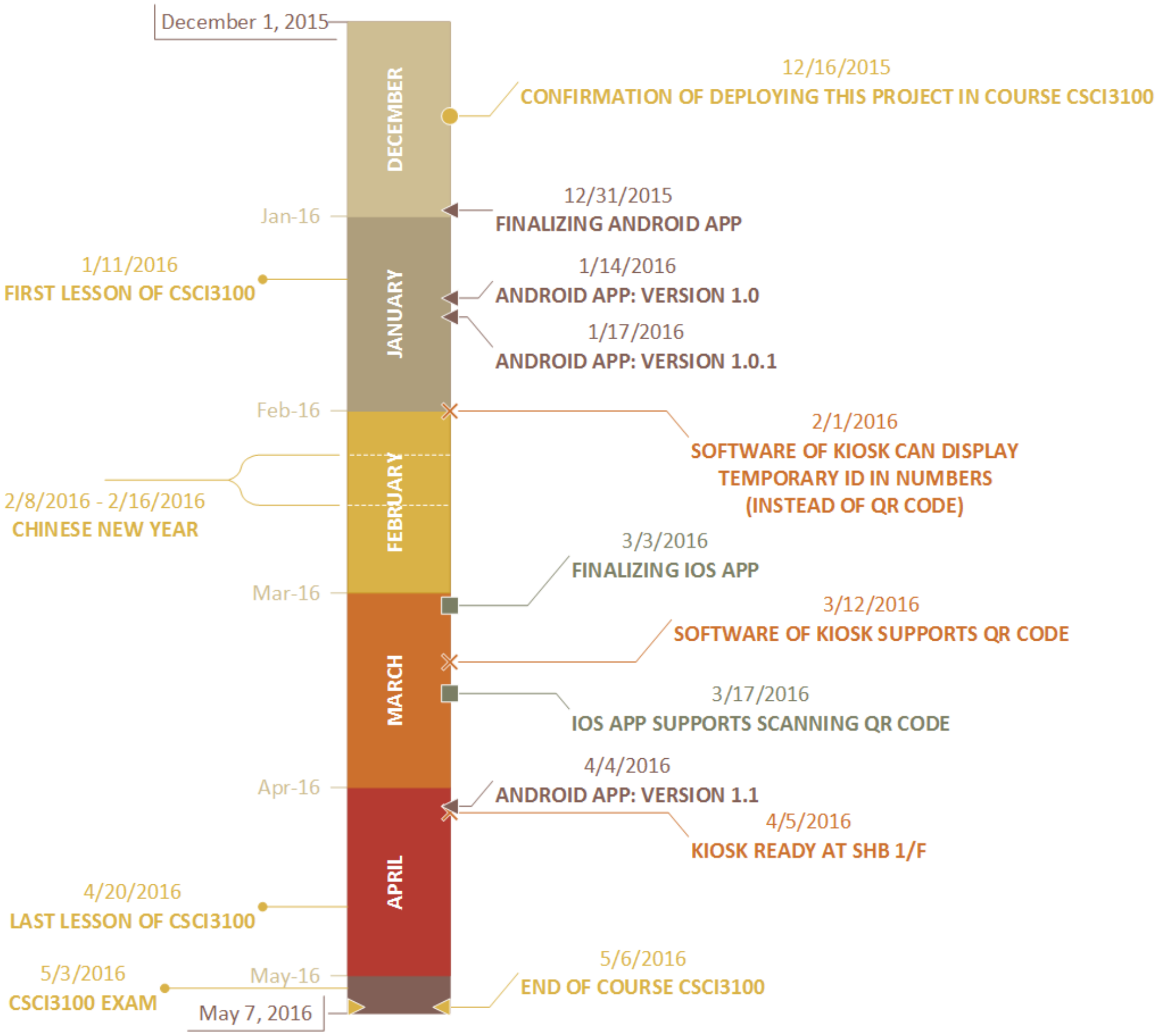


Figure 45 Project schedule

5.3 Software Publishing

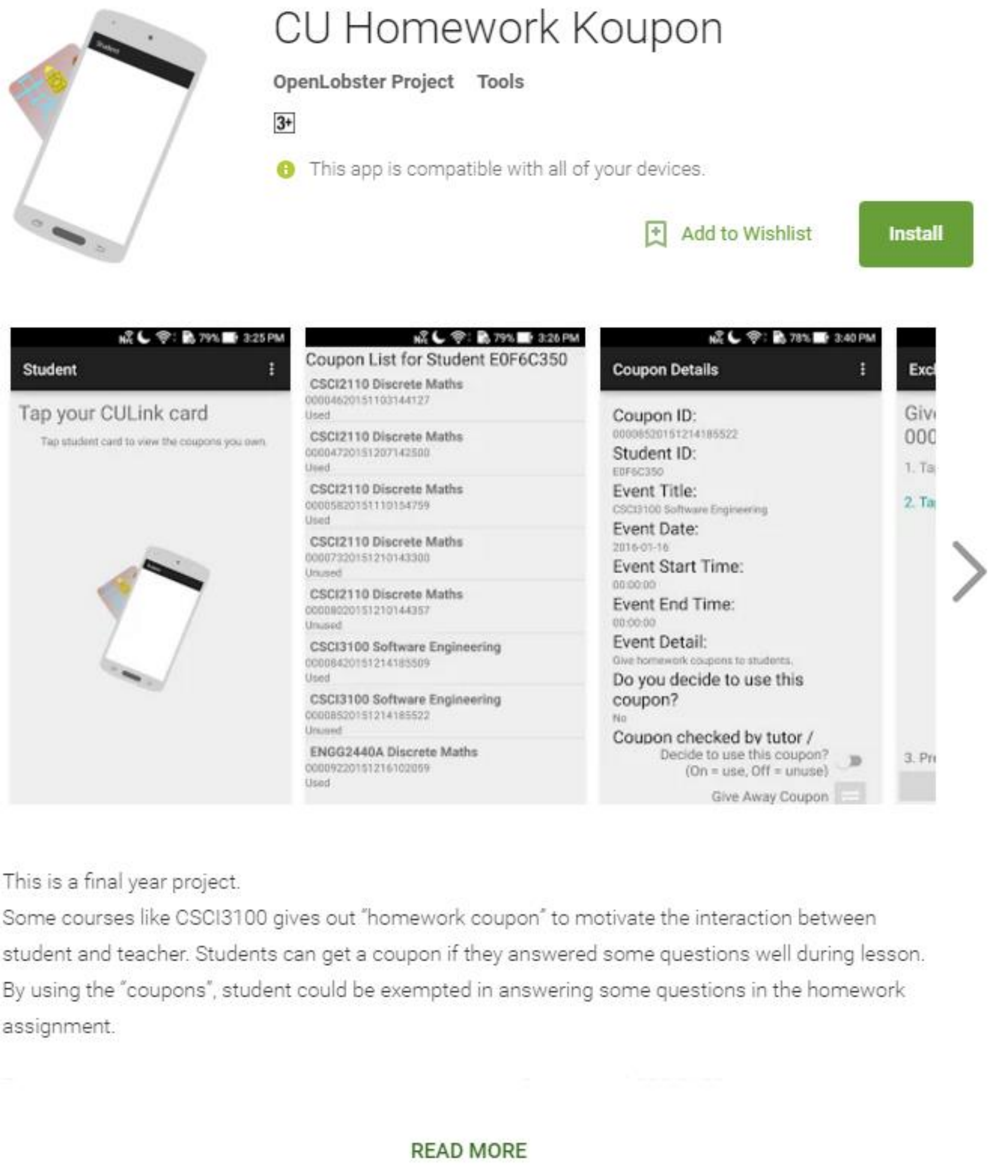
Student App

The Android App that finished at the first semester is published on Google Play. Google Play allow users to publish their own apps within a day is its advantage. Our app takes approximately 4 hours to publish. Therefore, we could improve our app promptly when there are crash reports.

Name: CU Homework Koupon

Classification in Google Play: Tools

Link: https://play.google.com/store/apps/details?id=fyp.lyu1501.student&hl=en



CU Homework Koupon

OpenLobster Project Tools

3+

This app is compatible with all of your devices.

Coupon List for Student E0F6C350

CSCI2110 Discrete Maths
000046201511103144127
Used

CSCI2110 Discrete Maths
000047201512071425OO
Used

CSCI2110 Discrete Maths
000058201511110154759
Used

CSCI2110 Discrete Maths
000073201512101433OO
Unused

CSCI2110 Discrete Maths
000080201512101443S7
Unused

CSCI3100 Software Engineering
000084201512141855O9
Used

CSCI3100 Software Engineering
000085201512141855 22
Unused

ENGG2440A Discrete Maths
000092201512161020S9
Used

Coupon Details

Coupon ID:
000085201512141855 22

Student ID:
E0F6C350

Event Title:
CSCI3100 Software Engineering

Event Date:
2016-01-16

Event Start Time:
00:00:00

Event End Time:
00:00:00

Event Detail:
Give homework coupons to students.

Do you decide to use this coupon?
No

Coupon checked by tutor /
Decide to use this coupon?
(On = use, Off = unuse)

Give Away Coupon

This is a final year project.
Some courses like CSCI3100 gives out "homework coupon" to motivate the interaction between student and teacher. Students can get a coupon if they answered some questions well during lesson. By using the "coupons", student could be exempted in answering some questions in the homework assignment.

