

**Capstone Design (2)**

**Project Proposal**



**Course Name : Capstone Design (2)**

**Professor : SangOh Park**

**Team Name : GoNaMi**

**Team Member : 20160040 Hoyun Ko**

**20163228 Yuseon Nam**

**20164897 Seungyun Lee**

Content

[1. About Project 3](#_Toc11017528)

[1.1 Project Name 3](#_Toc11017529)

[1.2 Project Idea 3](#_Toc11017530)

[1.3 Background and Goal 3](#_Toc11017531)

[1.4 Expectation 4](#_Toc11017536)

[2. Project Overview 5](#_Toc11017537)

[2.1 Application 5](#_Toc11017538)

[2.2 Book Box 6](#_Toc11017545)

[3. Development Content 6](#_Toc11017546)

[3.1 Roles and Project Schedule 6](#_Toc11017547)

[3.2 Development Environment 9](#_Toc11017550)

[3.3 Project Reference 9](#_Toc11017551)

[4. Project Implementation 10](#_Toc11017552)

[4.1 Struct of Project 10](#_Toc11017554)

[4.2 Application 10](#_Toc11017555)

[4.3 Database 27](#_Toc11017556)

4.3 Book Box 31

[5. Project Result 34](#_Toc11017597)

[5.1 Project Completion 34](#_Toc11017598)

[5.2 Limitation and Improvement 34](#_Toc11017599)

[5.3 Review 35](#_Toc11017602)

# About Project

## Project Name

BookBoxBook

## Project Idea

You can see the book market briskly in the opening of a course. We tend to buy expensive books in secondhand books at a fraction of the cost of new ones. How do you deal? Usually, I use direct or courier transactions a lot. We think the key to dealing is ‘Safe Trade.’ Are our deals always secure and reliable? Don’t you worry about whether or not the seller’s items are normal when you buy them? We also worry about the correct goods will be delivered when we deposit the money first.

We were wondering if we could make a safe deal, not an unstable one. Create a barrel to make a safe deal. A transaction in which the seller puts the goods in a box and the buyer pays to receive the goods. This will be a convenient deal for both sellers and consumers.

## Background and Goal

## Background

## Frequent Book Transaction Between Student

It is not hard to find students selling and buying textbooks because the price of them is not affordable for students normally. As this trend spreads around the universities, a lot of intermediation services are emerged such as the application ‘Everytime’ and many other websites. However, the existing services offer only mediation between seller and buyer. It is their job to decide when and where the deal taking place. Depending on this problem, we enforce our competitiveness through providing the service deciding the spot and time.

## Increase in Fraud in Transactions

Because the transaction between seller and buyer is totally up to them, the fraud can be easily happened. And when they are fraudulent, there is no way to respond to the fraud. To prevent this damage, we are going to get into the deal and make sure users of our services are guaranteed safe transaction. And the ‘Book Box’ will provide this by offering the secure transaction.

## Goal

Through providing secure transaction, we aim to

**first**, activate the trade of books between students

**second**, reduce the fraud during transaction.

**third**, offer the efficient trading for the users.

## Expectation

Through providing secure and efficient way of trading used book between university students, we are expecting to activate used book trading market. This will help students to save their money and time.

# Project Overview

## Application

## Member Management

Before using application, user have to register account(sign up) and log in. User can log out or delete your account. Also, user can modify his/her personal information like phone number, school and password.

## Adding Book

When user wants to sell book, user can add book information manually or through barcode. Then, user have to put detailed information about his/her book; whether it has writings, damages and memos. User can put image of the book, must choose school (where to sell) and price.

## Shopping Book

User can see books in Search Page. User can search book by book name or school. When user selects one book, user can see details, image that seller put and seller’s rating. User can book mark the book.

## Buying Book

From book detail page, user can buy book. When user click buy button, he/she choose university where to trade book. Then, payment is processed.

## Book Mark

User can see book list that he/she has book marked. When user click book mark icon (blank book mark icon), book mark is removed. When user selects one book, user can see book information.

## Transaction List

User can see book list that he/she has registered or bought and see the state of the book. Here, seller can reserve book box date, see QR ,put his bank account. Buyer can see QR and confirm his/her purchase

## Book Box

Inform user what to do step by step when they come to use Book Box like kiosk. User can recognize their QR code and barcode using camera. And lock/unlock the box by servo motor.

# Development Content

## Roles and Project Schedule

## Role

**Hoyun Ko**

* Application
* Database and Server

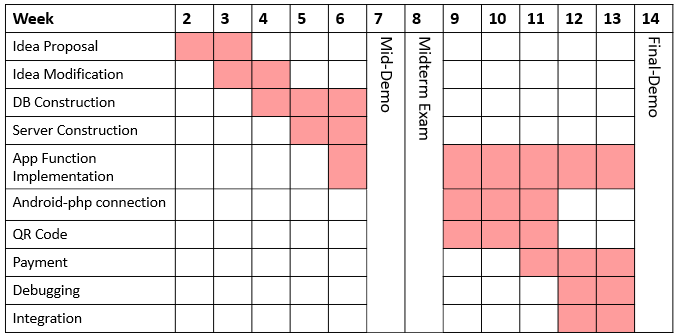
**Yuseon Nam**

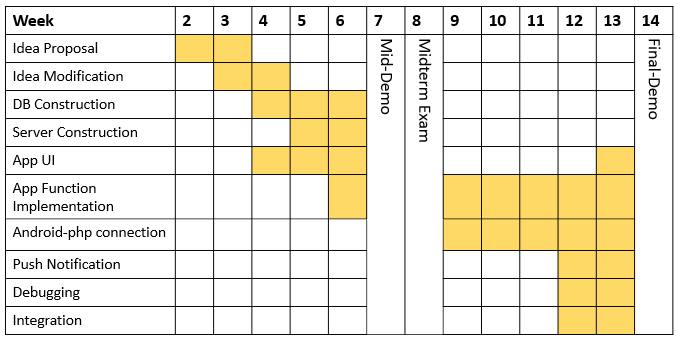
* Database and Server
* Application

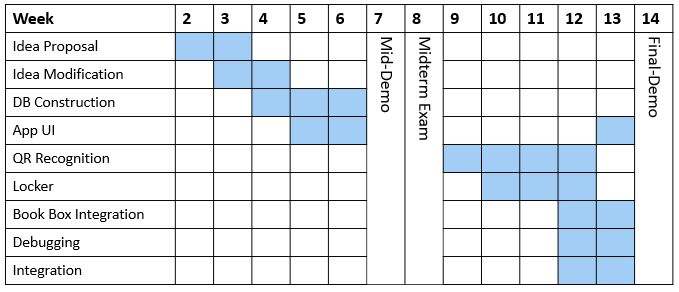
**Seungyun Lee**

* Raspberry Pi (Book Box)
* Application UI

## Project Schedule

**Hoyun Ko**

**Yuseon Nam**

**Seungyun Lee**

## Development Environment

|  |  |
| --- | --- |
| Project Component | Environment |
| Application | Java SDK 11.0.2  Android SDK 26.0  Android Studio 3.3.2 |
| Server | Ubuntu 18.04  Apache 2.4  PHP 7.0  MySQL 5.7 |
| Book Box | Raspbian 4.14  Python 3.6  OpenCV 3.4.3 |

