eBJUT Final Report

Tianze Wen, Xicheng Li, Xingren Wang, Xinyun Fang, Yiwen Shang

May 31, 2019

Contents

1 Introduction

1.1 Background

In Beijing University of Technology, there is no official mobile platform for students and teachers to pass and get information. At the present time, students and teachers are still using some IM (instant messaging) software like the WeChat or QQ which is annoying and mixes life and work together. Moreover, Searching information on every platform is wasting time. In response to this situation, eBJUT aims to provide a one-stop solution service. eBJUT contains three facilities which are News, Moments, ToolBox. ToolBox include six modules, which are Printer, Lost and Found, Market, Forum, Contacts and Calendar. By using the eBJUT, students and teachers are able to get and post the latest various information in a more convenient way. Not only for searching information, but eBJUT can also help students and teachers can find the things they lost before. We also provide a place for users to communicate. eBJUT group them together to help teachers and students have a convenient campus life.

1.2 Overview

eBJUT is a mobile campus APP software, it can provide students with the latest campus information. After installing the client, students can check the latest campus announcements, check moments, and publish their new ones. In addition, it also offers Lost & Found, forum and second-hand book transaction functions. While providing convenient and practical service for students, it can also make freshmen quickly familiar with the campus environment.

1.3 Technology Used

- **Flutter**: A cross-platform UI toolkit.
- **Dart**: The basic language of Flutter.
- **PHP**: Awesome server scripting language.
- **BunnyPHP**: A PHP framework that lightweight but powerful.
- **MySQL**: A classical relational database management system (RDBMS).
- **apiDocJS**: Inline documentation service for RESTful web APIs.

The source code is available on `https://github.com/nonPointer/eBJUT`.

2 Labour Division

UCD SN.	Name	Work
17205996	Tianze Wen	Front-end Design
17206240	Xicheng Li	Arch. Design, Back-end Dev
17205963	Xingren Wang	Back-end Dev
17206139	Xinyun Fang	Back-end Dev
17205969	Yiwen Shang	Front-end Design, Back-end Dev

Table 1: Principles of Module Disivision

As shown in the table 1, the major task has been assigned to each member as their preference, besides, further details are also provided.

2.1 Tianze Wen

- User interface design
- Android client implementation
- AMAP API docking
- Code reviews

2.2 Xicheng Li

- Prototype Design with Adobe Photoshop and Adobe XD
- Deployment of Nginx, MySQL master-master replication
- Deployment and configuration of apidocjs, MailGun and CloudFlare CDN service
- Push forward the implementation of RESTful API style and Github workflows
- Code Reviews
- Typesetting, Video Clipping

2.3 Xingren Wang

- Design and complete database of Forum and Moments
- Design and complete function structure of Forum and Moments
- Complete the documentation and web API implementation for Forum and Moments
- Draw the initial interface

2.4 Xinyun Fang

- Design database of lost and found & second-hand transaction
- Implement lost and found
- Implement second-hand transaction

2.5 Yiwen Shang

- Design and implement Lost & Found and Moment UI function
- Make the front-end template of the guide page
- Increase the functionality in the toolbox, including contacts and calendar
- The initial design of the back-end of the User Account System

3 System Details

3.1 User Model (GUI) - Tianze Wen

3.1.1 User Requirements

Users are able to easily input their usernames and passwords and press the button to log in. For those who first use the application, a button on the application bar will guide them to the register page. If users forget their password, they can touch a button and reset the password in a new page.

Users are not likely to re-login every time when they open the application, therefore login status will be stored. In a short period of time, login will be executed automatically.

After a successful login, the personal information will display and users are allowed to change part of them and execute log out operation.