READ MORE

Figure 46 Screenshot in Google play

The following table shows the version control of the student Android app with explanations.

Version	Uploaded on	Function
1 (v1.0)	Jan 14, 2016	Same as the production in the first term. Users can view, give away and use coupons
2 (v1.0.1)	Jan 17, 2016	An about page is added
3	Not published	Testing version, support using number string generated by kiosk to view the coupon.
4 (v1.1.1)	Apr 4, 2016	Devices which not support NFC function could also be used. Users could view the coupons by getting a QR code from the kiosk.

The iOS app finished development at late March 2016. Since uploading to iTunes takes weeks of time, so we decided to upload this app to CUHK mobile app store.

Version	Uploaded on	Function
1 (v1.0)	Pending	Users could view the coupons by getting a QR code from the kiosk.

Kiosk

The software of the kiosk is an Android app. The main function is to provide a QR code for user to scan when they tap their own student card. So that users can use their smartphone to scan the QR code and read the coupons.

The kiosk is originally put in the VIEW lab and it was not in use. We take away the original monitor in the kiosk and replaced with an Android smartphone with NFC function. Since the

NFC receiver is at the back of the smartphone, it is impossible to show the monitor and allow user to tap the smartphone at the same time in the kiosk. We connect the NFC pin in the phone body with a coil, acting as a NFC antenna. Thus produce magnetic induction when the user taps the NFC card.