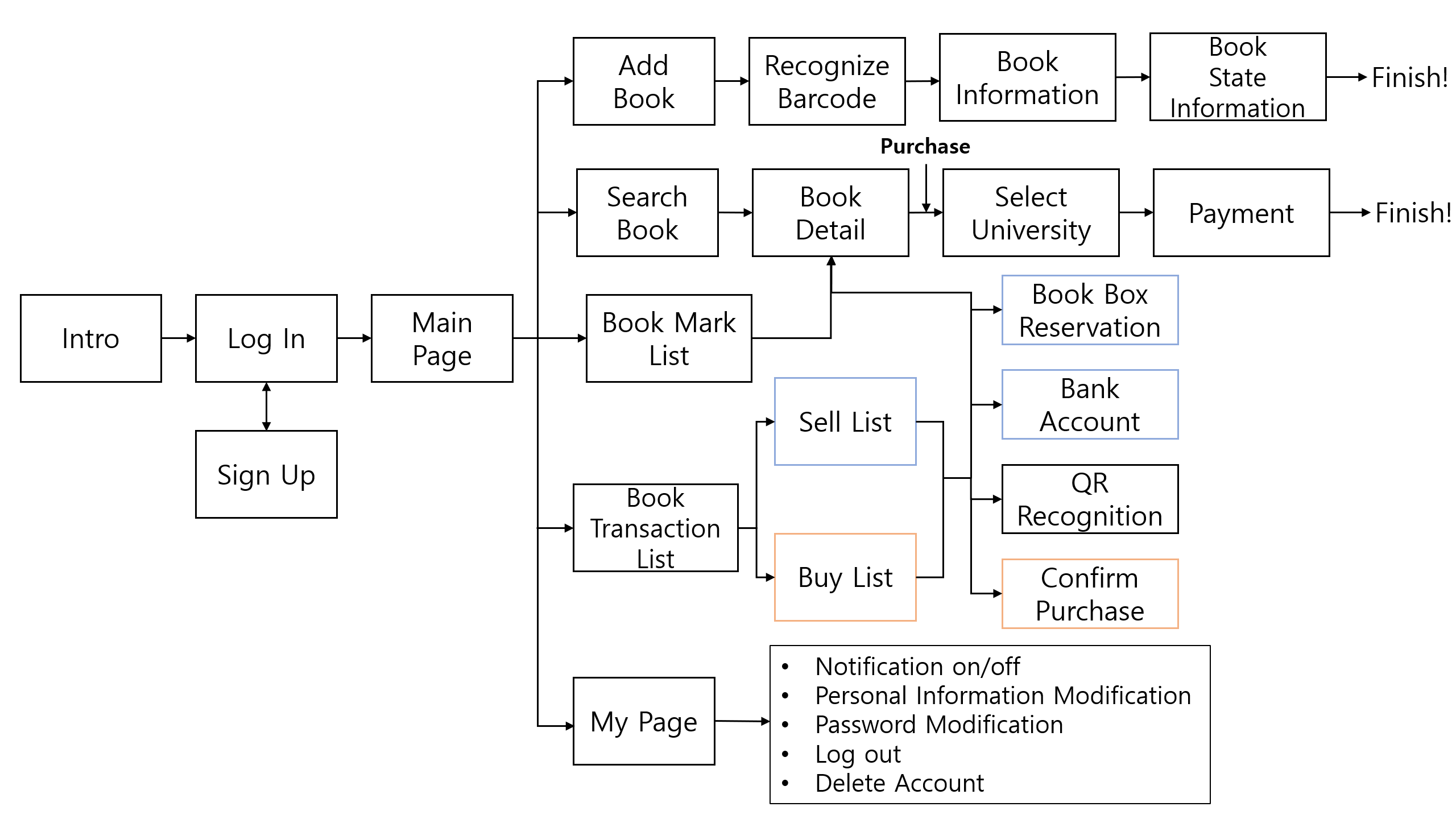
## Project Reference

|  |  |
| --- | --- |
| GitHub Organization Name : GoNaMi | |
| Application | https://github.com/Team-GoNaMi/Application |
| Server | https://github.com/Team-GoNaMi/Server |
| Book Box | https://github.com/Team-GoNaMi/BookBox |
| Documents | https://github.com/Team-GoNaMi/Docs |

# Project Implementation

# Structure of Project

## Application Outline



## Database Scheme

## Transaction Status

* **Status Button Image**

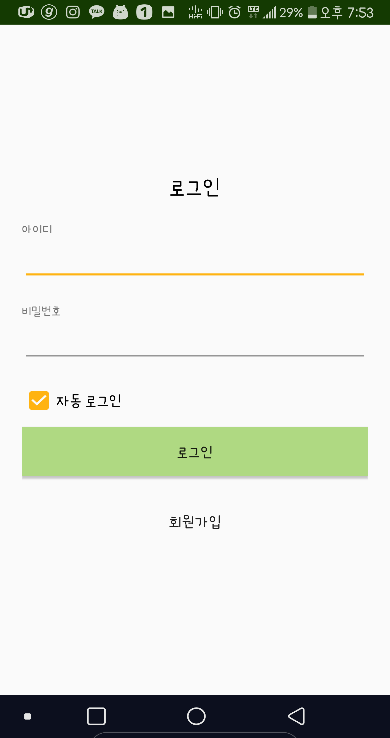
|  |  |  |
| --- | --- | --- |
| **Value** | **Seller** | **Buyer** |
| **0** |  |  |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |
| **5** |  |  |
| **6** |  |
| **7** |  |  |

# Application

## Intro



## Login and Sign Up

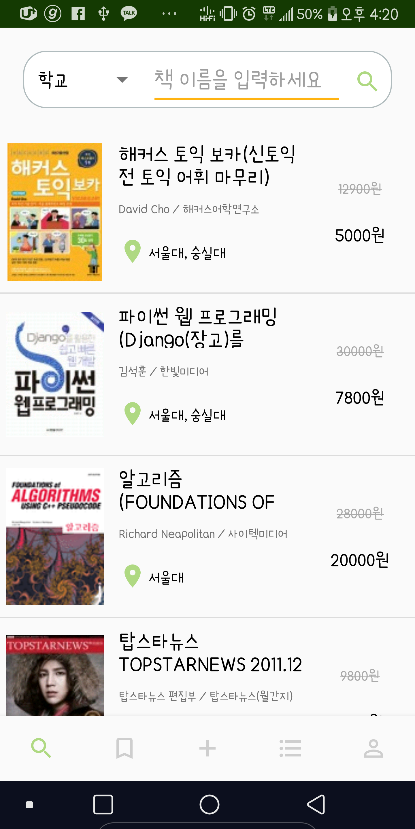
* + - **Login**

You can log in based on data registered in the DB. You can also log in automatically.

* + - **Sign up**

When registering as a member, verify that there is no information already registered in the DB and proceed with the registration. The ID and password must be at least 4 characters long.

## Search Page

****

* + - **List book information and Search book.**

It retrieves book information from the database and displays it in the customized Listview. You can search for a book based on its name and school name.

