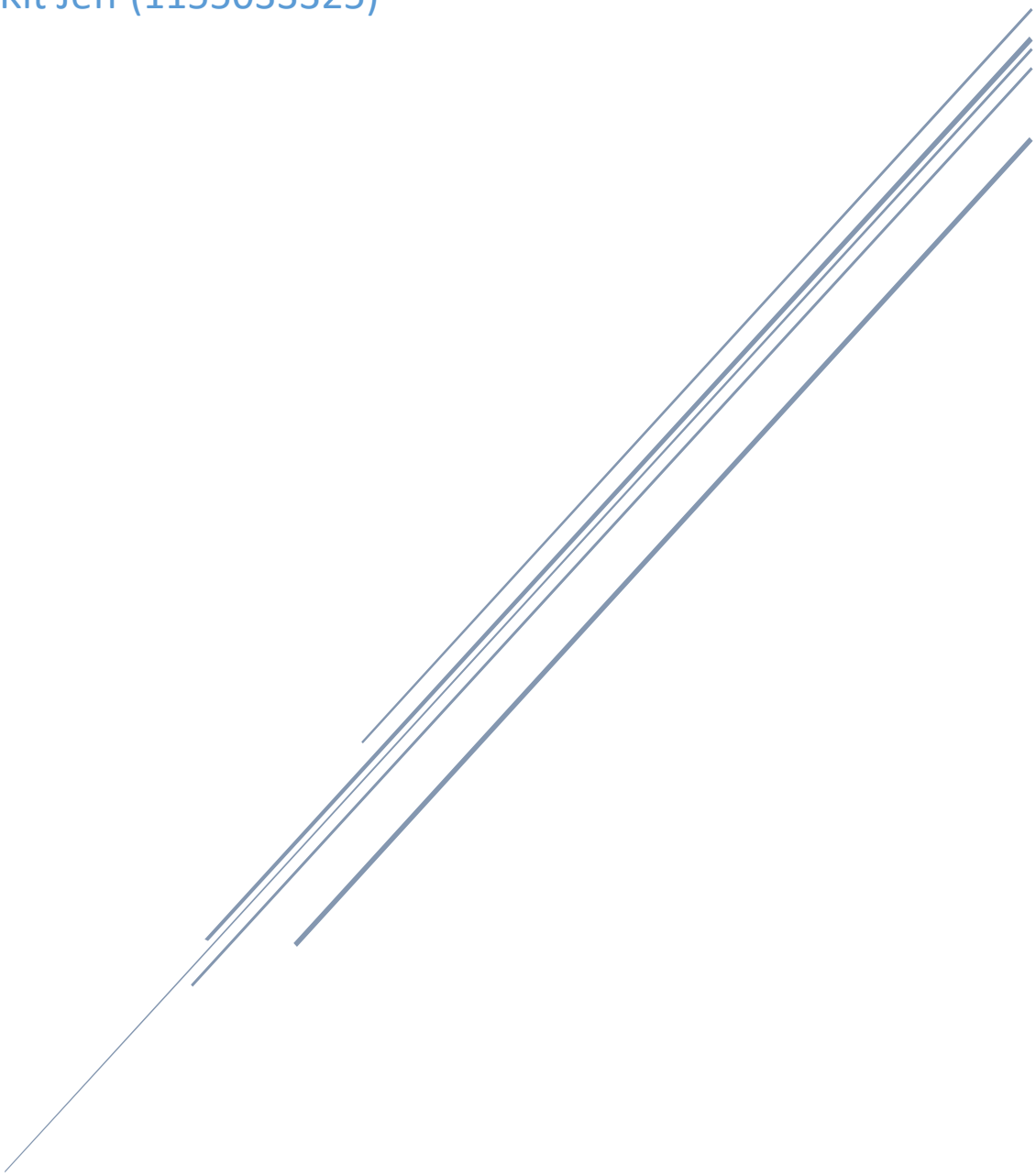
STUDENT ASSISTANCE SYSTEM BY CU LINK CARD

LYU1501 Final Year Project Report

Ho Pak Lam (1155034644)

Ng Tsz Kit Jeff (1155033325)



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 in this semester. “Homework coupon” is a type of coupon that teacher gives to student if students have answered questions well during class in order to improve interactivity. Student could also be exempted in doing some homework questions by using coupons, which is now used in course CSCI3100. Android applications are developed especially for teachers, tutors, and students. They could use their CU Link card, which is a NFC card, to tap on the android device as a token to recognize a user. 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 android device. Moreover, teachers may trace each coupon transaction and view coupon detail information at any time.

Apart from the android application, there is a server which contains a database to store those information. Also, database design, schema, and implementation are discussed in this project.

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 activities. 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.

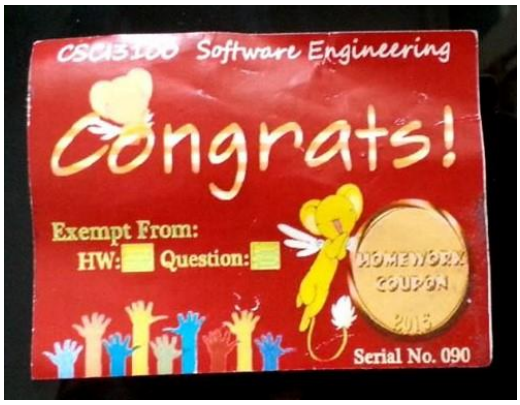


Figure 1 A sample of "homework coupons" used in course CSCI3100.

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.

2. Technical Support and Preliminary study

2.1 Development Environment

As a smartphone application with NFC function is going to be build, we use android studio with Java. We only develop the application on Android environment but not in iOS, because until iPhone 6, iPhone does not contain NFC module. For iPhone 6 and 6 Plus, Apple has locked its NFC chip function for apps other than Apple Pay. [1] Therefore, Android is our focus.

During 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.

2.2 Software tool: Android Studio

Figure 2 Icon of 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" [2], according to Android Studio. Suppose we want to change the compiling version, editing the plain text in the Gradle build files works.

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.

2.3 Software tool: Database by 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 3 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” [3] 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) [4]. 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 use 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. [5]

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” [6]. 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 two parts: server and client (application).

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

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.

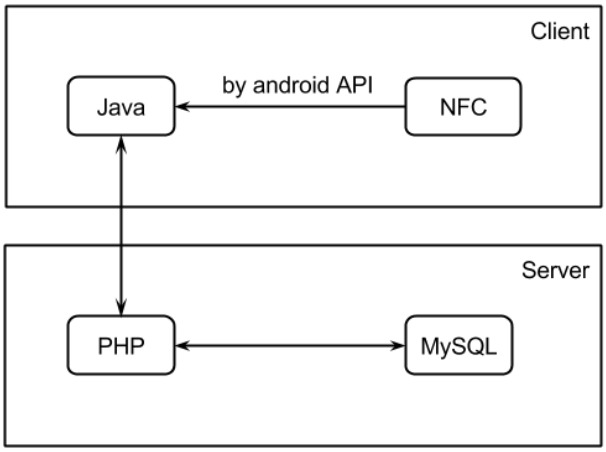


Figure 4 Simple graph showing the overall architecture of the system

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”.

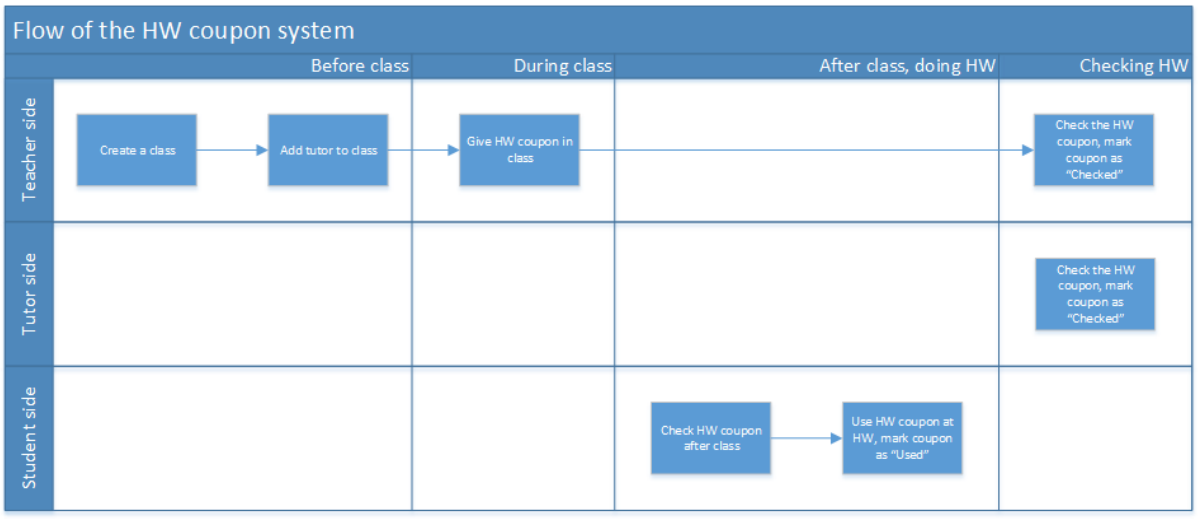


Figure 5 The flow of homework coupon system

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, 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 coupon 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.

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 6 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.