The software is developed during February and March. The kiosk is located at SHB 1/F lobby near SHB101 from April 5, 2016, right after the Android app version 4 has published.

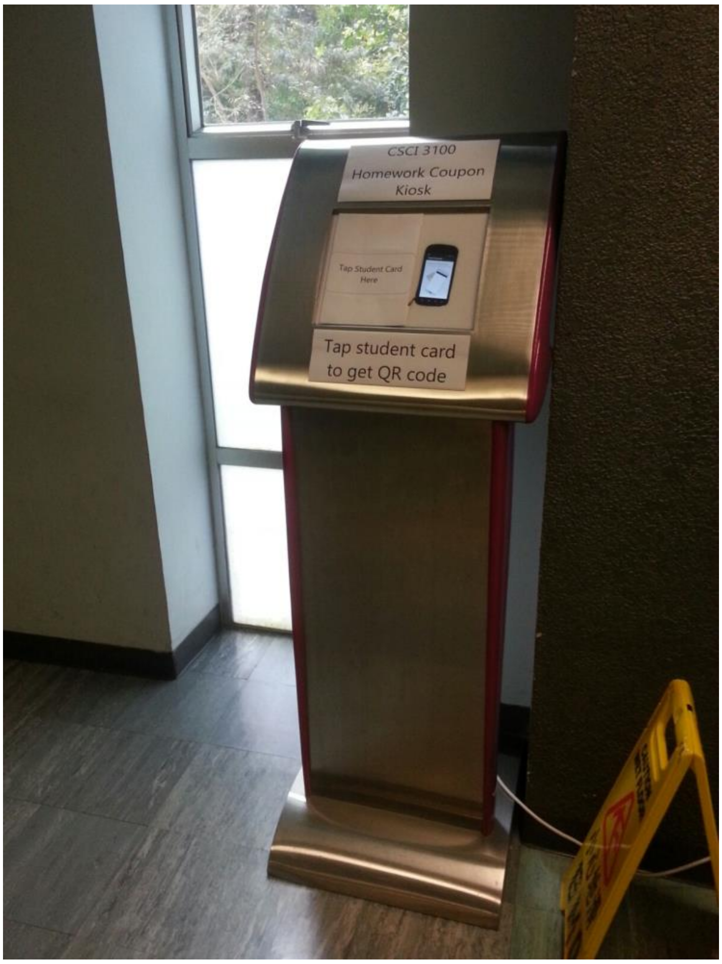
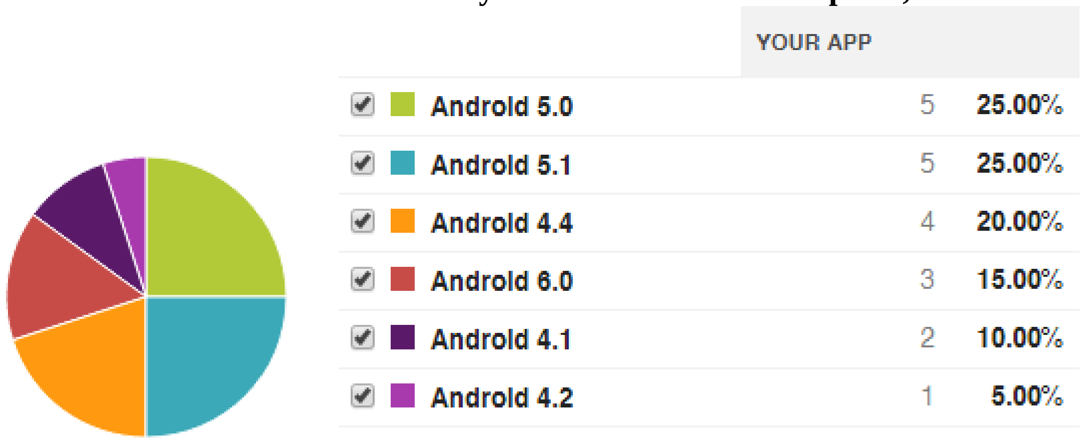


Figure 47 Image of the kiosk

5.4 Apps Statistics

Some statistics given by Google Play regarding the app is shown below:

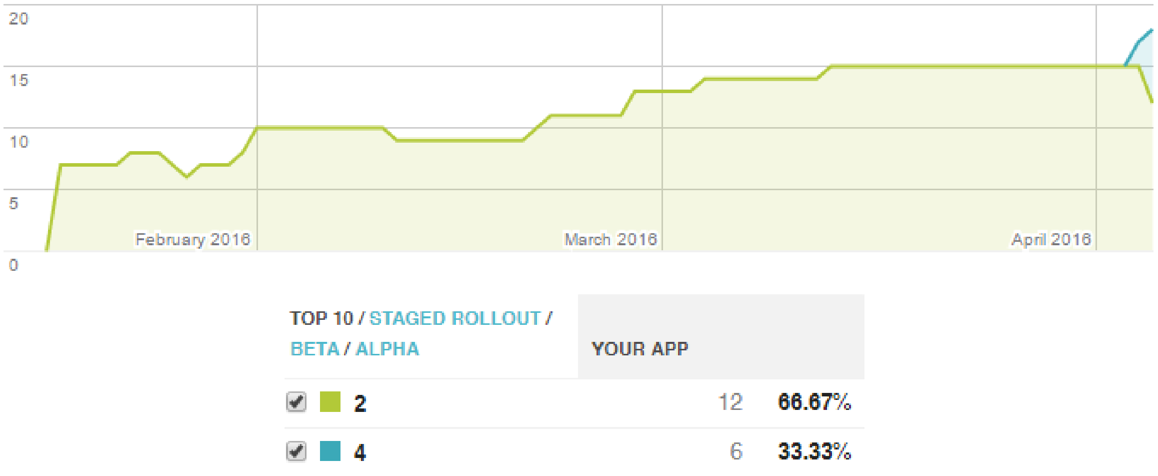
Total installs counted by Android version as of April 5, 2016