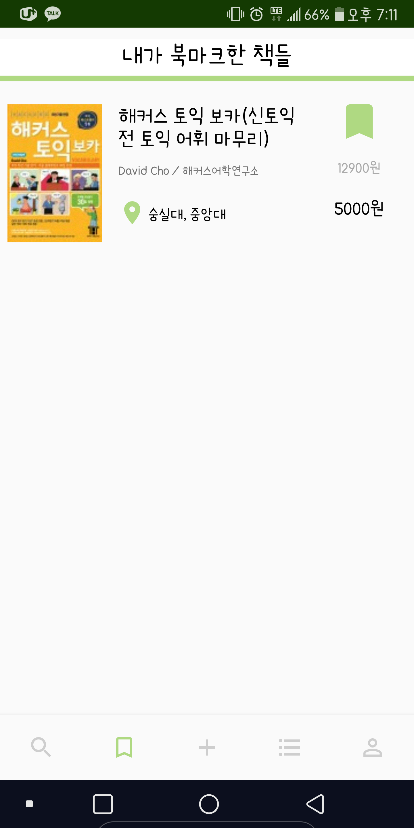
* + - **Book main image**

Using the Glide library, easily convert to an image by receiving server URL values. Receive the value of the image URL received from Naver API and show the main image of the book

|  |
| --- |
| implementation **'com.github.bumptech.glide:glide:3.7.0'** |

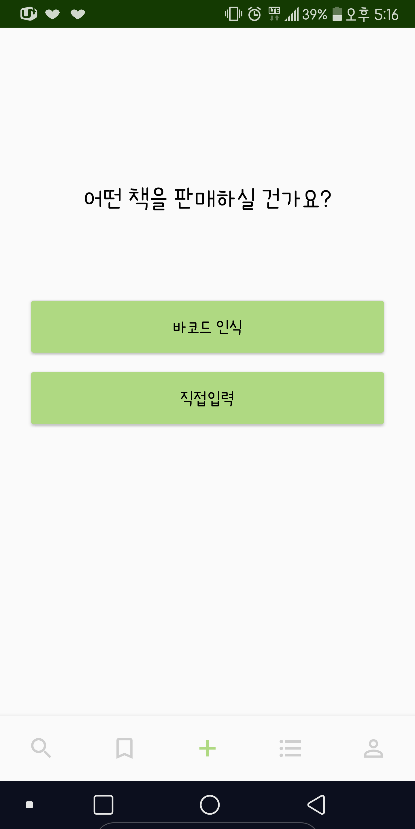
|  |
| --- |
| Glide.*with*(getContext()).load(split\_image[i]).override(***IMAGE\_WIDTH***,***IMAGE\_HIGHT***).into(bookImage); |

## Book Mark Page

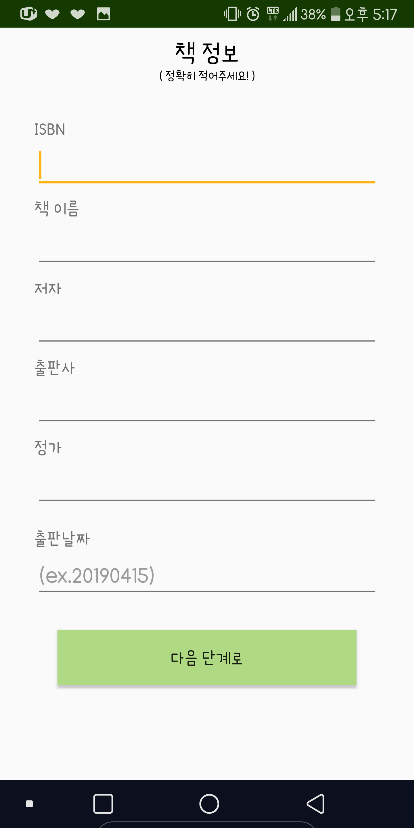
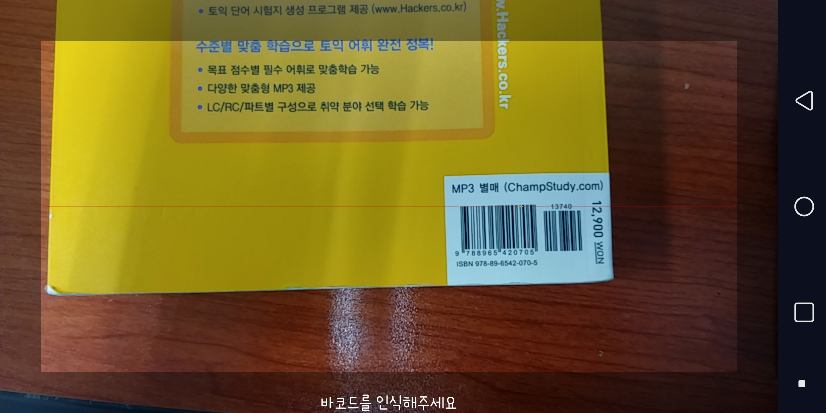
 

Book Mark information is brought from database. You can check the bookmark on the bookmark list by checking the bookmark on the book detail page.

## Adding Book Page

****

* + - **Standard book information**

**** ****

In order to add a book, you can enter the book information directly or by bar code. If you register a book using bar code, the form will fill itself up.

* + - * **Barcode recognition**

1. Barcode recognition

The ZXing library was used to implement the bar code. Then, using a variable type called IntentIntegrator, the initiateScan() function was called to implement it.

|  |
| --- |
| implementation **'com.google.zxing:core:3.3.0'** implementation **'com.journeyapps:zxing-android-embedded:3.5.0'** |

|  |
| --- |
| IntentIntegrator **scan** = **new** IntentIntegrator(**this**);  **scan**.initiateScan(); |

1. Naver API

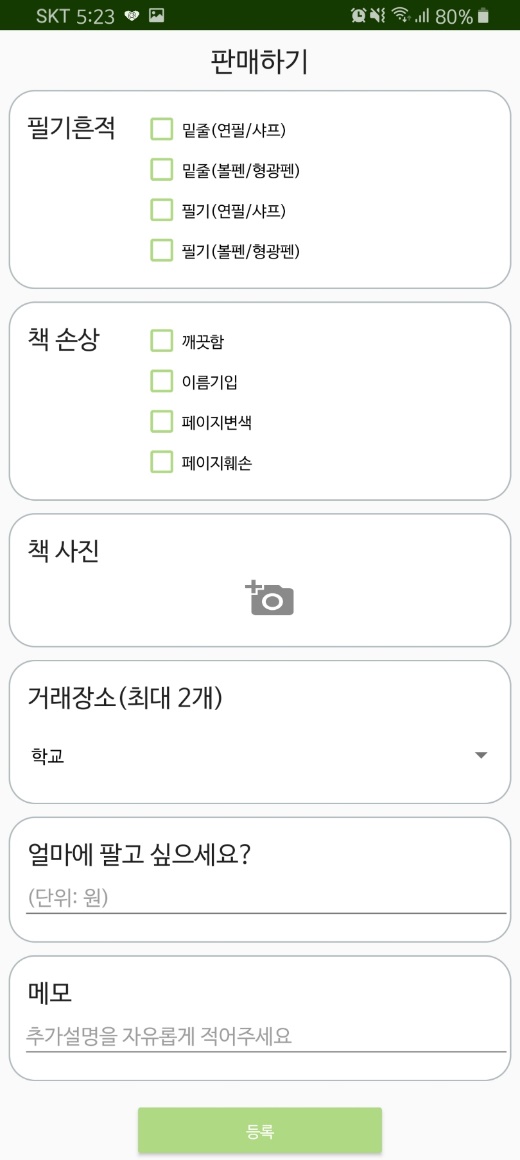
The pre-issued Client ID and Secret value connect to the URL containing the ISBN value obtained as a result of bar code recognition

|  |
| --- |
| **"https://openapi.naver.com/v1/search/book\_adv.xml?guery="** + isbn+**"&d\_isbn="**+isbn; |

Then we will receive an xml value. Only the required value will be parsed and stored. Save the title, image URL, author, price, publisher, public date, and ISBN values.