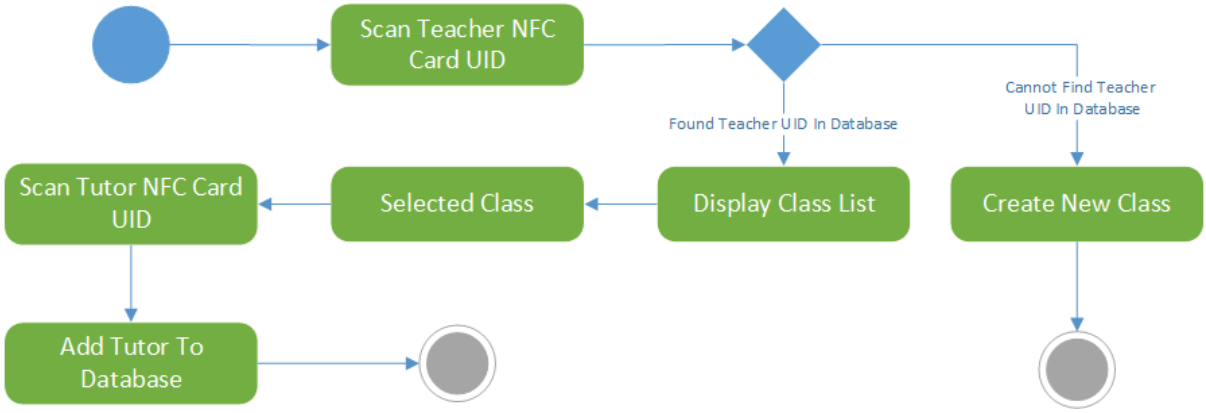


Figure 7 Activity diagram showing the flow of adding a tutor

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.

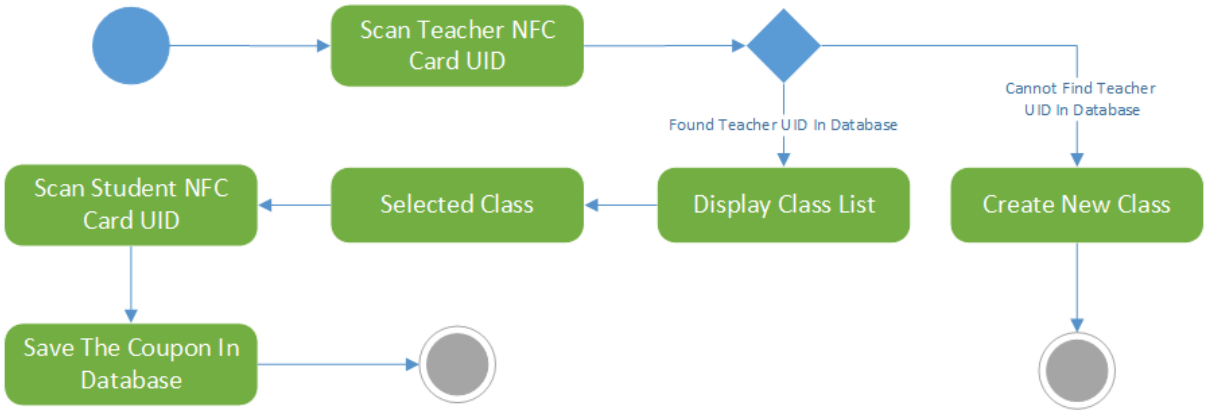


Figure 8 Activity diagram showing the flow of giving out coupons

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.

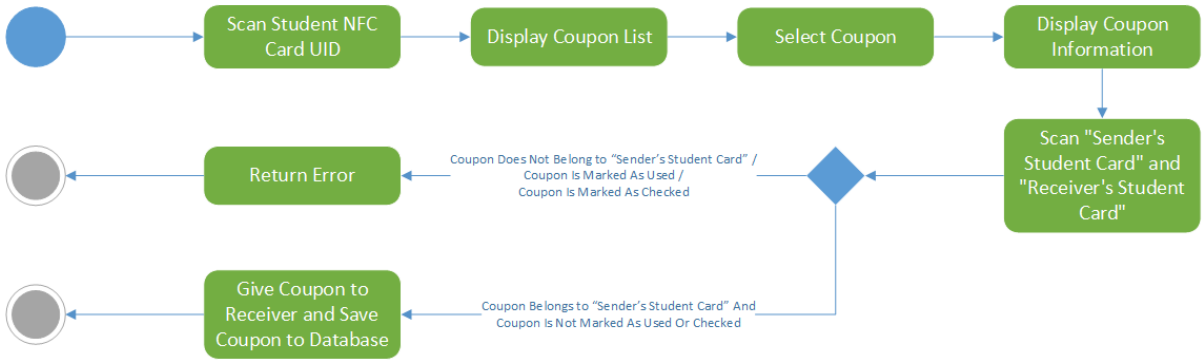


Figure 9 Activity diagram showing the flow of transferring coupon between users

View and Check Coupons

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.

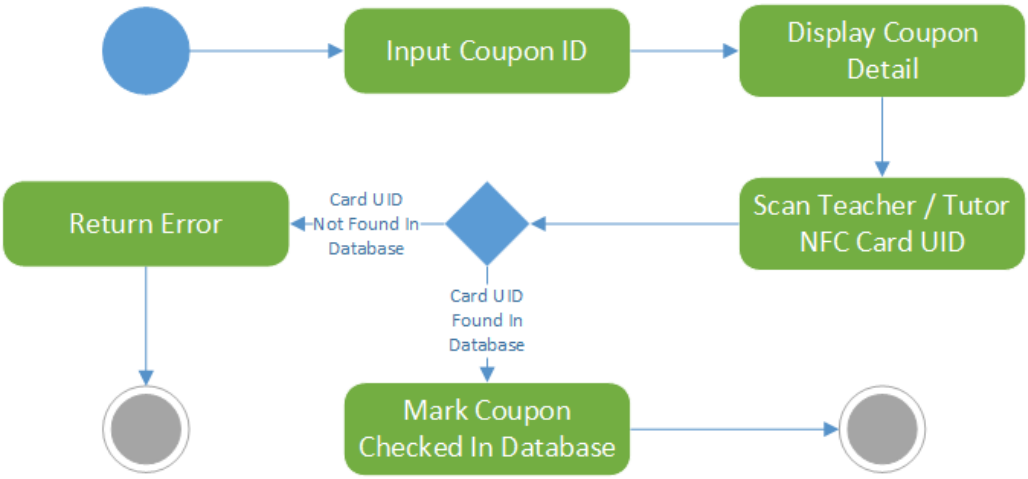


Figure 10 Activity diagram showing the flow viewing and checking coupons.

4. Design and Implementation

4.1 Server Side

4.1.1 Database: ER Diagram

The graph below shows the structure of our database.

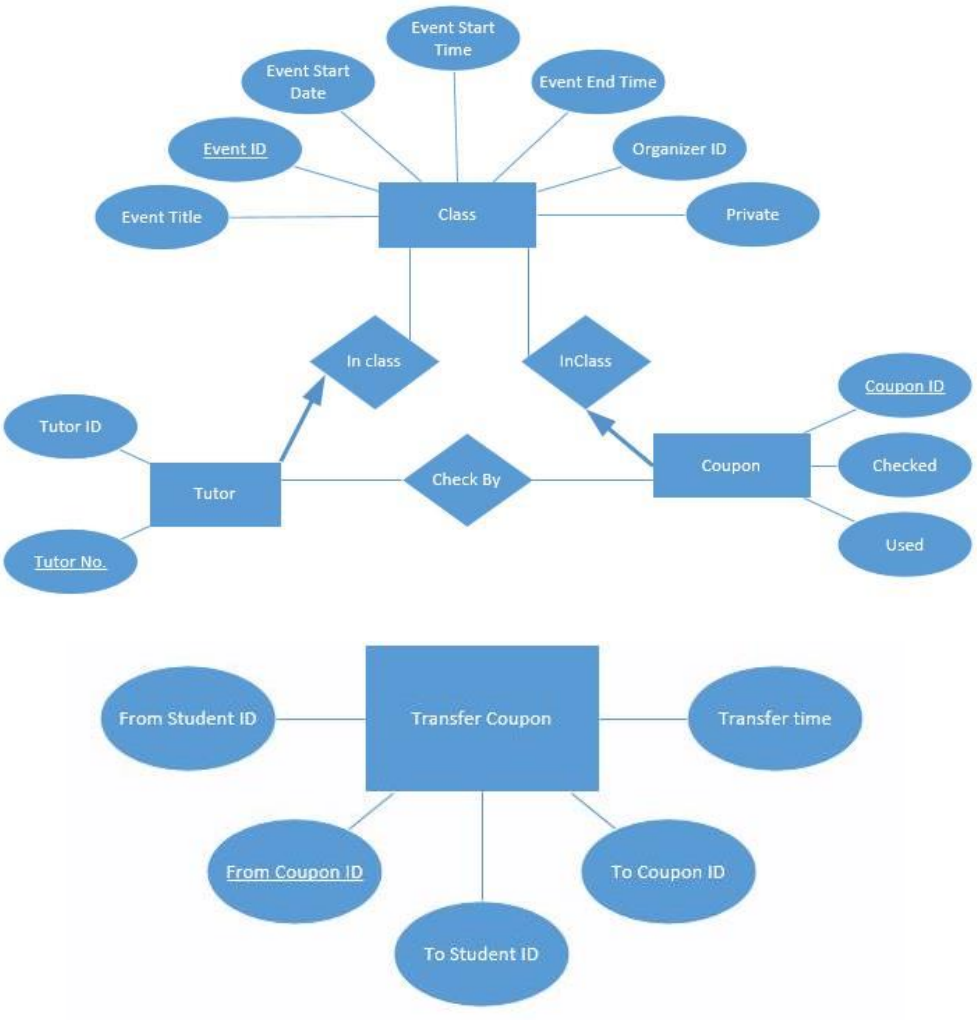


Figure 11 ER diagram showing the structure of our database.