			YOUR APP	
☑	🟩	Android 5.0	5	25.00%
☑	🟦	Android 5.1	5	25.00%
☑	🟧	Android 4.4	4	20.00%
☑	🟥	Android 6.0	3	15.00%
☑	🟪	Android 4.1	2	10.00%
☑	🟪	Android 4.2	1	5.00%

Figure 48 Total installs counted by Android version as of April 5, 2016.

Current installs by app version as of April 5, 2016



		TOP 10 / STAGED ROLLOUT / BETA / ALPHA	YOUR APP	
☑	🟩	2	12	66.67%
☑	🟦	4	6	33.33%

Figure 49 Current installs by app version as of April 5, 2016

The course CSCI3100 has 225 students. According to the data in our database as of April 12, 2016, a total number of 119 coupons has been issued to 30 students. Current install numbers is 18. It could be estimated that at most 60% of students who got the coupon has installed the app.

There are 4 coupons transited to other people.

8 coupons are consumed. According to the past experience, most of people will use their homework coupons in the last homework. The last homework will due on late April or early May, it explains the low usage rate of the coupons.

Moreover, the kiosk has published on early April. Until April 12, there are 74 tap records, including 38 different CU Link cards. There are 6 students who got coupons tapped their card on kiosk. The overall usage rate is satisfactory.

5.5 System Failure and Crash Report

The NFC function in the kiosk was not working on April 7, 2016. The problem occurs because the NFC pin on the phone body and our NFC antenna did not contact well. To solve this problem, the connection method used to connect the Android phone and the antenna is modified, from using two thin iron wire to using copper electricity wire and soldering them with the NFC antenna.

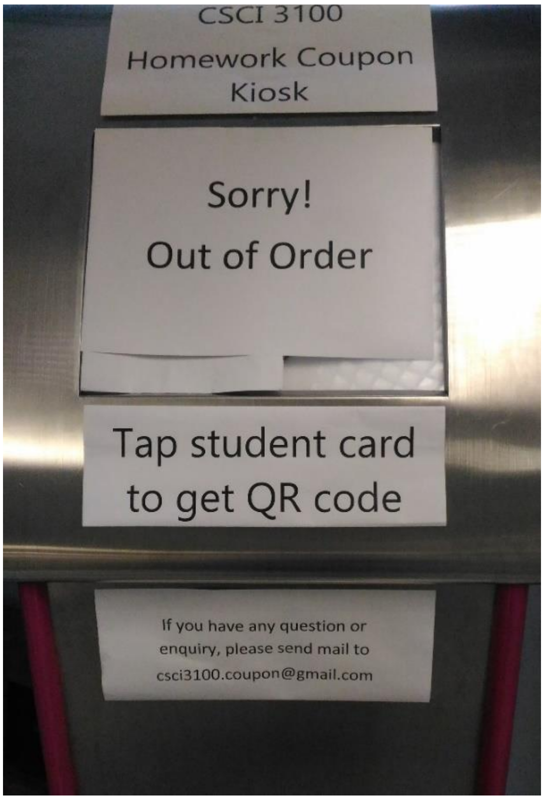

CSCI 3100
Homework Coupon
Kiosk
Sorry!
Out of Order
Tap student card
to get QR code
If you have any question or
enquiry, please send mail to
csci3100.coupon@gmail.com

Figure 50 The kiosk was out of order.

According to Google Play, there are no crash reports.

6. Limitation and difficulties

6.1 NFC security

We rely on using the UID in the NFC card as the token, so we hypothesize that other types of card could also be used. When we try to use other cards apart from CU Link card, however, we occasionally found that a particular NFC card could not be used, which is the EZ-link card from Singapore. The problem is that EZ-link card generates random UID on every touch. Therefore, this type of card cannot be used in our system. The following table shows the type of card we have tested:

Cards that are tested	Tag Type	Technology	UID randomness
CU Link card	ISO 14443-3A MIFARE	NfcA	Constant on every tap
EasyCard #	ISO 14443-3A MIFARE	NfcA	
Octopus	JIS 6319-4 FeliCa	NfcF	
Pasmo #	JIS 6319-4 FeliCa	NfcF	
EZ-Link #	ISO 14443-4	IsoDep, NfcB	Different on every tap

# These cards have similar applications as the Octopus card in Hong Kong, but available in different countries.

Although our system focus on CU Link card, the above test discovered a limitation on our system.

6.2 CU Link card

The CU Link card contains useful information, for example the unique identification number (UID), student number, college, but most of them are encrypted. Our current approach is using the UID in CU Link card which is not encrypted as a token to recognize the user, while keeping the real name/ student ID of the user unknown.