The image below is an example of the xml value obtained. Get title, image, author, price, publisher, publish date, isbn value.

* + - **Seller’s book information**

You can register the status of the book you are registering. Check the degree of handwriting/book damage. Book photos can be registered by taking photos/album selection. Up to two schools can be added to the deal. You can register prices and notes. Then all information is added in database and the image is uploaded in server.

* + - * **Add school**

The list of schools with book boxes is in the spinner. Up to two schools can choose from the value.

The pre-generated TextView and Button values are set to be visible to the user when selected. These values were also removed if the selection was deselected.

* + - * **Upload Photo**

In order to upload pictures, there are two ways to take pictures and to import pictures in the gallery.

When you take a picture, it's not that the file. Therefore, temporary images, not original images, are shown. The problem at this time was that the server could not be saved due to the rights issue. So, to import images by taking pictures, creating new image files, storing them in the gallery, and then importing the pictures.

Method to take pictures: Take a picture -> Create an image file -> Save an image file in the gallery -> Import the absolute path value of the stored location and show the image

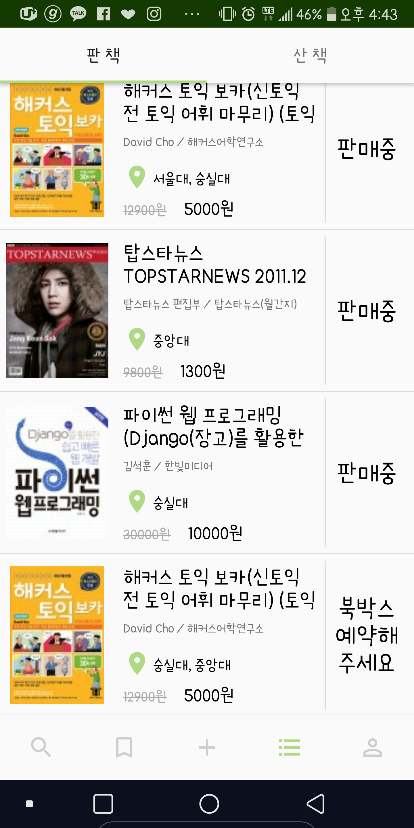
* + - * **Upload Photo to Server**

We used OKHTTP3 to upload images from application to server and we modified authority to upload image in server. we called ‘insert-photo.php’ in application and send image and book register id as parameter.

In that file, first check if there is a directory named book register id. If it does not exist, make directory named book register id. If there’s one, move uploaded file to that directory.

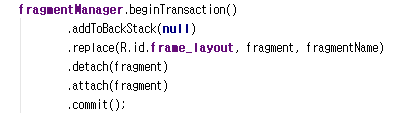
To bring the image from server, we saved image URL in database. Therefore, just by accessing URL address, we can get the image.

## Transaction List Page

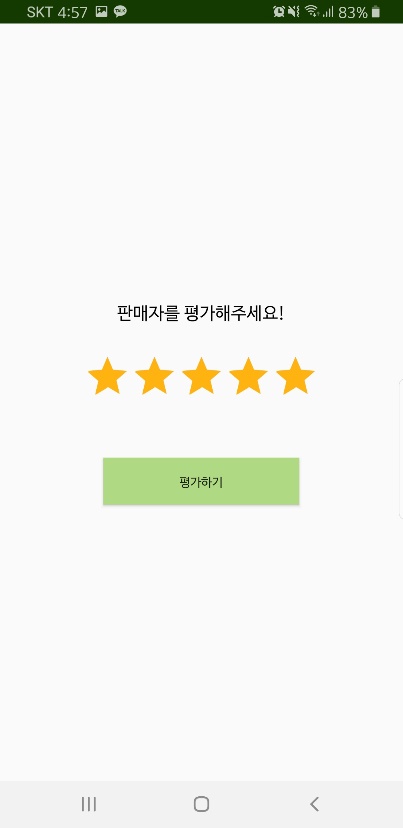
**** ****

You can check your sell or buy book list. You can also check the status of the books. All information is brought from database. Press the status button to move to the appropriate action page.

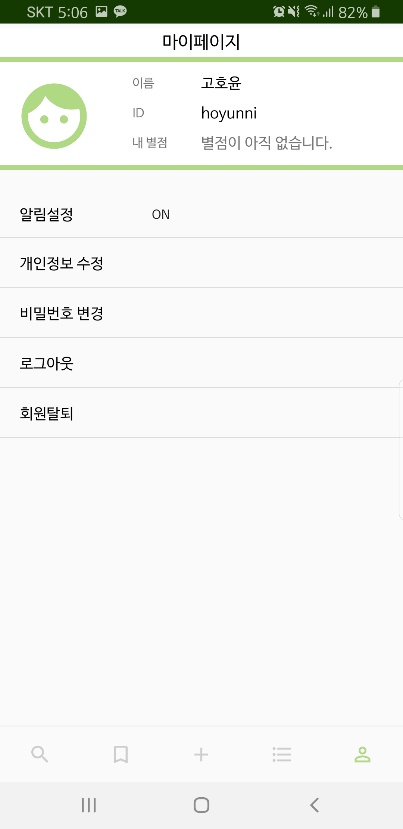
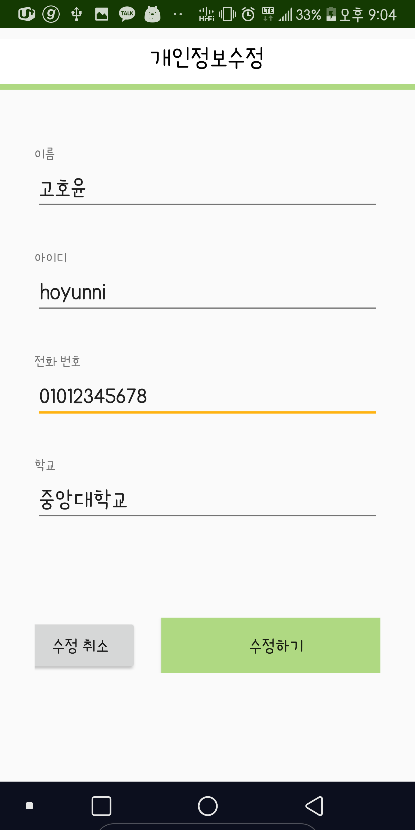
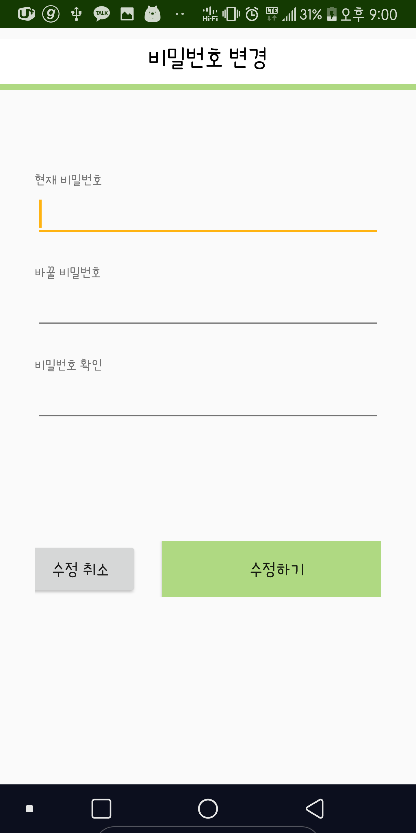
* + - **Refresh Status Button**