The database is used to store data including the class information, coupon information, etc., as mentioned in the previous section. The server offer 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(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)

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

Class Table

Each tuple represents a uniquely identified class/ event. `CouponID` is the primary key. When a user create 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
studentID	the card UID of the owner of this coupon	String
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
fromStudentID	The card UID of the sender of the coupon	String
fromCouponID	The original coupon ID, unique	String
toStudentID	The card UID of the receiver of the coupon	String
toCouponID	The newly generated coupon ID	String
tranTime	The timestamp of the occurrence time of this transaction.	Date Time

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

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 are 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, db_connect`
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`
Edit record in database	`mark_used_coupon, mark_checked_coupon`

Configuration module

This module include 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
        $con = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or
die(mysql_error());

        // Selecting database
        $db = mysql_select_db(DB_DATABASE) or die(mysql_error()) or
die(mysql_error());

        // returning connection cursor
        return $con;
}
```

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

// array for JSON response
$response = array();

// include db connect class
require_once __DIR__ . '/db_connect.php';

// connecting to db
$db = new DB_CONNECT();

if (isset($_POST["studentID"])){
      $studentID = $_POST['studentID'];

      // get all coupons from coupons table
      $result = mysql_query("SELECT * FROM lyu1501_coupon NATURAL JOIN
lyu1501_class WHERE studentID = \"$studentID\"") or die(mysql_error());

      // check for empty result
      if (mysql_num_rows($result) > 0) {
            // looping through all results
            // coupon node
            $response["coupon"] = array();

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

                  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 {
            // no coupons found
            $response["success"] = 0;
            $response["message"] = "No coupon found";

            // echo no users JSON
            echo json_encode($response);
      }
```

```
}      else {
    $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 WHERE fromCouponID LIKE '$searchID%' OR toCouponID LIKE '$searchID%' ORDER BY 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(studentID, couponID, eventID) VALUES('$studentID', '$nextCoupon', '$eventID') INSERT INTO lyu1501_trancou (fromStudentID , fromCouponID , toStudentID , toCouponID , tranTime) VALUES ('$fromTeacherID', '$logString', '$studentID', '$nextCoupon', '$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 WHERE studentID = '$fromStudentID' AND couponID = '$fromCouponID' AND checked = '0' AND used = '0' UPDATE lyu1501_coupon SET studentID = '$toStudentID' , couponID = '$nextCoupon' WHERE CONCAT(couponID) = $fromCouponID" INSERT INTO lyu1501_trancou (fromStudentID , fromCouponID , toStudentID , toCouponID , tranTime) VALUES ('$fromStudentID', '$fromCouponID', '$toStudentID', '$nextCoupon', '$sqlNowString')

Edit Record in database

There are 2 modules. When the student decided to use the coupon, he/ she can click the button available in the android app. 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`)

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

A student indicate that he/ she want to use the coupon.
1. Search the coupon ID.
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`)
	`SELECT checked`
	`FROM lyu1501_coupon`
	`WHERE couponID = \"$couponid\"`
	
	(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'`

Database Normalization

The main purpose of doing normalization is to reduce data redundancy, and inconsistence dependency. [7] 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 ID of the sender and receiver, 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. [8] 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. This database is not in 3NF. For example, in the table “trancou”, the attribute toStudentID depends on toCouponID, which a non-prime attribute depends on another non-prime attribute. Similarly the pair fromStudentID and fromCouponID are dependent. However, as “fromCouponID” is declared as primary key, it is not the concern of 3NF. We focus of the pair “toStudentID” and “toCouponID”.

The following changes could be made to the database. Strikethrough lines means the attribute should be removed so as to fit 3NF. An additional table is created, called “belong”, which is a table indicates the dependency of couponID with studentID.

```
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)

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

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.

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(studentID:varchar, 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 produce same set of useful result but the join attribute is duplicated. If we choose inner join, "eventID" will appear 2 times in every tuples, 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. [9] 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.

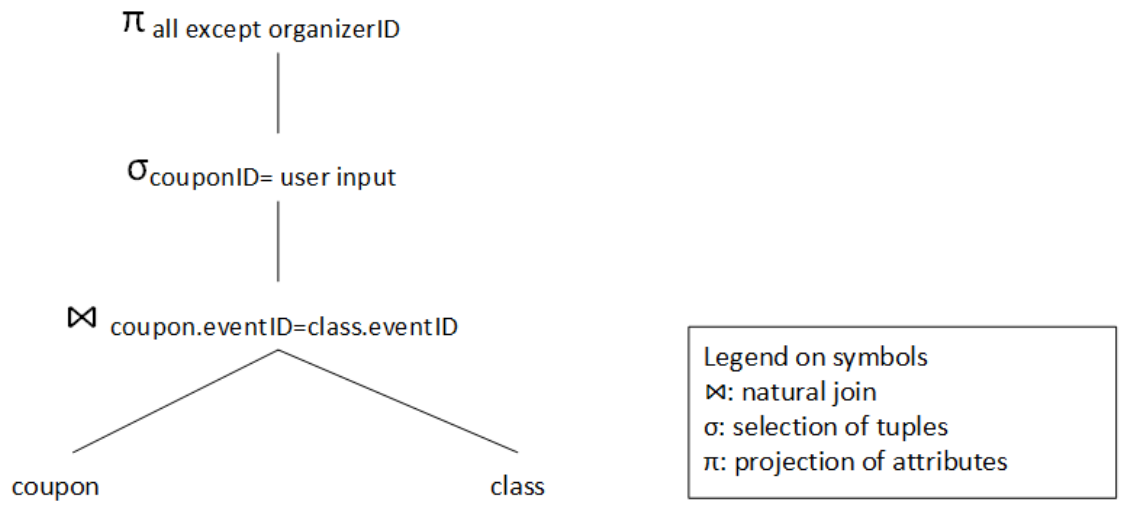


Figure 12 Logical query plan of "join, select, project"

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:

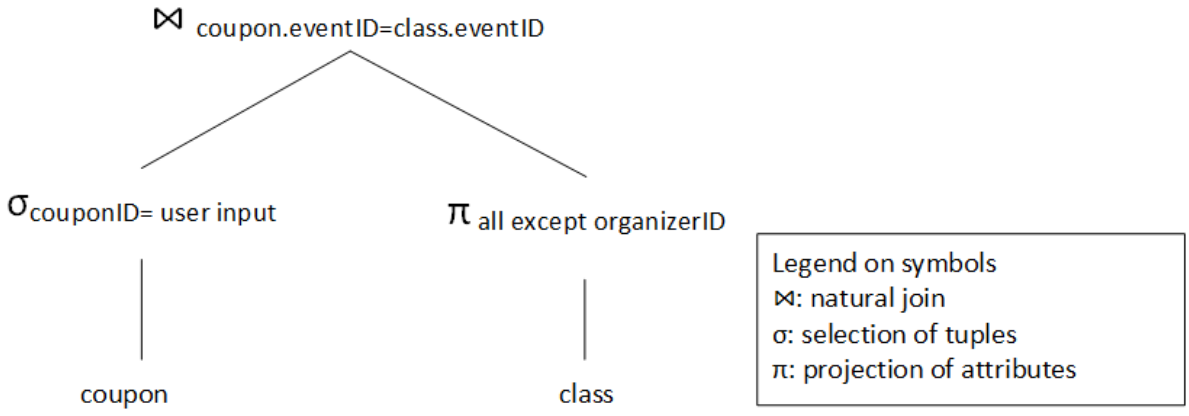


Figure 13 Logical query plan of doing "select, project, join"

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 are 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. Until now, 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. [10] 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 [11]. In next semester, we will use MySQLi to fix this problem.

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.

4.2 Client Side

4.2.1 Modules design: Concerning different users

In the “homework coupon” system, we could divide functions available in the android app into 7 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”.

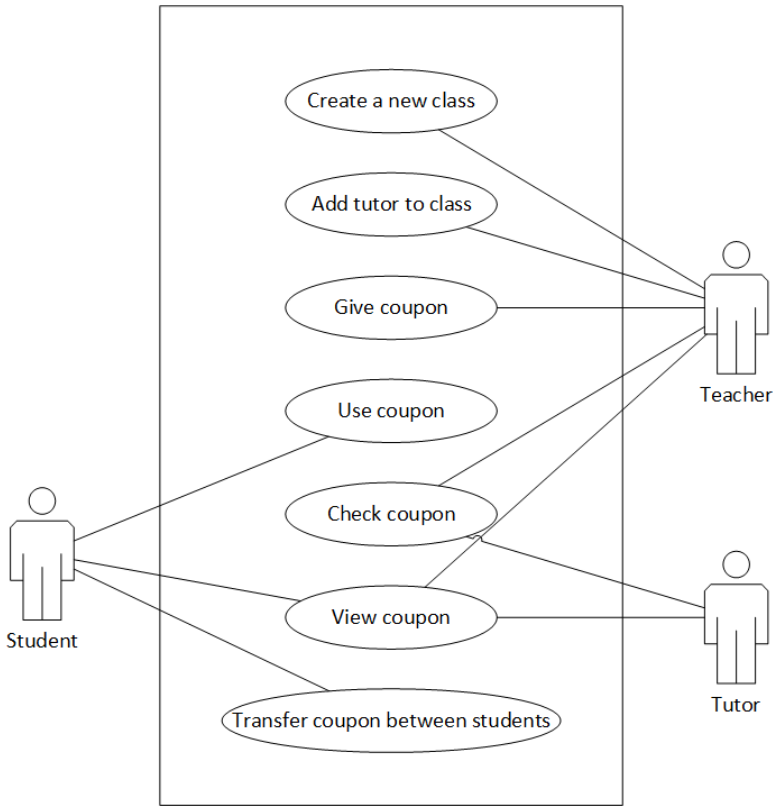


Figure 14 UML use-case diagram showing 7 major functions on basis of actors and actions

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”.