7. Future works

After two semesters, we have implemented numbers of functions of the student assistance system. Nevertheless, there are some future works could be done to enrich the functionality of the system.

7.1 Develop Windows Phone applications

In smartphone market, Windows phone is the third largest market share and some of the student are using Windows Phone. Moreover, as Microsoft published the library for a Windows Phone NFC reader, applications with NFC function could be developed. With NFC function, teacher, tutor and student application could be used without the aid of kiosk.

7.2 Exchange coupons on non-NFC supported smartphone

In the current system, non-NFC supported smartphone could only view owned coupon and mark the use of coupons with the aid of kiosk. If students need to exchange coupons but they have no NFC supported Android phone, they need to find their classmate to lend NFC Android phone or find tutors.

This problem may be solved by the kiosk, for example, choosing “exchange coupon” on the kiosk and tap the two CU Link on the kiosk, a QR code will be generated for exchanging coupon. Student could scan that QR code in the exchange coupon page on their mobile phone to confirm the operation.

7.3 Simplify the procedure of using homework coupon

Student needs to write down the 20-digit homework coupon ID on their homework in order to use the homework coupon. We might simplify the whole process with the aid of QR code.

For example, student could mark that he/ she will use a particular coupon on their smartphone and type their student ID. Then, student could print out the coupon QR code on a kiosk, which include information about the coupon ID and student ID. Student could just stick that QR code on their homework in order to use the coupon.

Also, this will make checking coupon easier, as tutor could scan the QR code on homework rather than typing the 20-digit coupon ID to their smartphone.

8. Contribution and Reflection

In the following part, the contribution and reflection will be discussed.

Table below shows our division of labor:

Work	Jacky	Jeff
Android App – UI	X	X
Android App – Logic	X	
iPhone App – UI		X
iPhone App – Logic		X
Kiosk - Software	X	X
Kiosk - Hardware	X	X
PHP Server	X	
Database Design	X	
SQL	X	X
Report	X	X
Handling Student Query	X	X
Leaflet		X

In this project, I focus more on the android application, the kiosk and the backend of the system.

For the android application, there are three versions published on google play store.

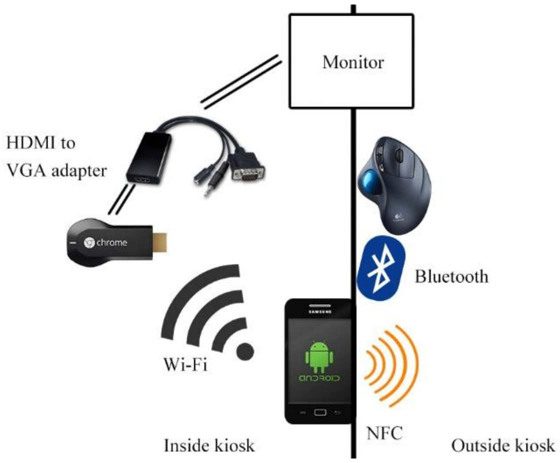
The first version includes the basic function of the student side, which include viewing coupon list, mark used coupon and exchanging coupon. However, this first version application could only download by android phone with NFC function. If a student does not have a NFC android phone, they need to find us after the lecture or leading others NFC android phone.

After publishing the first version of application, some of the student told me they have difficulty to contact us. So, for the second version, the “about us” page is added to the application.

For the third version, which is the current version, a new function – viewing coupon by QR code is added. If the android phone is not NFC supported, it will display the “View coupon by QR code” page. For a NFC supported android phone, user could still select “View coupon by QR code” at the menu. As a result, student have an android phone without NFC function could view and mark use their coupon on their own android phone with the aid of the kiosk.

For the design of kiosk, the goal was allowing user tap their NFC card on it and providing an image screen to user on the same face of a kiosk.

The first design is to use an android phone as a core, stack its back to the case of kiosk for scanning NFC card. For the display, Chromecast will be used to generate HDMI signal, convert it to VGA signal and display it on the original monitor. The original monitor placed on kiosk is a touch monitor, however, the driver of that touch sensor has been lost, a Bluetooth trackball mouse is required as a pointing input device for the android phone.



However, after some research, it is found that there are NFC antennas which could transmit NFC signals from place to place. This greatly simplified the design of kiosk as it solved the main technical problem – how to scan NFC card and providing image screen on the same vertical plan. Moreover, it is found that a NFC antenna could be hand-made. To reduce the cost, a hand-made on have been used finally.

For the software of kiosk, there are two versions. The first version is not published. In that version, after student scan their CU Link card, a 9-digit temporary ID will be shown on the screen on kiosk. Users need to type that 9-digit ID on their phone in order to view their coupon. However, this is not convenience for users. So, the display method of the temporary ID is changed to QR code afterwards, which is the second version of the software.

For the backend part, the database is placed in the CUHK CSE server (appservdb). However, as it is in the CUHK CSE firewall, CSE VPN is required to directly access that database. To solve this problem, the personal web server of CSE student have been used. The mobile applications and kiosk will send the required information to the personal web server, execute database queries in the personal web server, and return result to mobile applications or kiosk.

During the project, I have learned a lesson on both software and hardware development.

On software part, it is found that a lot of changes needs to be made when adding new function in semester 2 due to the software design. At the code written in semester 1, there are not enough comments which explains what is the function of each segment of lines. That makes both Jeff and me taking a lot of time to understand codes and adding new functions. So, in the new written code, we tried to add more comment in it. It is found that by adding comments, it will be more easy to locate bugs and explain what is happening in each line of code.