3.1.2 Design

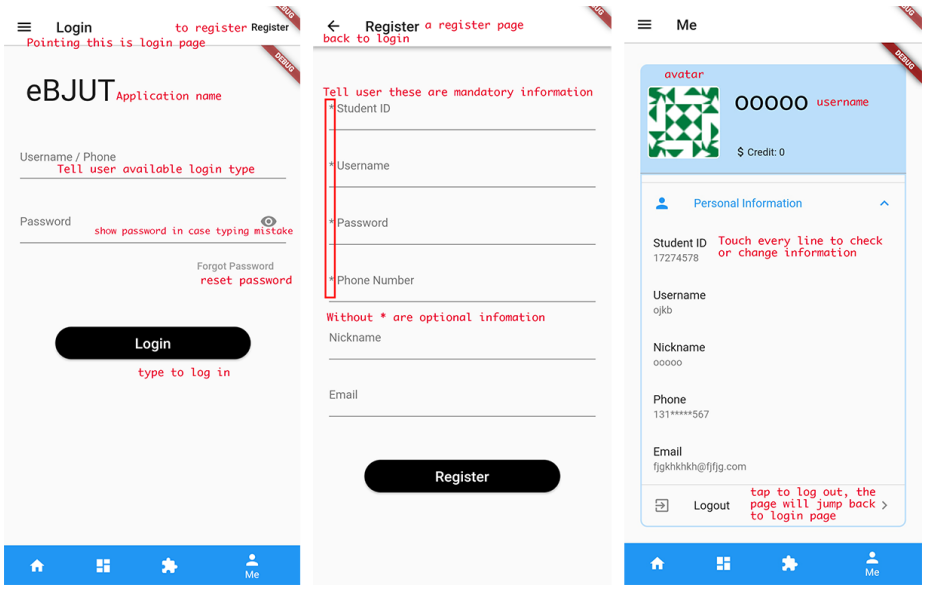
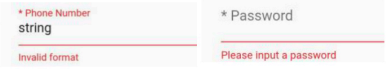


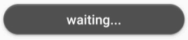
Figure 1: User Module

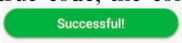
As shown in the figure 1 and figure 2.

3.1.3 Implementation

The interface is constructed by Flutter. The user model contains four pages: profile page, login page, register page, and personal page.

Information format will be checked, if any thing wrong, the color of text line will become red until user input correctly. 

After pressing login/register button, the text becomes "loading" to tell user the operation is processing 

When server return a true code, the color of buttons change into green and text become "Successful". 

Then in register page, it will route to login page automatically. For login page, it jumps to personal information page.

If a false code return by server, the color of buttons change into red and text become "Failure" which require user to fill again 

Figure 2: User Input

There is no interface in the profile page which just acts as an entrance. In the profile page, a variable named `isLogin` is created to determine whether the user is already logged in and if logged, the value of `isLogin` is changed into "yes". On the contrary, the value is null which is the default value.

Navigator is used to switching pages. Actually, all used pages are stored in a stack. Therefore, the showing page is on the top of others. When a new page needs to be displayed, the method Navigator.push() will push it into the stack and method Navigator.pop() is used to pop the current page.

To maintain the login status, local preference storage is used to store login status and user information. When user successful login, this information will be stored and the next time entering the profile page, this information will be reassignment. Then, if `isLogin` is true, it will jump to the personal information page. Otherwise, the login page is returned which requires users to re-login.

3.1.4 Testing - Login Status

- **Testing Case**: Install the application in a mobile phone, login account then exits the application, restart the application.
- **Expected Result**: The login status is maintained and no need to re-login.
- **Realistic Result**: The application reports an error and re-login is required.
- **Causes**: The type of `isLogin` is boolean initially. When users firstly enter into the profile page, the value of `isLogin` is restored from local preference

storage which is empty because of the first run. In Dart language, a boolean value cannot be null.

- **Improvements**: Change the type of `isLogin` into a string. If the value of `isLogin` is null, the application realises its first run. If the value of `isLogin` is "yes", the user is no need to log in.

3.1.5 Testing - Page Routing

- **Testing Case**: Press the return button after successful login, log out, and register respectively.
- **Expected Result**: An alert dialog will be popped up to ask users if want to exit the application when pressing the return button after a successful login or log out. For register, there is no need for users to do any operations, it will jump to the login page automatically.
- **Expected Result**: Realistic result: User returns to the login page when press return button in personal information page which cannot be approached anymore.
- **Causes**: The previous page and login page are not popped out of the stack.
- **Improvements**: In the user model, every finished page is popped to make sure users cannot access it anymore. Therefore method Navigator.pop() is used to pop the login page from the stack and then push the personal information page in.

3.1.6 Management

The user model is a part that needs close cooperation between the front and back end, but the integrated function only can be used after back-end finishing. Therefore, I drew the user interface and set aside the functions which need back-end API, making sure interface and API finished in the same period of time. Then the work of the front and back end docking began.Through continuous adjustment, all functions were finally completed.