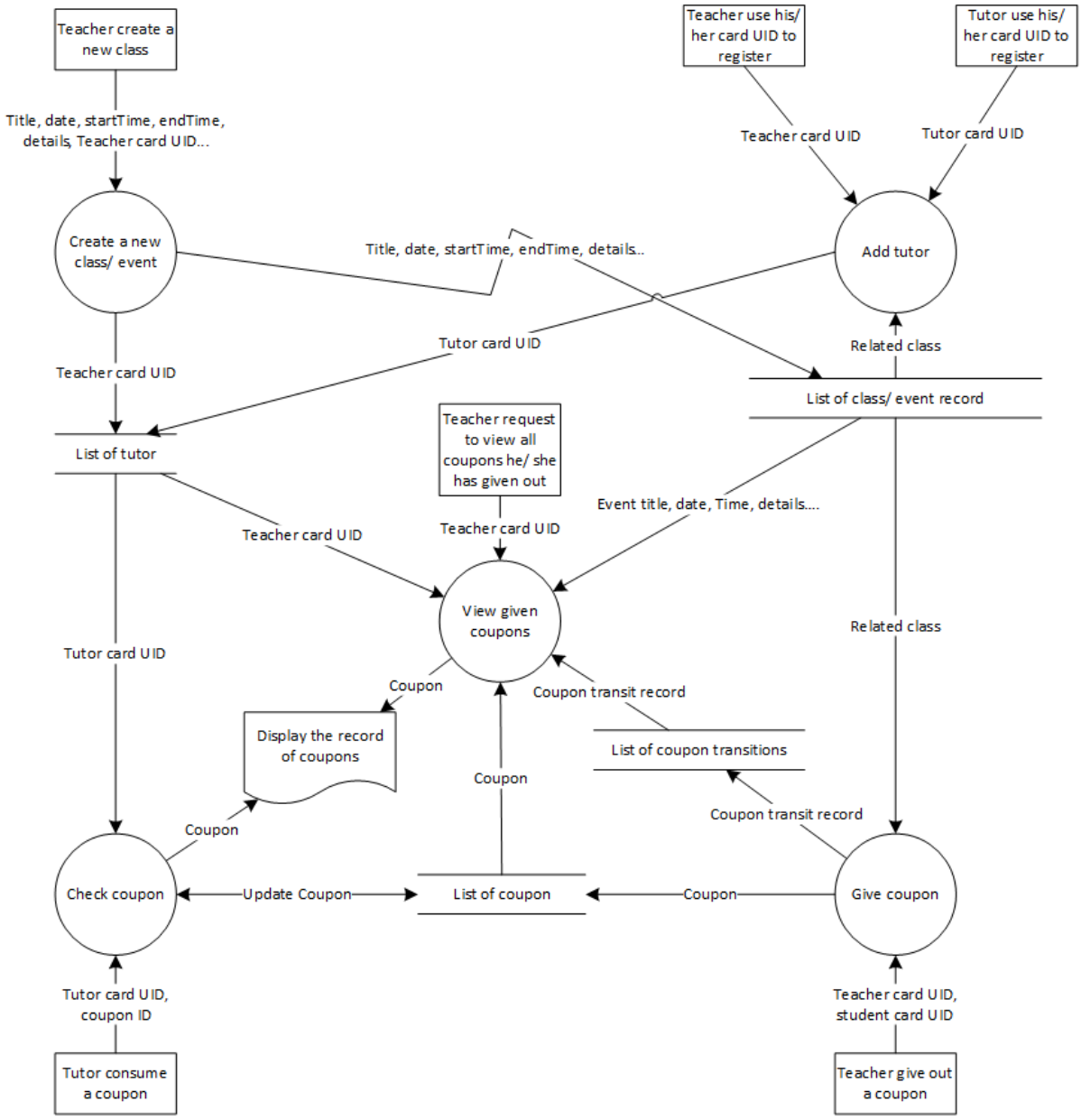


Figure 15 Data flow diagram showing the major functions related to teacher & tutor

The UML use-case diagram in the previous section shown that a student can use 3 major functions. One of the functions “View given coupons” is described in the table above. The remaining 2 functions are described below.

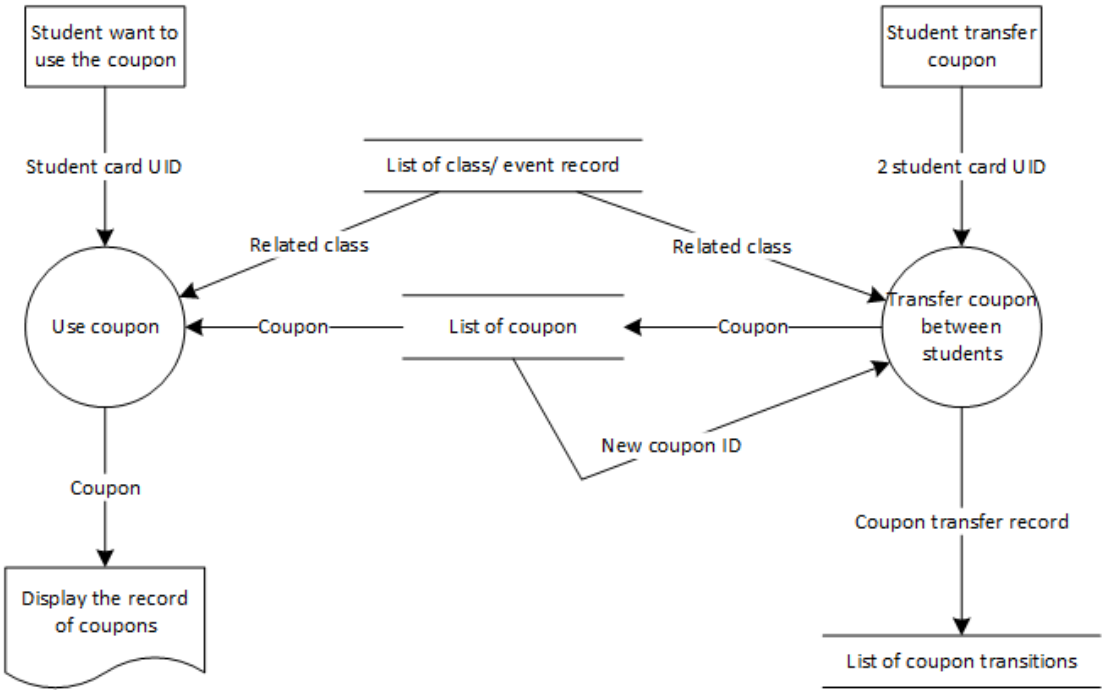


Figure 16 Data flow diagram showing the major functions related to student

4.2.3 User interface design

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. 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 histor. The record is shown on the screen.

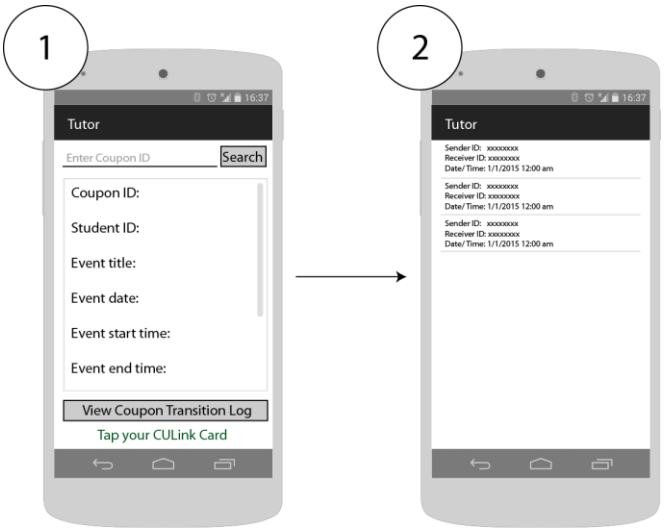


Figure 17 Graph showing the flow of screen for tutor

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.

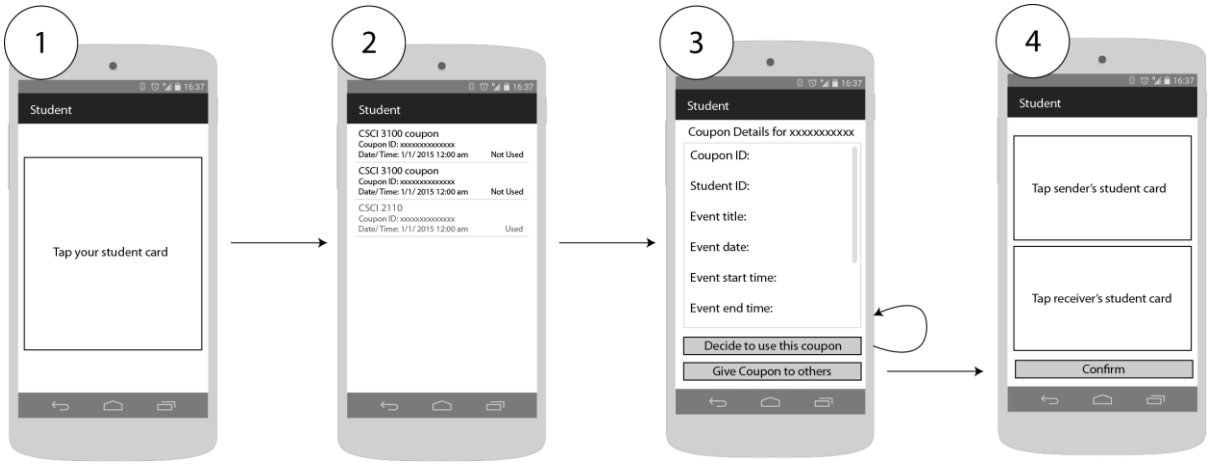


Figure 18 Graph showing the flow of screen for student

1. Home screen (tap student card)
2. List of coupon a student got
3. Details of a particular coupon
4. Give coupon to others