The next reflection is about the update of applications. User may not update their application even if you make announcement to users. According to the Google statistics, only 65% of the user have updated their application after the new version of the application released for one week. It will be a great problem if there are bugs in the old version and it is fixed in the new version, as lots of users are still using application with bugs. To solve this problem, we may force user to update the application by checking the application version when starting the application.

On the user experience part, it is found that user prefer less clicks, however, user may easily make mistakes if there are too few clicks. It is difficult to strict a balance between easily access and preventing wrong clicks, especially if there exist an exchange coupon function and giving coupon function, which are hardly reversible. And finally, we keep every functions could be access in no more than three clicks or actions.

On the hardware side, it is the first time that we are required to design a kiosk. We have spent lots of time to think about the IO of the kiosk as NFC function and displaying result is a must for the kiosk. In order to make every hardware part can be change easily, we have not fix the connection wire well when making the kiosk and resulting a bad connection between the NFC antenna and the Android phone. We needed to spend extra time for finding out the bad connection, fixing the connection and reinstalling the components into the kiosk. We have learned that we should solder every tightly after we confirm out design to avoid failure on hardware.

9. Conclusion

In the first semester, the technologies of NFC are briefly studied. We have also discovered a special type of NFC card which generates random UID.

A set of application with NFC function, also a server with PHP and MySQL database is developed. Although we experienced a barrier that the ITSC of the university could not give us any information nor access to the encrypted information, we then turn out the using UID as a token to identify users. The functionality of the system is the main point in first semester. Moreover, data security is a crucial yet sophisticated field, we start to notice this issue when we consider how this project could be deployed in a real life situation. Although we faced different minor difficulties in coding, the mobile app could be finished at last.

In the second semester, the homework coupon system is deployed in course CSCI3100 “Software Engineering”. System robustness in one of our focus points as student’s coupons are saved in our system and the exist of coupons may affect the grading of the course. So, we modified the server side script to prevent SQL injection and reduce the system response time. Another focus point is to increase the system completeness. We need to provide supports to students who do not have NFC-equipped Android phone. Therefore, a kiosk is produced and the iOS version application is also developed. Student could access to their coupon with the aid of kiosk and smartphone.

Despite the limitations and difficulties, continuous improvement to the system had been done.

10. Acknowledgements

We would like to express our gratitude to Prof. Lyu Rung Tsong Michael for his guidance to our project. Moreover, we would also like to give thanks to Mr. Edward Yau for giving us technical and hardware support, also exchanging valuable ideas towards the project. Without the help from Prof. Lyu and Mr. Yau, our project must face more difficulties and barriers. Also, the tutors of course CSCI3100 cooperate with us well, and thus the system could run smoothly. They are: Su, Yuxin; Chen, Guangyong; He, Pinjia; Zhu, Fengyuan; Zheng, Jichuan; Zheng, Qingqing; Li, Jian; and Gao, Cuiyun.

11. References

[1] "Configuring Gradle Builds," Android Developers, [Online]. Available: http://developer.android.com/tools/building/configuring-gradle.html.

[2] "phpMyAdmin: Introduction. Supported features," phpMyAdmin, [Online]. Available: https://phpmyadmin.readthedocs.org/en/latest/intro.html.

[3] "What are the operating modes of NFC devices?," NFC Forum, [Online]. Available: http://nfc-forum.org/resources/what-are-the-operating-modes-of-nfc-devices/.

[4] H. N. H. Chan and S. Y. Chan, "Mobile Application Using NFC.," The Chinese University of Hong Kong, [Online]. Available: http://www.cse.cuhk.edu.hk/lyu/_media/students/lyu1301_term1report.pdf.

[5] "Host-based Card Emulation," Android Developers, [Online]. Available: http://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[6] "Description of the database normalization basics," Microsoft, [Online]. Available: https://support.microsoft.com/en-hk/kb/283878.

[7] "Second Normal Form," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Second_normal_form.

[8] "CROSS JOIN opereation," Oracle, [Online]. Available: http://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljcrossjoin.html#rrefsqljcrossjoin.

[9] "PHP: mysql_real_escape_string," The PHP group, [Online]. Available: http://php.net/manual/en/function.mysql-real-escape-string.php.

[10] "mysql_real_escape_string SQL injection," SQLINJECTION.NET, [Online]. Available: http://www.sqlinjection.net/advanced/php/mysql-real-escape-string/.

[11] "AsyncTask," Android Studio, [Online]. Available: http://developer.android.com/reference/android/os/AsyncTask.html.

[12] "Model-View-Controller," Apple Inc., [Online]. Available: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html.

[13] R. Fu, "GitHub - SwiftyJSON," [Online]. Available: https://github.com/SwiftyJSON/SwiftyJSON.

[14] C. Colby, "The Pay-off: Can you lose the plastic and use only Apple Pay or Android Pay?," [Online]. Available: http://download.cnet.com/blog/download-blog/the-pay-off-can-you-lose-the-plastic-and-use-only-apple-pay-or-android-pay/.

12. Appendix

12.1 “CU Homework Koupon” leaflet

This leaflet is given to student who received a coupon and is uploaded to the course webpage.