3.2 News Feed (GUI) - Tianze Wen

3.2.1 User Requirements

The index page of the application. Users are able to check the campus news which supports switching departments and categories. Users can refresh to read the latest news and load more to view previous news as well. Moreover, some selected articles will be published on the page.

3.2.2 Design

The details of design is shown in the figure 3.2.2.

Figure 3: News Feed

3.2.3 Implementation

An `initState()` method is provided by Flutter which allows the application to execute some operations before the interface is drawn. Therefore, the program requests the news list first using the asynchronous method and then judges the return code, if success, dynamic lists are used to store titles, dates, departments, categories, and news ids in returned data. After that, for every piece of news information, a Card widget is created to arrange them. By using a loop, the news list is formed. Finally, after all of the above is completed, the page is drawn.

For the refreshment and page-turning functions, the application monitors users' gestures. When users pull down on the top of the page, the application will recall the method to get the latest list from the server and display an animation to inform users. To get more news, users pull up at the end of the news list, then the page number pluses one to request news.

When users click on the new title, it will jump to a new page to display the news body. Before painting the interface, the news id in the list is posted to the server to get news content. There are two new types, one with text and the other with external links. If no text but a link is returned, a webview is created in the page to display it, at present time, the page is similar to a browser.

3.2.4 Testing - Page Load

- **Description**: Load the page and scroll to test if the page is loaded completely.
- **Testing Case**: Click every news card to check whether all news content pages can load the news body.

- **Expected result** : Smooth sliding and each card shows news information, after clicking the cards, the news body is displayed to the user.
- **Realistic result**: As the expected result, however, when returning to the news list from the news body, the news list is lengthened. For example, twenty news is displayed originally, after returning, the list is doubled and half of the news is the same.
- **Causes**: Method list.add() is to add element to a list. After returning operation, the news list page is recalled, so methods in the page are re-executed. The news is already in the list which is added again.
- **Improvements**: Before adding news in the list, method list.clear() is executed to make sure is list is empty.

3.2.5 Management

In the beginning, I use HTML, CSS, and JavaScript to build a news page. Web front-end technology extremely rich our creation using it, I created nice animations and interface in a short period of time. However, when integrating the page I made into the application by using webview to display it, the problem has arisen. Flutter has very poor compatibility with web pages which makes some animation display incorrectly and sliding insensitively.

The time was not half over. Therefore, I made another plan: keep small updates the original one and build pages using native methods. It is glad to see that the plan B works, although some animations were sacrificed, the sliding is smooth which improves user experience. Therefore the development of the original plan is canceled, focusing on the native methods.

3.3 News Feed - Xicheng Li

At present, most universities websites still stay at Web 1.0, as a centralised host. They are so close and not user-friendly so that it is no student likely to use them.

Take the news column as an example, all news posts were categorised into a list of departments. While the irrelevant records occupy the space of web page, it is so hard users to find the most important information. Besides, in the mobile Internet era, it doesn't provide responsive web design, which directly affects the user experience.

3.3.1 User Requirements

There are three main user requirement

1. Scrape news records and store them into the database.
2. Select latest records from the database.
3. Support keywords filtering.

3.3.2 Design

1. Spider
    - Can scrape records from the website
    - Support direct access or Web VPN bypass to the website by the simulate authentication process
    - Detect external resources (attachments and images) and set up marks
    - Support fail-over or retry mechanism if network issue happens
2. Database
    - Store title, published department, published date, image status, attachment status, external URL
    - Prevent insertion of redundant records
    - Low access latency, high availability
3. CDN (Content Delivery Network)
    - Support persistent cache by page rules
4. Back-end
    - Support keywords filtering
    - Crontab execution
5. Middleware
    - JSON format
    - RESTful API style

3.3.3 Implementation

Libraries

- BeautifulSoup4
- Requests
- Regular Expression
- MySQL Connector

BeautifulSoup4 is a powerful python library that can be used to scrape information from web pages. 'It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree'[1]. Requests is an HTTP library for Python, which is used to generate HTTP requests. The regular expression, also known as RegEx and rational expression, is

[1]https://pypi.org/project/beautifulsoup4/

a powerful language that used to define searching patterns and to do matching, replacing in the string. MySQL connector is the MySQL driver for Python. By using all of them, I implemented a Python spider, which supports MySQL connection and web VPN bypass features.