Teacher App

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

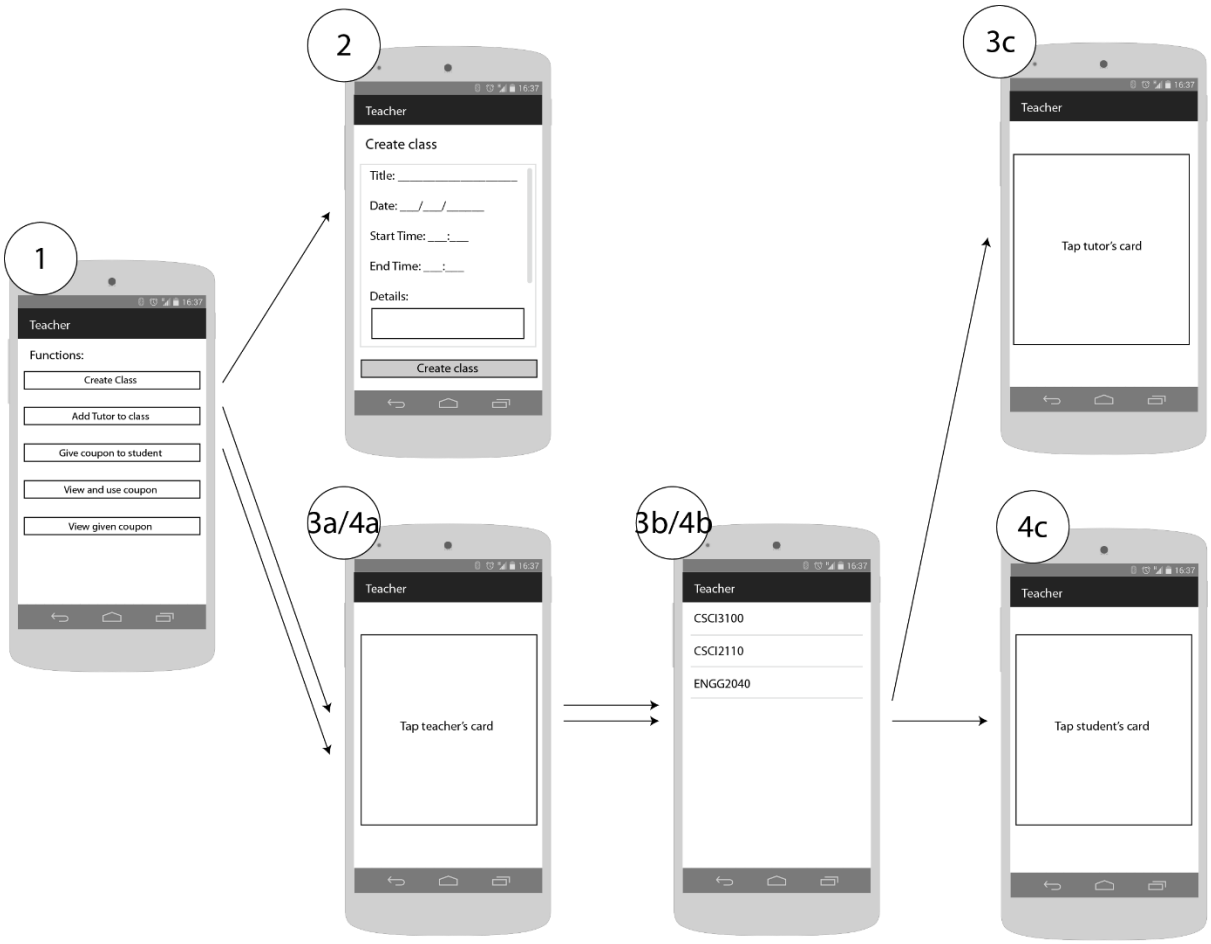


Figure 19 Graph showing the flow of screen for teacher.

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”.

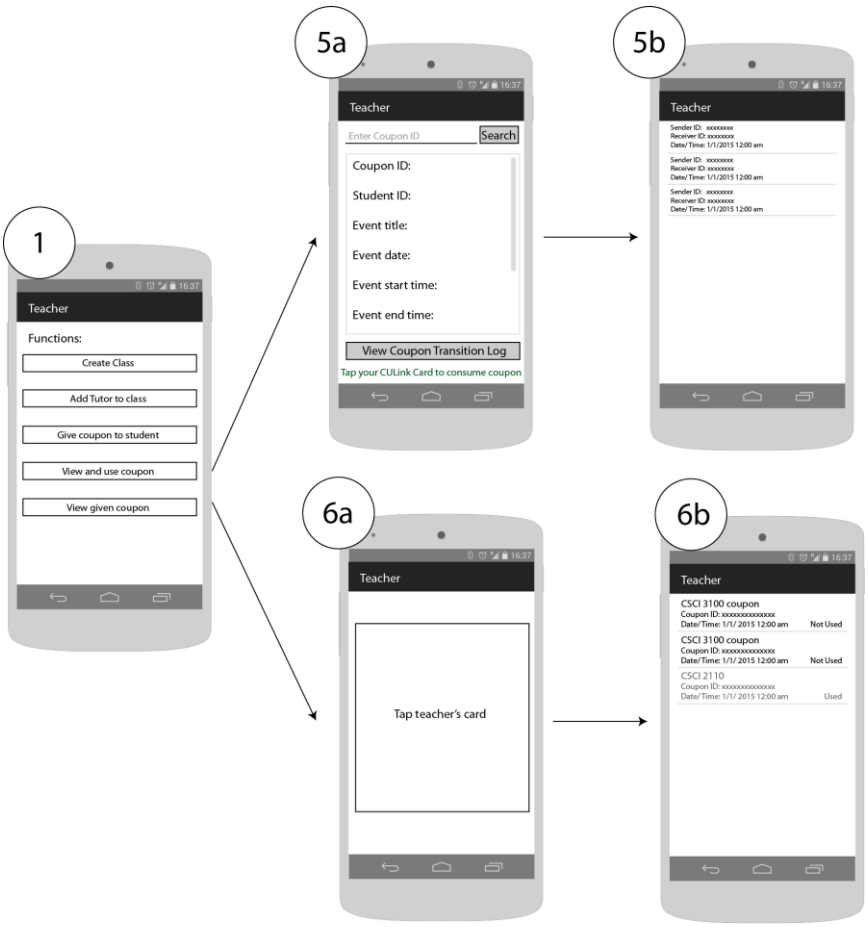


Figure 20 Graph showing the flow of screen for teacher.

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).

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.

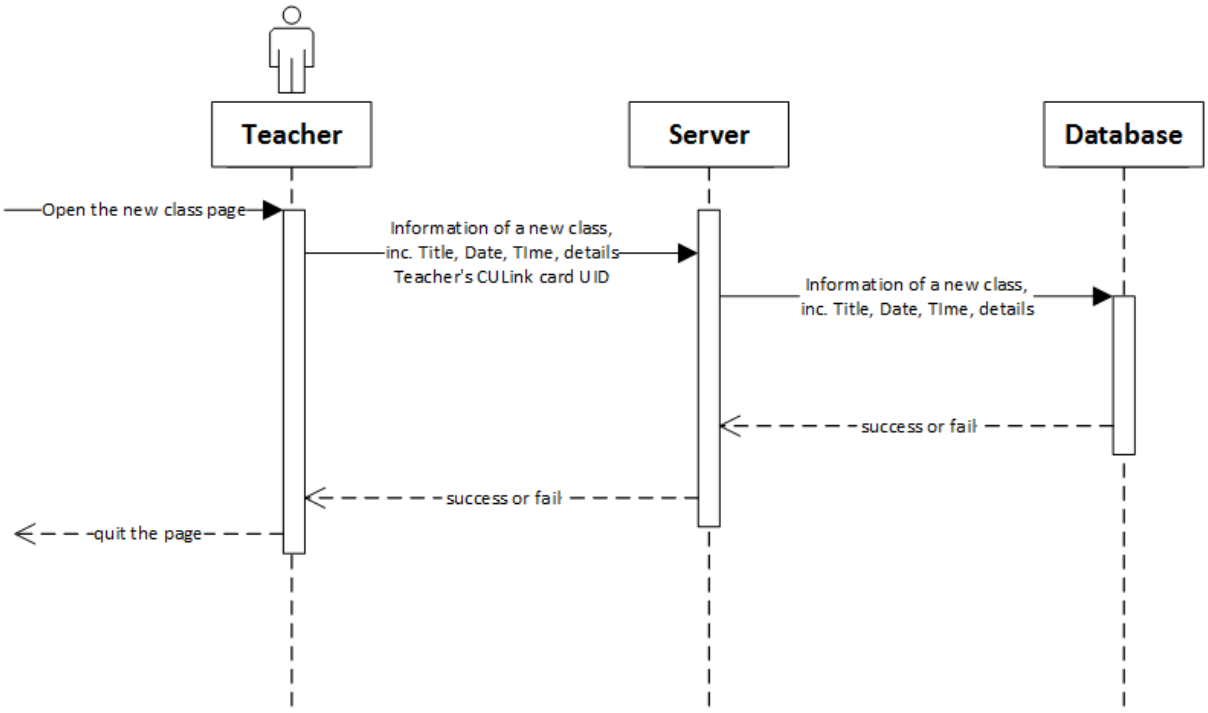


Figure 21 UML sequence diagram showing the data exchange of adding a new class/ event.

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.