- The first one is the draft version which used in the first lesson of the class for us to explain this project.
- The second one is used to distribute to students on every lessons.
- The third one is used after the kiosk is ready.

Online Homework Coupons for CSCI3100

Final Year Project, 2015-2016

Quick Start Guide

① smartphone with NFC function

② Download the App here: or https://goo.gl/2vdA25

I don't have a valid smartphone

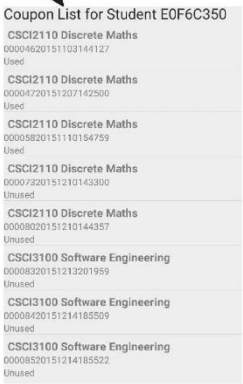
We will be in the class every Monday. Find us to check your coupons.

Check my coupons

1 Open the app, turn on NFC

2 Tap your CU Link card

3 You will see a list of coupons. This should match with the number of physical coupons on your hand.



Coupon List for Student E0F6C350

CSCI2110 Discrete Maths
00004620151103144127
Used

CSCI2110 Discrete Maths
00004720151207142500
Used

CSCI2110 Discrete Maths
00005820151110154759
Used

CSCI2110 Discrete Maths
00007320151210143300
Unused

CSCI2110 Discrete Maths
00008020151210144357
Unused

CSCI3100 Software Engineering
00008320151213201959
Unused

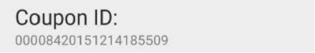
CSCI3100 Software Engineering
00008420151214185509
Unused

CSCI3100 Software Engineering
00008520151214185522
Unused

Use my coupons in homeworks

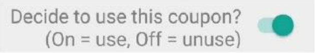
Do step **1-3**

4 Copy the coupon ID from the app to your homework.



Coupon ID:
00008420151214185509

5 Turn on "Decide to use this coupon"



Decide to use this coupon?
(On = use, Off = unuse)

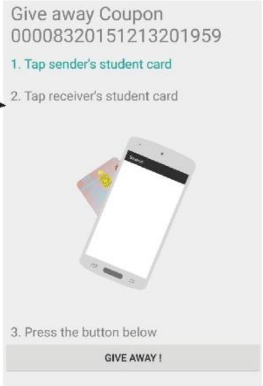
6 Submit homework with the physical coupon AND the coupon ID. **(important!!!)**