After seller reserve book box, put bank account, or buyer confirm the purchase, application refreshes fragment to show the most current state of transaction. Function related to refresh is detach() and attach().

* + - **Other Pages according to Transaction State**



## My Page

User can get user’s information from the DB and check it on My Page. User can see his/her name, id and rate. User can turn on/off push alert. User can modify his/her personal information and change his/her password. Also, user can log out or delete his/her account.

## Book Detail Page

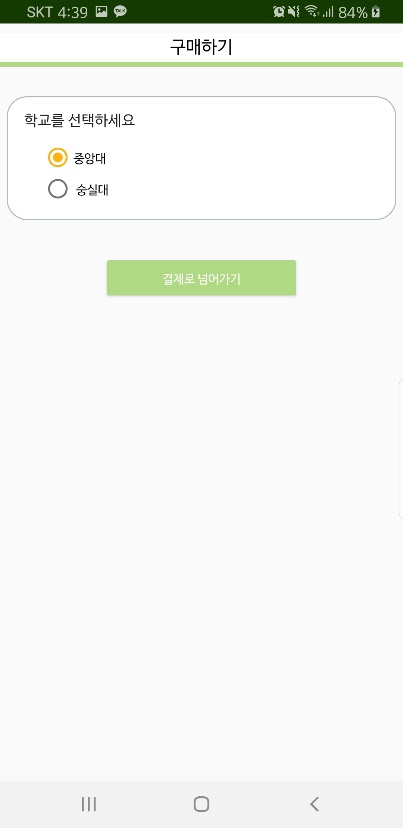
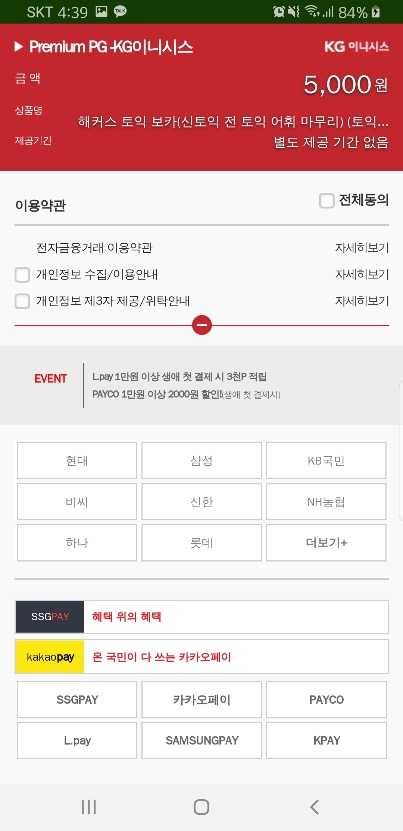


The information of the book is brought from the DB and presented in page. It also calculates the seller's rate which is from database. From this page, you can book mark the book and go to Purchase page.

* + - **Get book image that seller was registered**

The Glide library was used as in the Search page. I received the URL value of the image stored in the server and changed it to the image.

## Buying Book

You can buy a book when you choose a school to trade and complete the payment.

* + - **Iamport**

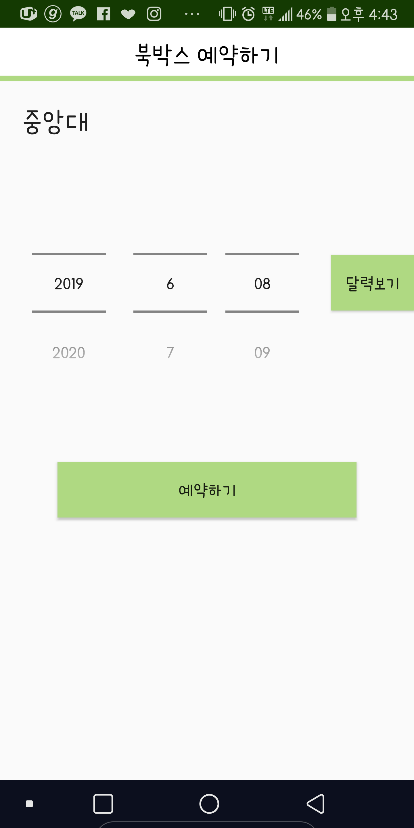
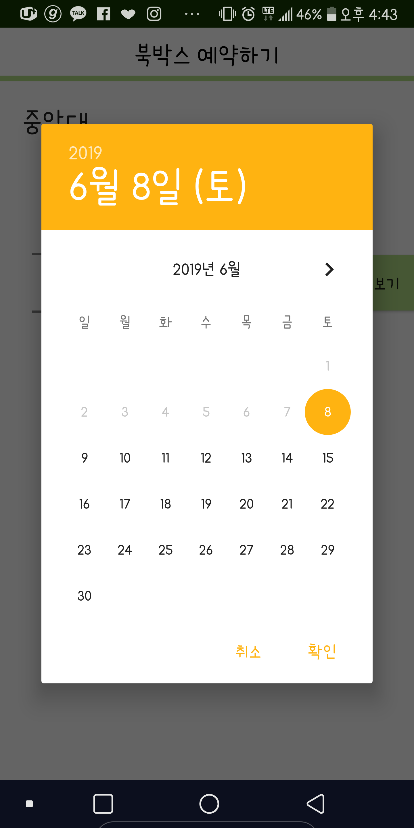
Payment page was linked using web view. The values required by Iamport were transaction number, price, product name and user telephone number. I handed over the values in posturl.

To pass this value, you set the settings of the web view to setJavaScriptEnabled (true). Therefore, the code written in JavaScript can be modified immediately. However, because a server is needed to hand over this value, PHP code was used inside JavaScript code to accept the value.

|  |
| --- |
| **webViewTransaction**.postUrl(**"https://"** + *IP\_ADDRESS* + **"/payment\_js.php"**, **postData**.getBytes()); |

Pass these values, declare the merchant identification code in JavaScript code, and call the IMP.request\_pay function to execute the Iamport payment window.