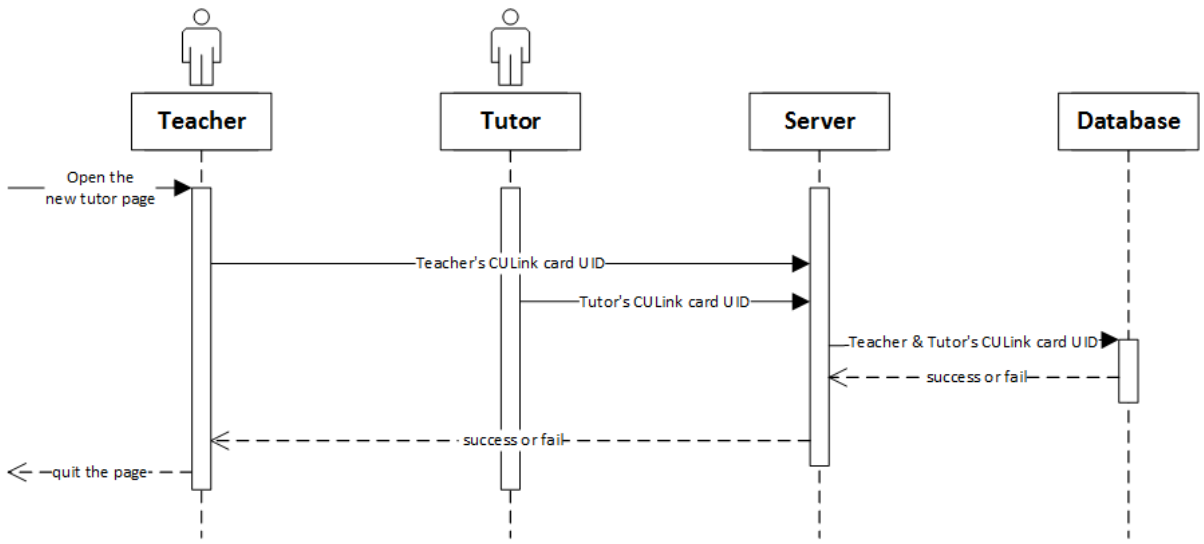


Figure 22 UML sequence diagram showing the data exchange of adding a tutor to a class.

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.

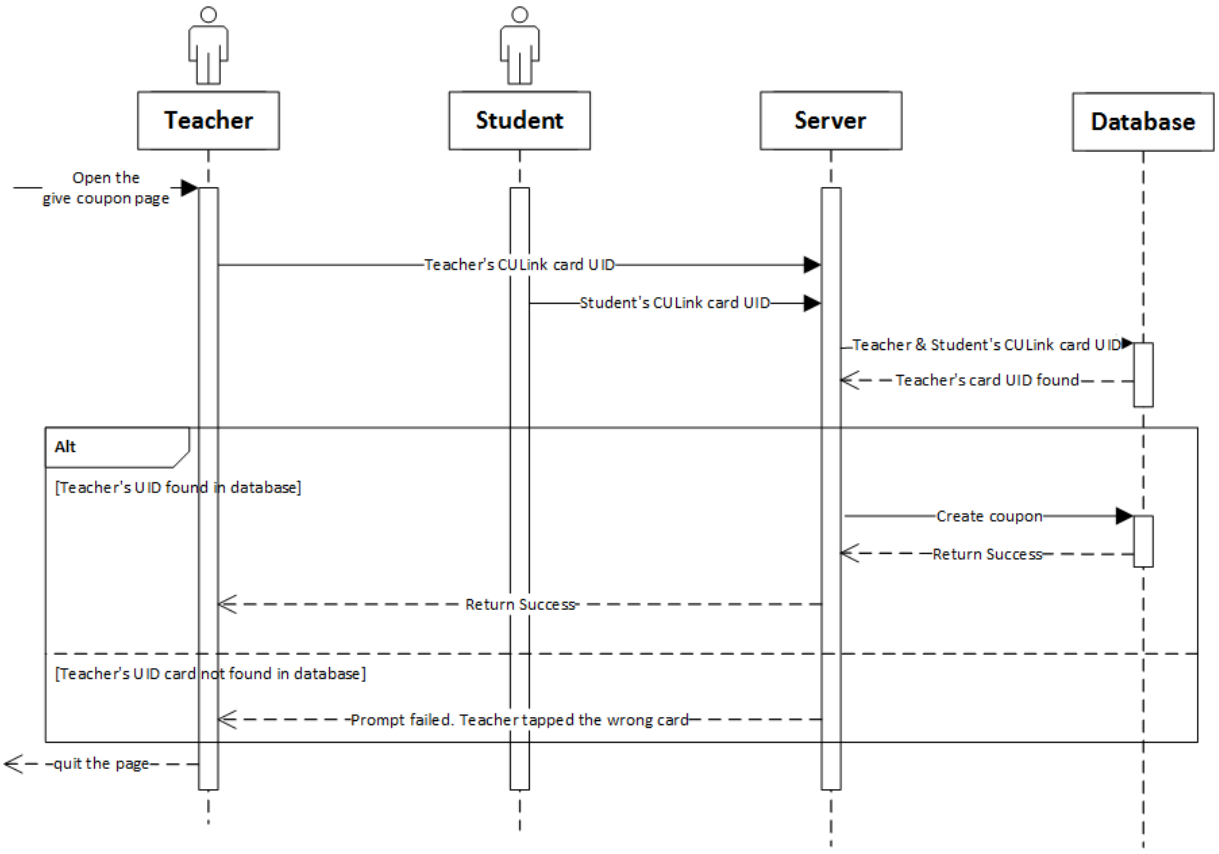


Figure 23 UML sequence diagram showing the data exchange of giving out a coupon.

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.

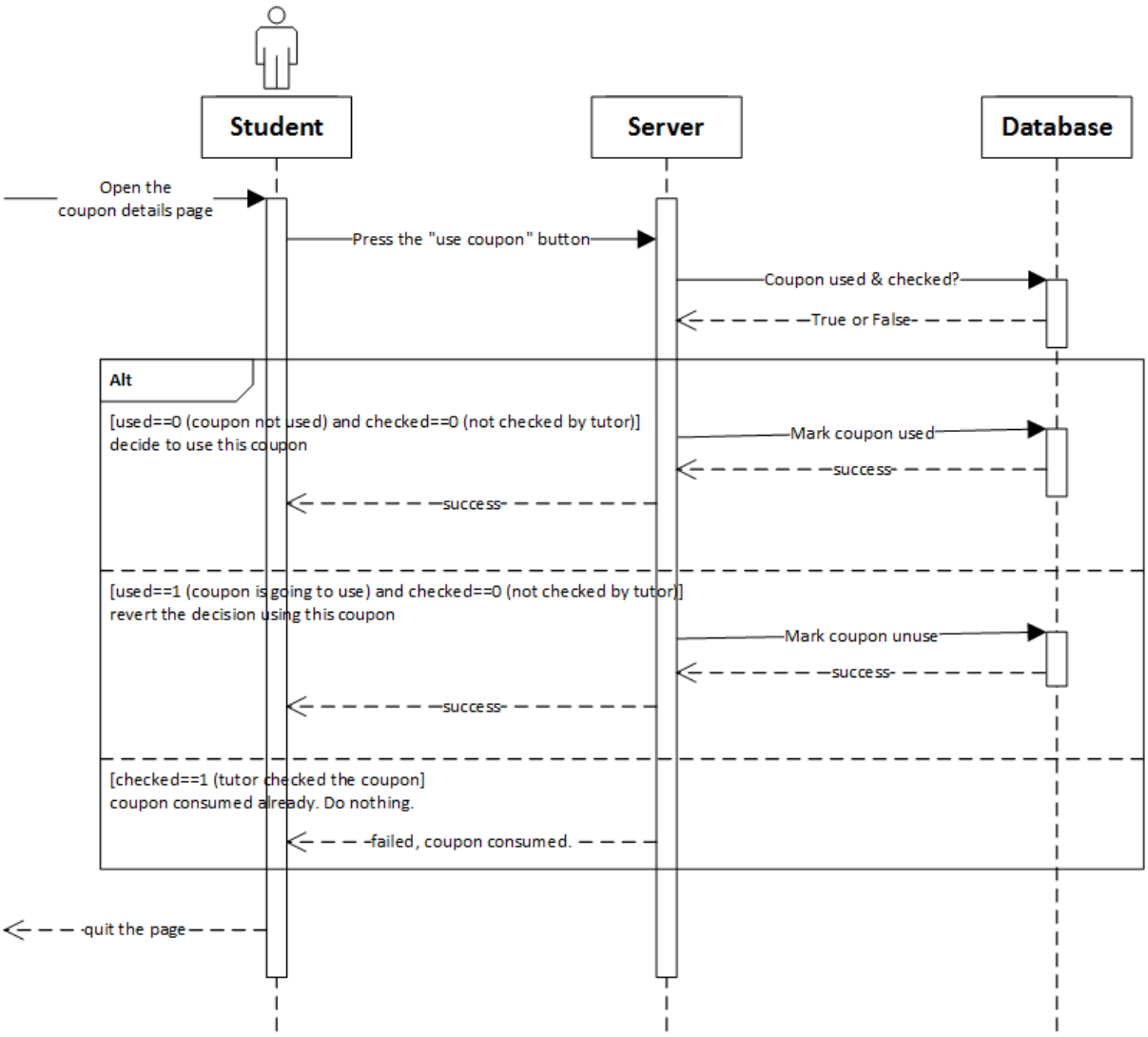


Figure 24 UML sequence diagram showing the data exchange of a student indicating whether he/ she is going to use a coupon.

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.