Give my coupons to friends

Do step **1-3**

4 Choose a coupon you wish to give away.

5 Follow instructions on the app.



Give away Coupon
00008320151213201959

1. Tap sender's student card

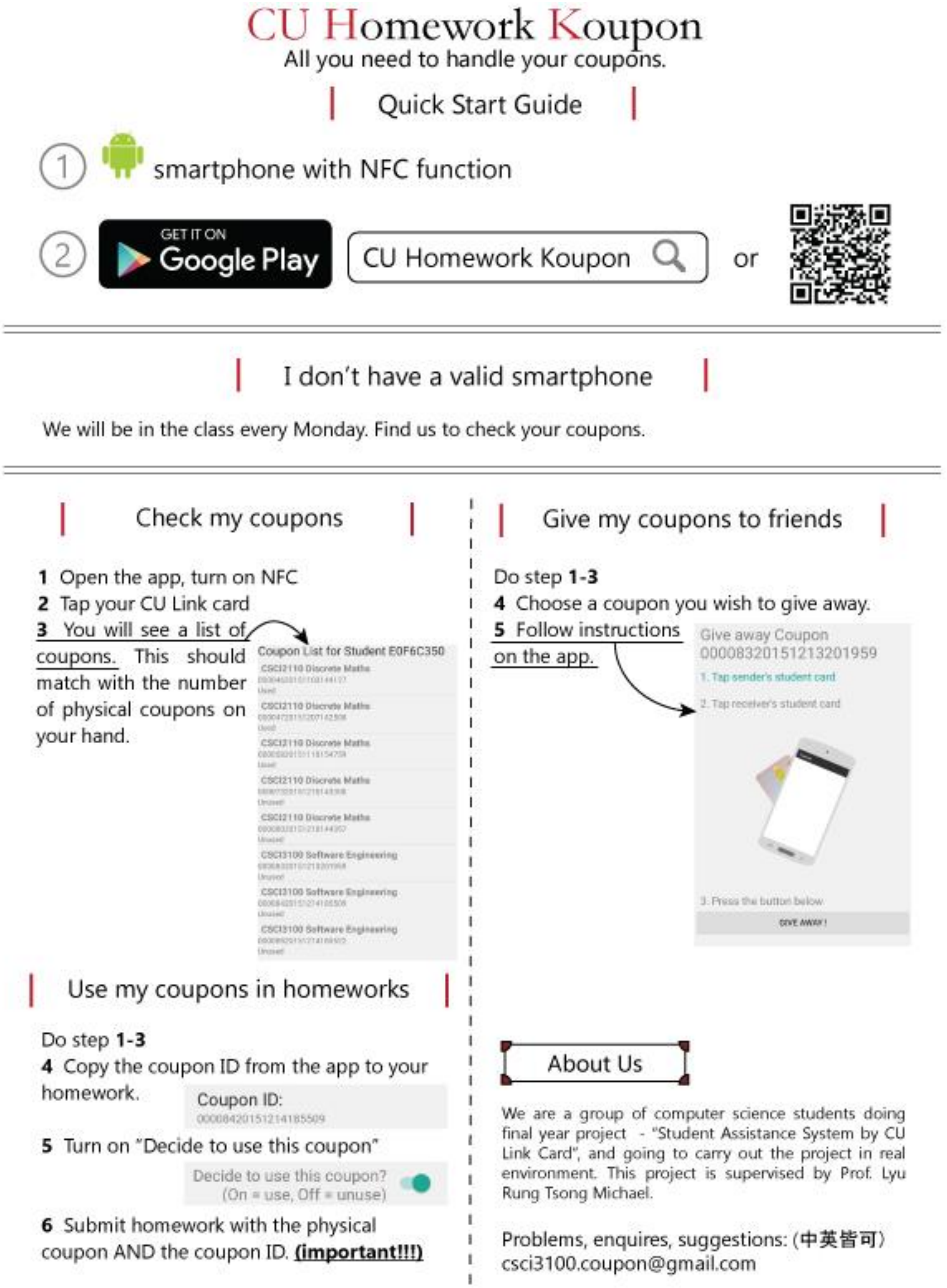
2. Tap receiver's student card

3. Press the button below

About Us

We are a group of computer science students doing final year project - "Student Assistance System by CU Link Card", and going to carry out the project in real environment. This project is supervised by Prof. Lyu Rung Tsong Michael.

Problems, bugs reporting, enquires, complains, suggestions 英文中文也可:
plho3@cse.cuhk.edu.hk (Jacky),
tkng3@cse.cuhk.edu.hk (Jeff)



CU Homework Koupon

All you need to handle your coupons.

Quick Start Guide

① smartphone with NFC function

② GET IT ON Google Play CU Homework Koupon or

I don't have a valid smartphone

We will be in the class every Monday. Find us to check your coupons.

Check my coupons

1 Open the app, turn on NFC
2 Tap your CU Link card
3 You will see a list of coupons. This should match with the number of physical coupons on your hand.

Coupon List for Student E0F6C350

CSCI2110 Discrete Maths
[illegible]
Used

CSCI2110 Discrete Maths
[illegible]
Used

CSCI2110 Discrete Maths
[illegible]
Used

CSCI2110 Discrete Maths
[illegible]
Unused

CSCI2110 Discrete Maths
[illegible]
Unused

CSCI3100 Software Engineering
[illegible]
Unused

CSCI3100 Software Engineering
[illegible]
Unused

CSCI3100 Software Engineering
[illegible]
Unused

Use my coupons in homeworks

Do step **1-3**
4 Copy the coupon ID from the app to your homework.

Coupon ID:
00008420151214185509

5 Turn on "Decide to use this coupon"

Decide to use this coupon?
(On = use, Off = unuse) ●

6 Submit homework with the physical coupon AND the coupon ID. **(important!!!)**

Give my coupons to friends

Do step **1-3**
4 Choose a coupon you wish to give away.
5 Follow instructions on the app.

Give away Coupon
00008320151213201959

1. Tap sender's student card

2. Tap receiver's student card

3. Press the button below

GIVE AWAY !

About Us

We are a group of computer science students doing final year project - "Student Assistance System by CU Link Card", and going to carry out the project in real environment. This project is supervised by Prof. Lyu Rung Tsong Michael.

Problems, enquires, suggestions: (中英皆可)
csci3100.coupon@gmail.com

CU Homework Koupon

All you need to handle your coupons.

Quick Start Guide

GET IT ON Google Play | CU Homework Koupon | or

News

Android smartphones without NFC can use our app.

I don't have a valid smartphone

We will be in the class every Monday. Find us to check your coupons.

Check my coupons

My smartphone have NFC

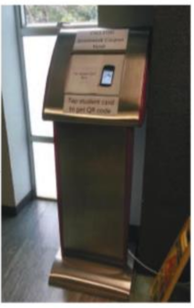
1 Open the app, turn on NFC
2 Tap your CU Link card
3 You will see a list of coupons you have.

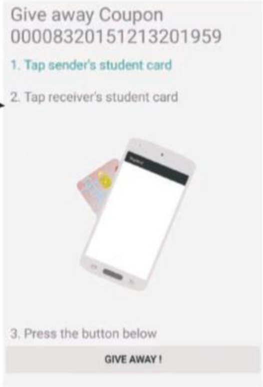
My smartphone do not have NFC

1 Go to SHB 1/F lift lobby, near SHB101
2 Tap your CU Link card to kiosk
3 Open the app, scan QR code
4 You will see a list of coupons you have.

Give my coupons to friends

Do step **1-3**
4 Choose a coupon you wish to give away.
5 Follow instructions on the app.





Give away Coupon
00008320151213201959

1. Tap sender's student card

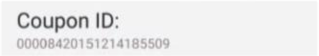
2. Tap receiver's student card

3. Press the button below

Use my coupons in homeworks

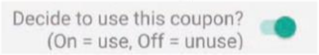
Do the above steps. Then:

1 Copy the coupon ID from the app to your homework.



Coupon ID:
00008420151214185509

2 Turn on "Decide to use this coupon"



Decide to use this coupon?
(On = use, Off = unuse)

3 Submit homework with the physical coupon AND the coupon ID. **(important!!!)**

About Us

We are a group of computer science students doing final year project - "Student Assistance System by CU Link Card", and going to carry out the project in real environment. This project is supervised by Prof. Lyu Rung Tsong Michael.

Problems, enquires, suggestions:
csci3100.coupon@gmail.com