## Reserve Book box

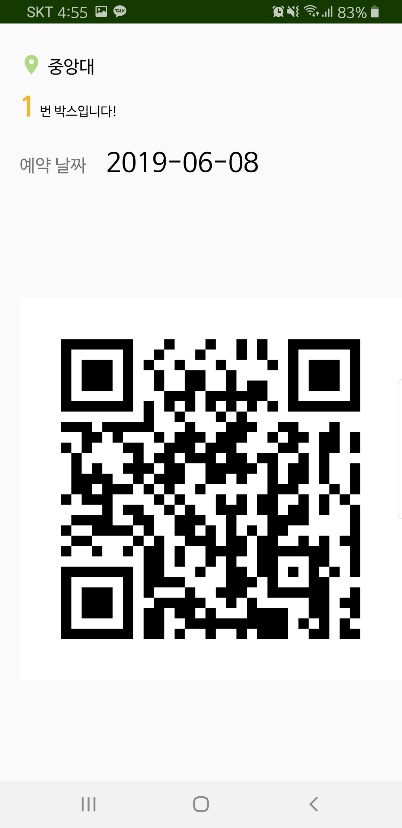
 

Select the date on which you want to reserve a bookbox, when you click Bookbox, automatically generate an empty book box number based on the selected school and date.

* + - **Select date**

This application used DatePicker to schedule a book box date. It was possible to select a date format in the form of a spinner and a date format in the form of a calendar. Furthermore, the minimum value was set to the current date to prevent the previous date from being set.

## Reservation page

****

This page allows you to check the reservation information.

* + - **Make QR code**

It is made into a String value with necessary information that can recognize the user in the book box. If the user is a seller, the value of ISBN, book registration number, and user ID was added. If the user is a buyer, the book registration number and the user ID were added.

Using the variable type QRCodeWriter, encode the barcode format by setting it to QR\_CODE. Then, Insert the generated QR code image into the bitmap.

|  |
| --- |
| *toBitmap*(qrCodeWriter.encode(contents, BarcodeFormat.***QR\_CODE***, 1000, 1000)); |

## Shared Preference

Android has a function that stores data in user’s phone which is called Shared Preference and we used this function in SaveSharedPreference. Here, we saved check state of auto login, user’s id, password, name, phone number and check state of push alert.

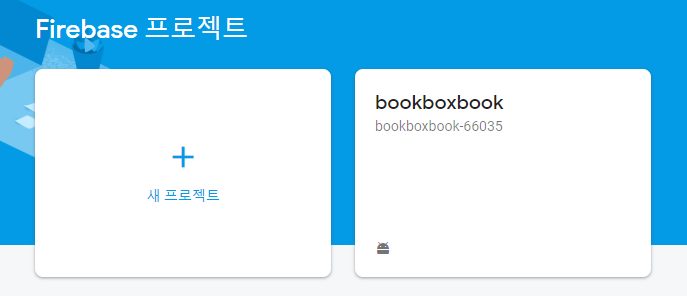
For auto login, we use user’s check state of auto login and if it is checked, we used user’s id and password stored in SaveSharedPreference. For log out and delete account, I deleted all data using clear() in SavedSharedPrefences.

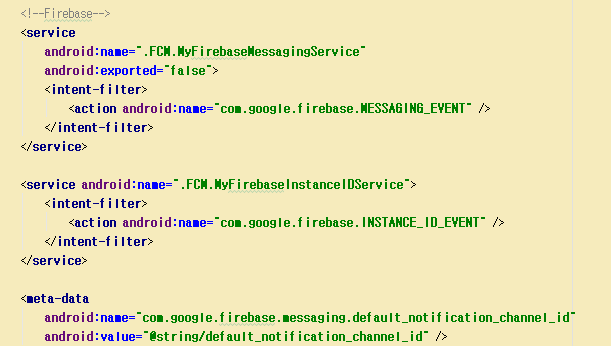
## Push Notification

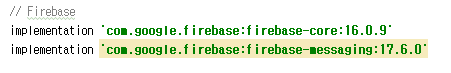
To send notification, we used FCM(Firebase Cloud Messaging). First, we registered our application to firebase and used Cloud-Messaging. Then set up to use firebase; added required codes in AndroidManifest, build.gradle and inserted google-services.json, FirebaseInstanceIdService.java and FirebaseMessagingService.java.

Then, to send push alarms from server, we installed php7.0-curl in server. Function to send notification is declared in ‘send-notification.php’. We included this file in some relevant php files to send push alarm in appropriate time like when buyer buy the book, seller put the book or buyer confirm the purchase.

To send notification to certain user, we saved user’s phone token in database. When it is time to send push alert, we get user’s token from database. Then type appropriated content of message in form of title and body. Finally call above function passing on user’s token and message as parameters.

 Registration in Firebase



Android Manifest



build.gradle

# Database

The only way to access database from application is through server. In application, we use HttpURLConnection to connect server, and bound parameters. When application calls server address(~~~~.php), then it starts its role. In server, we used php and PDO(PHP Data Object) to connect database. We used SQL queries to insert, update, get desired data.

These are the SQLs that we used to insert/get data from database.