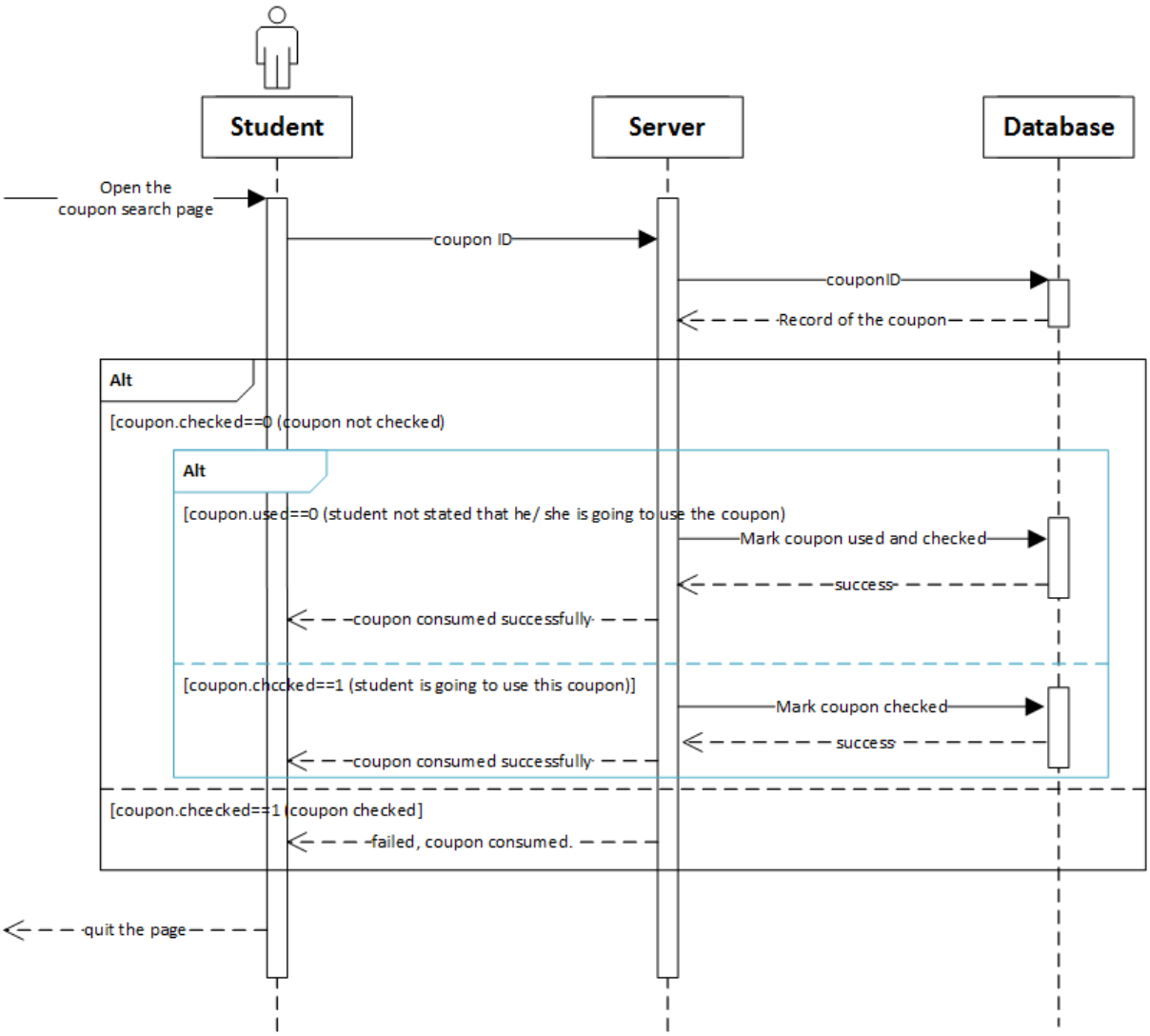


Figure 25 UML sequence diagram showing the data exchange of searching out a particular coupon to validate (mark the coupon as consumed).

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.

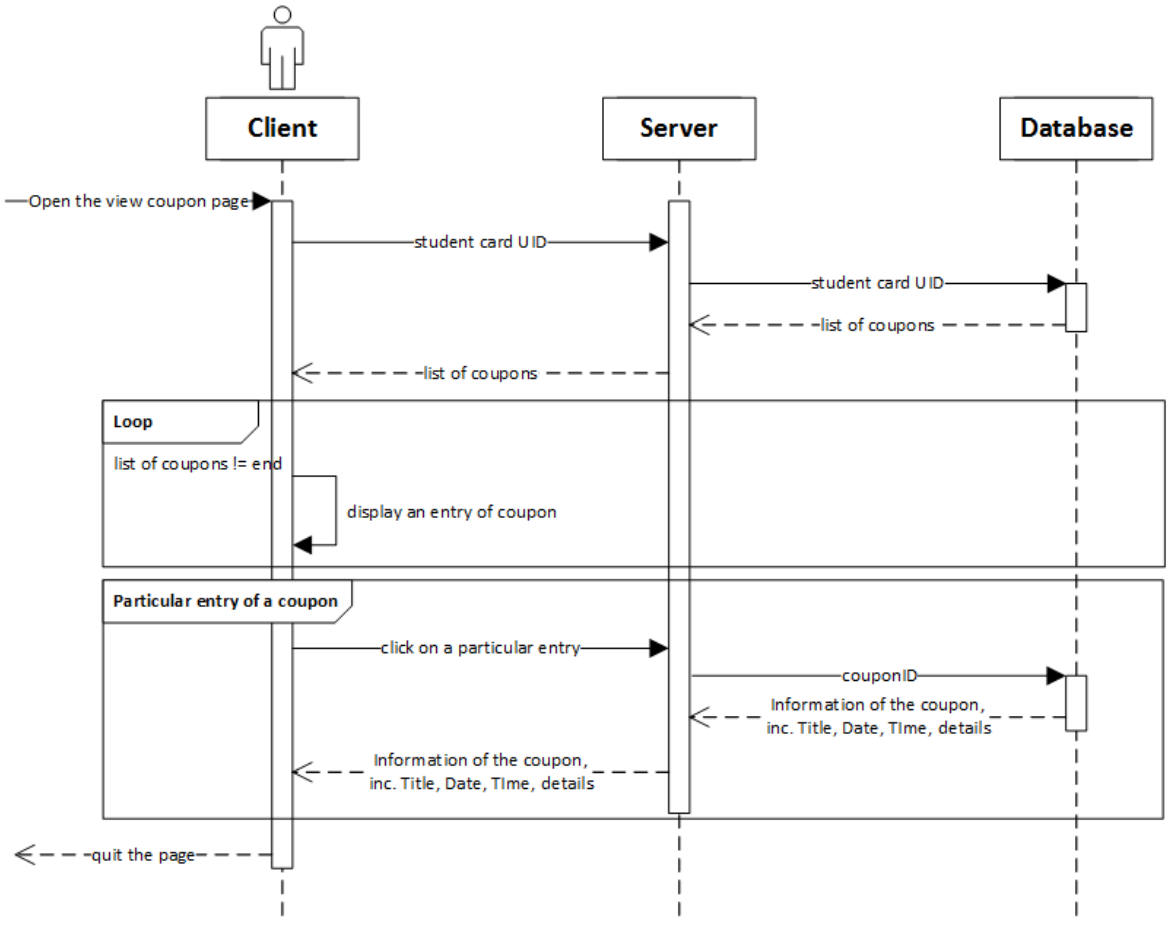


Figure 26 UML sequence diagram showing the data exchange of viewing the detail of coupons the user has.

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.

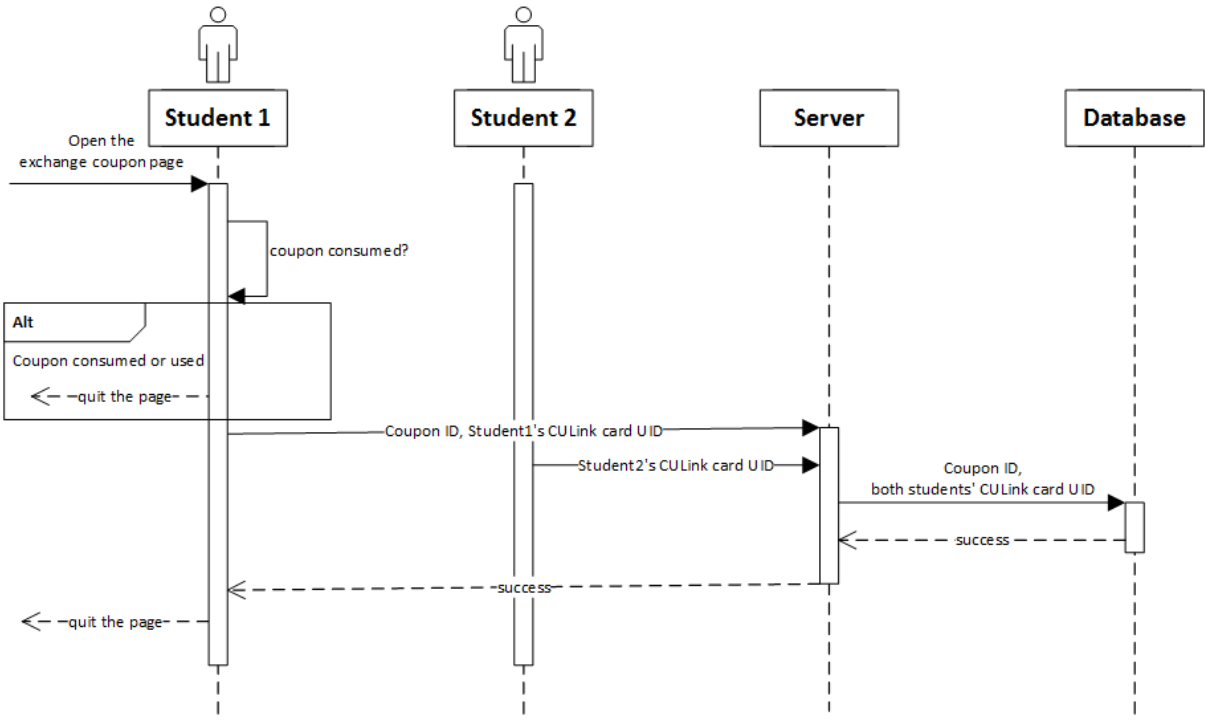


Figure 27 UML sequence diagram showing the data exchange of exchanging coupons between students.

4.2.5 User Interface Implementation

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.