|  |  |  |
| --- | --- | --- |
| Page | Function | SQL Query |
| Sign Up | Insert User Information | INSERT INTO member(member\_id, password, name, phonenum, school)  VALUES(:id, :hashed\_pw, :name, :phonenum, :school) |
| Insert User Token | INSERT INTO token(member\_id, token) VALUES(:id, :token) |
| Check  Duplicated ID | SELECT \*  FROM member  WHERE member\_id=:id |
| Log In | Check and Get User Information | SELECT \*  FROM member  WHERE member\_id=:id LIMIT 1 |
| Update Token | UPDATE token  SET Token=:token  WHERE member\_id=:id |
| Add Book | Check If There is Book in DB | SELECT \*  FROM book  WHERE ISBN=:isbn |
| Insert Book | INSERT INTO book(ISBN, name, author, publisher, original\_price, publish\_date, book\_image) VALUES(:isbn, :name, :author, :publisher, :original\_price, :publish\_date, :book\_image) |
| Insert User’s Book Information | INSERT INTO register\_book(book\_register\_id, ISBN, seller\_id, selling\_price, memo, buy\_avail, underline, writing, cover, damage\_page) VALUES(:book\_register\_id, :ISBN, :seller\_id, :selling\_price, :memo, :buy\_avail, :underline, :writing, :cover, :damage\_page) |
| Insert in Trade | INSERT INTO trade(book\_register\_id, state) VALUES(:book\_register\_id, :state) |
| Insert Schools of This Trade | INSERT INTO book\_school(book\_register\_id, school) VALUES(:register\_id, :school) |
| Insert Photo URL of Trade | INSERT INTO book\_photo(book\_register\_id, photo) VALUES(:register\_id, :photo) |
| Search Book | No Search Word,  No Selected  University | SELECT \*  FROM register\_book NATURAL JOIN book  WHERE buy\_avail=1 ORDER BY book\_register\_id DESC |
| No Search Word,  Selected University | SELECT \*  FROM book\_school NATURAL JOIN register\_book NATURAL JOIN book  WHERE school=:school and buy\_avail=1  ORDER BY book\_register\_id DESC |
| Search Word,  No Selected University | SELECT \*  FROM register\_book NATURAL JOIN book  WHERE name LIKE '%$searchWord%' and buy\_avail=1  ORDER BY book\_register\_id DESC |
| Search Word,  Selected University | SELECT \*  FROM book\_school NATURAL JOIN register\_book NATURAL JOIN book  WHERE name LIKE '%$searchWord%' AND school=:school and buy\_avail=1  ORDER BY book\_register\_id DESC |
| Get University  of a Book | SELECT \*  FROM book\_school  WHERE book\_register\_id=:register\_id |
| Book Detail | Get All Information about Book | SELECT \*  FROM register\_book NATURAL JOIN book  WHERE book\_register\_id=:register\_id LIMIT 1 |
| Show If this Book is Book Marked | SELECT \*  FROM book\_mark  WHERE book\_register\_id=:register\_id AND member\_id=:user\_id |
| Get University of a Book | SELECT \*  FROM book\_school  WHERE book\_register\_id=:register\_id ORDER BY school DESC |
| Get Photo URL of a Book | SELECT \*  FROM book\_photo  WHERE book\_register\_id=:register\_id |
| Get Rate of Seller | SELECT avg(rate) rate  FROM rate  GROUP BY seller\_id  HAVING seller\_id=:seller\_id |
| Buy | Insert Buyer | UPDATE trade  SET buyer\_id=:buyer\_id, state=1  WHERE book\_register\_id=:register\_id |
| Select University | UPDATE book\_school  SET selected=1  WHERE book\_register\_id=:register\_id AND school=:school |
| Set that Book is Sold | UPDATE register\_book  SET buy\_avail=0  WHERE book\_register\_id=:register\_id |
| Set Book Mark | Set Book Mark | INSERT INTO book\_mark(book\_register\_id, member\_id)  VALUES(:register\_id, :user\_id) |
| Delete Book Mark | DELETE FROM book\_mark  WHERE book\_register\_id=:register\_id AND member\_id=:user\_id |
| Book Mark List | Get Book Mark | SELECT \*  FROM book\_mark  WHERE member\_id=:member\_id  ORDER BY book\_register\_id DESC |
| Get Book Information | SELECT \*  FROM register\_book NATURAL JOIN book  WHERE book\_register\_id=:register\_id LIMIT 1 |
| Get University  of a Book | SELECT \*  FROM book\_school  WHERE book\_register\_id=:register\_id |
| Transaction List | Sell List | SELECT \*  FROM register\_book NATURAL JOIN trade  WHERE seller\_id=:member\_id  ORDER BY book\_register\_id DESC |
| Buy List | SELECT \*  FROM trade  WHERE buyer\_id=:member\_id  ORDER BY book\_register\_id DESC |
| Get Book Information | SELECT \*  FROM register\_book NATURAL JOIN book  WHERE book\_register\_id=:register\_id LIMIT 1 |
| Get University  of a Book | SELECT \*  FROM book\_school  WHERE book\_register\_id=:register\_id |
| Reserve Book Box | Find Vacant  Book Box | SELECT b.box\_id  FROM (SELECT \*  FROM book\_box  WHERE box\_id LIKE '$bb\_location%') b  LEFT JOIN (SELECT \*  FROM reserve\_bb  WHERE date=:date) r  ON b.box\_id=r.box\_id  WHERE r.date IS NULL LIMIT 1 |
| Reserve Book Box | INSERT INTO reserve\_bb(box\_id, book\_register\_id, date)  VALUES(:bb\_id, :book\_register\_id, :date) |
| Update Trade State | UPDATE trade  SET state=2  WHERE book\_register\_id=:book\_register\_id |
| QR Recognition | Get Reserved Information | SELECT \*  FROM reserve\_bb  WHERE book\_register\_id=:register\_id LIMIT 1 |
| Confirm Purchase | Update Trade State and  Insert Rate | UPDATE trade  SET state=5  WHERE book\_register\_id=:book\_register\_id |
| INSERT INTO rate(book\_register\_id, seller\_id, rate) VALUES (:register\_id, :seller\_id, :rate) |
| Got Report  -> Update  Trade State | UPDATE trade  SET state=7  WHERE book\_register\_id=:register\_id |
| SELECT seller\_id, Token  FROM register\_book JOIN token ON register\_book.seller\_id=token.member\_id  WHERE book\_register\_id=:register\_id LIMIT 1 |
| Bank Account | Insert Seller’s Bank Account | UPDATE trade  SET bank=:bank\_info, account\_num=:account\_num, state=6  WHERE book\_register\_id=:register\_id |
| My Page | Get All User Information | SELECT \*  FROM member  WHERE member\_id=:id LIMIT 1 |
| Get Rate of User | SELECT avg(rate) rate  FROM rate  GROUP BY seller\_id  HAVING seller\_id=:seller\_id |
| Personal  Information  Modification | Update  Phone number and University | UPDATE member  SET phonenum=:phonenum, school=:school  WHERE member\_id=:id |
| Password Modification | Update Password | UPDATE member  SET password=:hashed\_pw  WHERE member\_id=:id |
| Delete Account | Delete User Information | DELETE FROM member  WHERE member\_id=:member\_id |

# Book Box

## Box Material

- Body: Paper board.

- Door: Acrylic panel so that user can students easily see the inside of the book box.

- Locker: Servo motor

\* pictures below are front and side view of the Book Box.

 - Camera: Logitech web cam

- Screen: 5-inch touch screen with HDMI cable

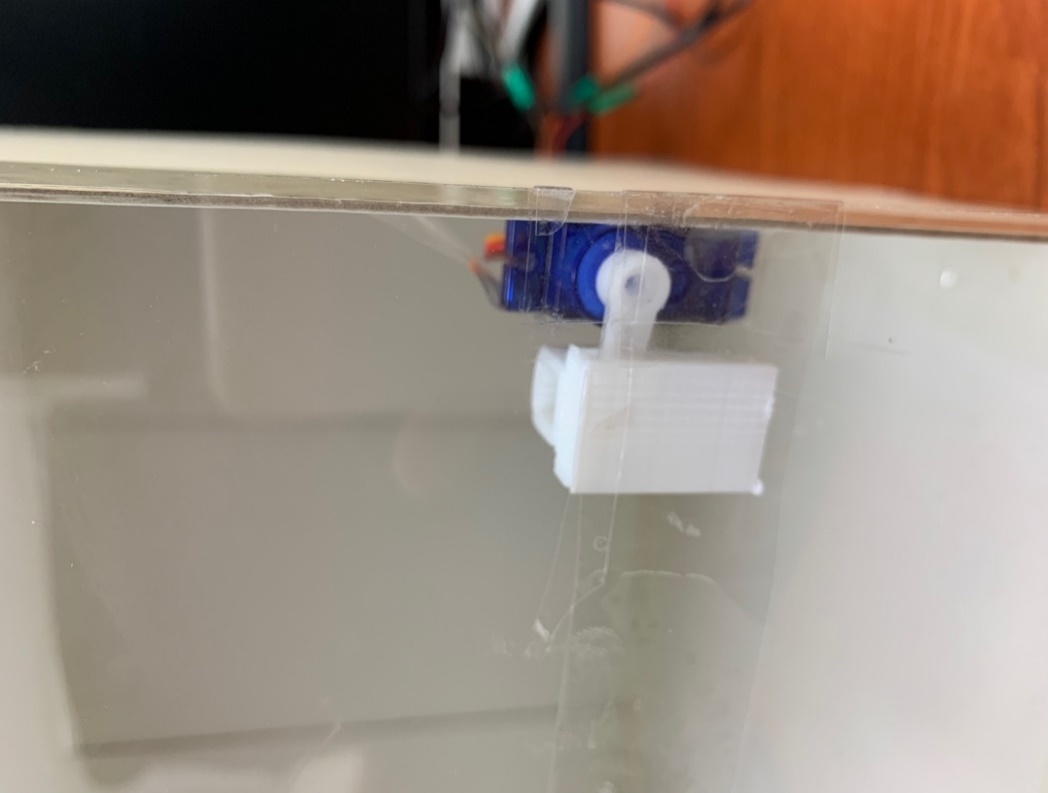


## Recognize QR/Barcode

To decode QR code and barcode is one of the main functions of Book Box. We used Logitech USB camera which have better focusing than pi camera module.

The code of recognizing and decoding the QR or barcode is processed based on the opencv and pyzbar library. And the script is written by python.

## Locker

As a locker, we choose servo motor which is operated by simple command. The power of motor was provided by Raspberry pi by using GPIO. And we custom the latch with 3D printer since the shape was not that simple.

lock

unlock



this is how the latch looks like



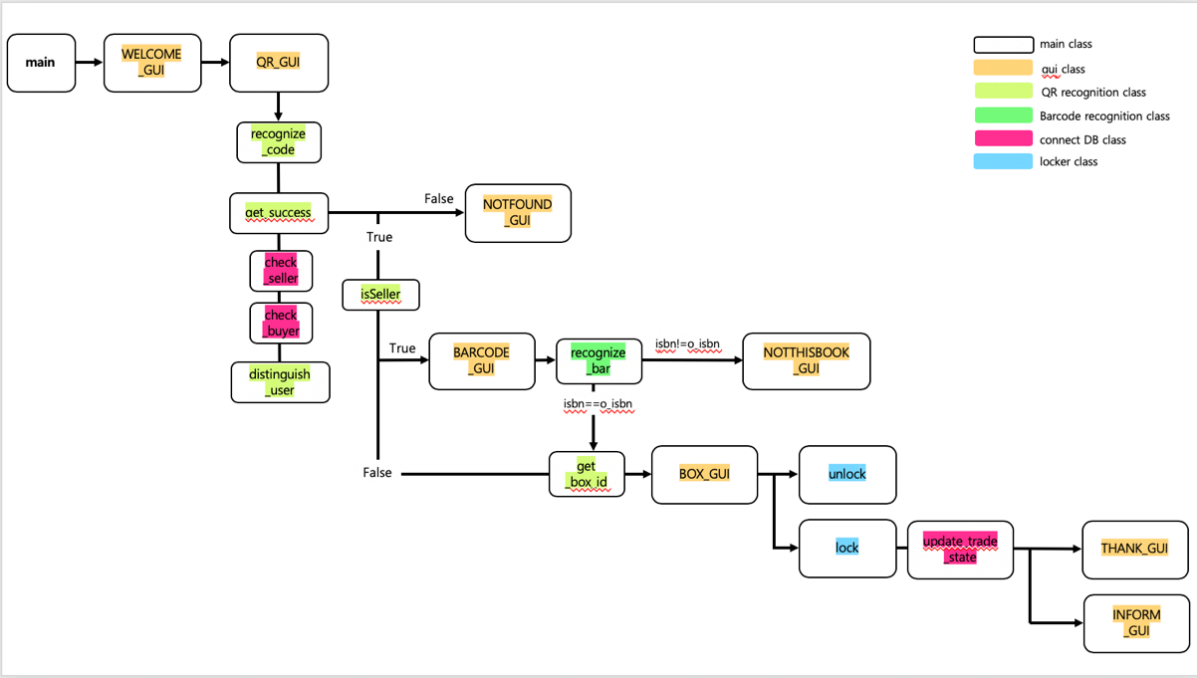
## Kiosk

Kiosk was implemented by simple python GUI library ‘tkinter’. Root page was ‘WELCOME\_GUI’ and the others such as ‘QR\_GUI’, ‘BOX\_GUI’ etc. are implemented with toplevel of root page. As the steps are progressed, toplevel pages are keep appearing and disappearing with iconify and disiconify.



## Integration

: The charts below show how the Book Box works.



# Project Result

## Project Completion

Green : Added Functions, Blue : Modification, Red : Not Complete

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | **제안** | **구현** | **결과** |
| Member Management | Sign Up  Sign In  Sign Out | Sign Up  Sign In  Sign Out  Delete Account  Information Modification | Complete,  Added Functions |
| Book Shopping | Adding Book  Searching Book  Book Detail Page  Payment | Adding Book  Searching Book  Book Detail Page  Payment | Complete |
| Book List | Book Mark  Transaction List | Book Mark  Transaction List | Complete |
| Book Box Reservation | Book Box Reservation  Sending Money to Seller | Location Reservation  Date Reservation  Sending Money to Seller  Push Notification | 90% Complete,  Modification, Added Functions, |
| Book Box | Book Box Cabinet  UI(Kiosk)  Locker  QR, Barcode Recognition  Connecting Database | Book Box Cabinet  UI(Kiosk)  Locker  QR, Barcode Recognition  Connecting Database | Complete |

## Limitation and Improvement

## Limitation

- Cannot send money to seller automatically when a buyer confirms his/her purchase.

: Through API ‘IMPORT’, we must register seller as franchise to send money. However, this is inefficient way to register all sellers whose number could reach to thousands.

- Materials of Book Box

: Camera, body of Book Box was not good enough to satisfy our standard. Firstly, camera has low resolution and auto focusing. It was hard to recognize QR and barcode fast. Second, body was not durable since it was paper. Moreover, there was just one box, so we cannot show how it works when there is more than one box.

## Improvement

- Find API which provide more efficient way of send money

- Use more high quality of materials for Book Box

## Review

**Hoyun Ko**

I started with nothing to know, so I didn’t know how to implement it. At first, I felt very frustrated about the speed of implementation and the ability to complete it. However, I’m so proud that the final demo has implemented almost all the functions I had thought of. Maybe I ran for this project this semester. I could learn many things in a short time and it was a meaningful time to look back.

**Yuseon Nam**

I was so worried about this project because I have never used database and server before and it was first time to be a team leader. And while doing, I doubt if we could really finish all this. There has been a lot of big problems doing project and sometimes I spend time doing in wrong way. However, we eventually completed all functions and I am proud of myself and team members. It was really tough, but I learned how to communicate with team members and I grew a lot while working on the project.

**Seungyun Lee**

It seems so hard and long before I start the project. However, time has passed so quickly. Through the project, I could understand the progress of the project and how to communicate with others. I am proud myself and my team members that we finish the project successfully and peacefully. I learned a lot in four months while conducting the project. This experience will last for long time.