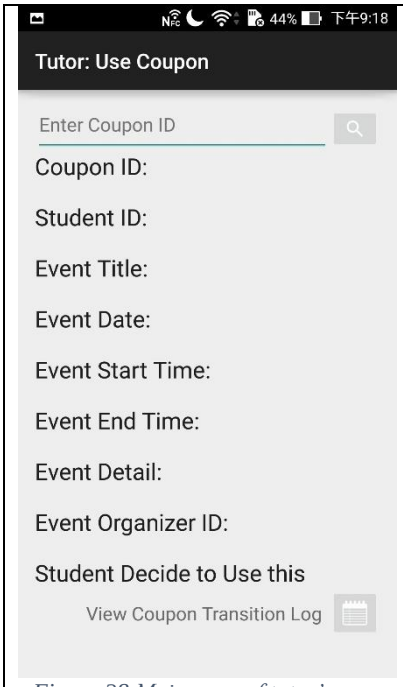


Figure 28 Main page of tutor's app

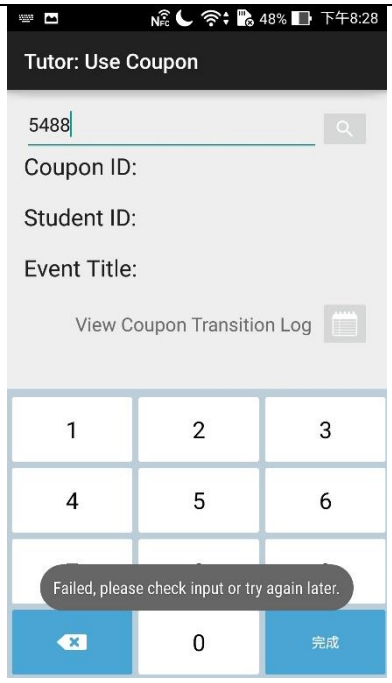


Figure 29 Simulating wrong input

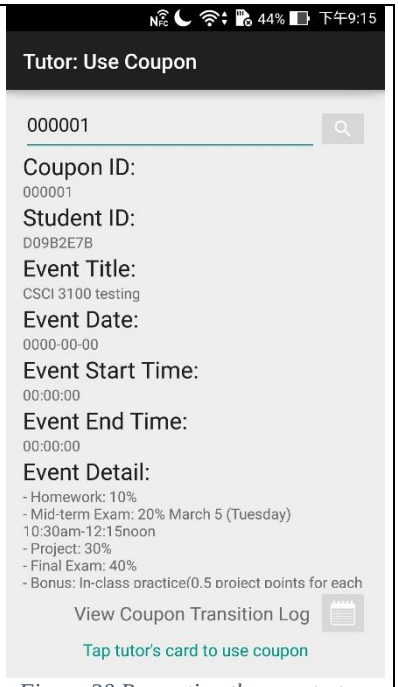


Figure 30 Prompting the user to tap CU Link card

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

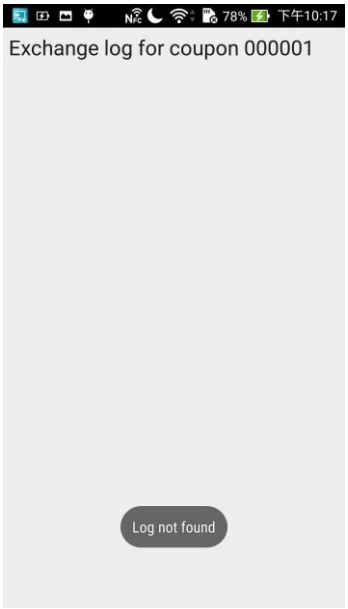


Figure 31 Sample screenshot showing “Log not found”

For other pages, both in student app and teacher app, on button pressed, the toast appear 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 30, a text in green is shown to prompt user to tap the card any time to perform a particular function.

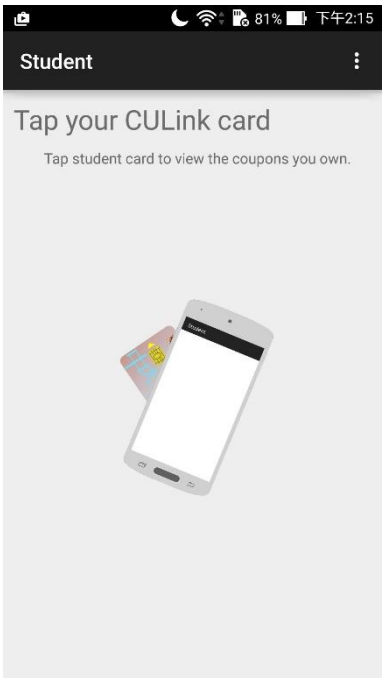


Figure 32 Screenshot prompting the user to tap CU Link card to view the coupons.

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 are also similar, with an icon and text description next to it. For simplicity, not all of the screenshots showing the design are shown.

4.2.6 Java Implementation

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" [12], 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 code 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.

```
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.

```
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.

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.

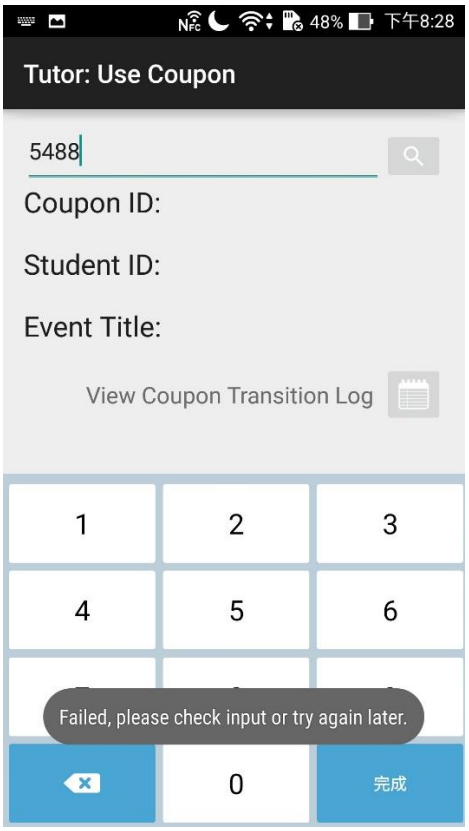


Figure 33 A sample screenshot showing the usage of toast

5. Limitation and difficulties

5.1 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.

5.2 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.

6. Future works

We have stated some problems and deficiencies in our project. Therefore, in the next semester, we aim to do the following tasks:

- Database design: Improves our database to comply with 3NF.
- Make a Kiosk: Make a kiosk for students who do not have an NFC android phone to view the coupons they got.
- Database security: Coping with SQL injection.
- Deployment: Examine the feasibility of deploying "Virtual homework coupon" system on course CSCI3100 in semester 2.
- Extension: Using the same set of functions/ modules created in first term, extend the system.

6.1 Database design

The current database system works and it is in second normal form (2NF). We aim to improve it into 3NF. 3NF ensures the database will not have update, insertion, and deletion anomalies.

Some changes are needed to the schema of the database which is also mentioned in part 4.1.2, the discussion of database normalization. Changes in SQL queries are also required as the schema is changed. There are no changes need to made for the android application. We aim

to finish this task before others, because migrating data would be easier if there are fewer records stored in database. It is estimated that this update could be finished quickly.

6.2 Make a kiosk

As we need to read a student's CU Link card and get its UID for a token of system, our system required user having android device with stable network connection and NFC function. However, not every student has an android device with NFC function. To serve those students, we aim to make a kiosk which allows students to view and mark their coupons.

We could make use of the kiosk that is currently not in use in the VIEW lab. By adding hardware including NFC reader, pointing device and software including Android emulator and simplified student client, student could view and mark their coupon in the kiosk with their own CU Link.

As the kiosk needs a power supply and a stable network connection, we consider placing the kiosk outside the VIEW Lab, which is at the first floor of Ho Sin-Hang Engineering Building.

6.3 Database security

Database is the most important part of our project. Protecting the database is required. PHP documentation gave out some suggestions on database security. One of the suggestions is to establish the connections over SSL. This method will be investigated in the coming semester. Another strategy for hacking the database is by SQL injection. User could destroy the database by such method. In this semester, we have done some measures on preventing SQL injection, previously mentioned in part 4.1.4. The major loophole is the data input field of the teacher app, since string is allowed. We will make use of MySQLi in semester 2 to prevent such problem.

6.4 Deployment

The “virtual homework coupon” system allows more functions than the existing one, including viewing the coupon transitions, knowing whether the coupon are really used successfully, etc. But there are more considerations if we really deploy the system in next semester. To get prepared of deployment.

Considerations:

1. Some students may not have NFC device.
2. Some of them may purposely intrude/ corrupt the system.
3. Measures to be done in case of system failure.
4. Method of deployment.

For point 1, it could be solved and it is mentioned in the section 6.2.

For point 2, we try to improve the security level and reduce possibilities of having SQL injection by using MySQLi in the coming semester, also mentioned in section 6.3.

For point 3, database backup is performed regularly to prevent data loss.

For point 4, we consider parallel operation, which the virtual coupon system runs, at the same time physical coupons are also given. Even there is any problem in the new system, students need not suffer a loss. It is considered that a brief introduction of the system should be given to students who takes the course.

6.5 Extension

“Virtual homework coupon system” is one of the applications of our system. In the next term, we consider extend our system to include taking attendance function. Student could take attendance with their CU Link and teacher could view attendance information with their Android phone. This may lower the cost of buying or renting a CU Link reader.

7. Conclusion

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

Although it is our first time to build an android application. It is great for us to have a chance to develop an application with NFC, also a server with PHP and MySQL database. In the beginning, we already experienced a barrier that the ITSC of the university could not give us any information nor access to the encrypted information. We then turns out the using UID as a token to identify users. Throughout the brainstorming process, we came up with different idea, and we found that “virtual homework coupon” is an interesting and useful example to visualize our project in this 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.

Despite the limitations and difficulties, will do continuous improvement to our system. We also aimed to provide more function in the coming semester and deploy the project in a real life situation if possible.

8. 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.

9. References

[1] 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/.

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

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

[4] "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/.

[5] 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.

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

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

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

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

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

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

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