During the implementation of the web VPN bypass feature in Python spider, things are not going well as usual: the login request does not work. After several tests, it shows that the login page of the web VPN interface uses a hidden type of form field to track user event, which means that I cannot send login request directly to the gateway. In this case, I use a regular expression to pick the hidden field in the first request, then send it along with the login request and keep the session cookies persistent in the whole scraping process.

3.3.4 Testing - Spider

- **Description**: Scrape news records from campus website correctly and then insert them to the database.
- **Testing Case**: Using console output and database preview to check whether the data is correct or not.
- **Expected Result**: All news records have a continuous `news_id` attribute.
- **Realistic Result**: The records in the database are having non-continuous `news_id` attribute.
- **Causes**: `INSERT IGNORE` clause instead of `INSERT` is used to prevent from runtime exception of the spider when doing redundant SQL insertion. In this case, even the new record was rejected by MySQL server, the attribute which is `AUTO INCREMENT` will be increased by 1 as usual.
- **Improvements**: Because there is only one daemon which running the news spider so that we can simply select the max `news_id` from the database before the insertion, then specify the `news_id` for each record manually.

3.3.5 Management

According to the web APIs of news feed, there are many fields in the response payload so that it highly relies on group programming and pre-shared API documentation. Therefore, we held pair-programming and group meeting regularly on-site, with Git version control system by following the Github workflow, project kanban and forced pull request with code reviews online. Besides, during the mid-term iteration, we migrated our RESTful web APIs documentation from Markdown at Github Wiki to the apiDocJS, a powerful inline documentation system with online request testing and version control strategies.

3.4 User Management System - Xicheng Li

3.4.1 User Requirements

The user management system is a core module of an online multi-user system, which should provide authentication mechanism, password encryption and session control, to name a few. Besides, enough permission levels are needed to configure different user groups.

- **Register**: Create a new user account.
- **Login**: Log in with registered account.
- **Logout**: Log in with a registered account.
- **Change Password**: Change the password for the user.
- **Validate Functions**: Detect invalid and sensitive user input.
- **User Avatar**: The strategy that store and access user avatar.
- **Access Authentication**: Verify the identity of a user who wishes to access the system and web APIs.

3.4.2 Design

- Strong and robust hash function with a random salt generator
- Access Token Control
- Customised Hierarchical Permission Structure
- OAuth 2.0 Authorisation (Upcoming)
- Single Sign On (SSO) based on LDAP (Upcoming)

3.4.3 Implementation

To ensure the hashed value of password in the database are strong and robust enough, I used two-time MD5 hash with a randomly generated salt to increase the entropy and keep in safe. Also, the unique id generator with hashed micro timestamp is used to guarantee hash collision not to happen.

The access token of a session is the unique user identity, which also generated by multi-time MD5 hash with unique id generator and micro timestamp. The token will be expired if and only if user login via another device or it reaches to the expired time.

We also embedded mailing service powered by MailGun, which can be used to verify email address, reset password and do Two-factor Authentication. Moreover, I also considered the requirements of SSO with LDAP, which can be adapted to the student accounts of BJUT.

3.4.4 Testing

- **Description**: To be more convenient, users should be able to log with his basic identity key pair: username and password, but also his phone number. In consideration of the LDAP SSO authentication in the further plan, the student number is not permitted to be used as user identity at present.
- **Testing Case**: Create a test account use temporary information and try to log in it with both authentication approaches.
- **Expected Result**:The user account can be logged in successfully without any collision.
- **Realistic Result**: The username may be conflicts with other's phone number, that would cause authentication bypass. Besides, the default nickname is picked from the username attribute if it was not specified in the registration process, which may be used to spoofing.
- **Causes**: The regular expressions that I used in the username validation are not considered to eliminate the pattern of student number and phone number. Besides, the registration process does not inspect the default nickname that was picked from the username whether is redundant to the existing record or not.
- **Improvements**: Do a double check of attributes by revised regular expressions in the registration process, also inspect the nickname collision. Besides, the character set of username and nickname are also strictly limited.

3.4.5 Management

In the first stage of work division, Yiwen was assigned to take in charge of the user account system in the backend. However, it seems that she is confused about the implementation of session control and the authentication mechanism among front-end and back-end, also, she presented interests in UI design and flutter at the meanwhile. So that at the beginning of the mid-term iteration, we made a discussion and decided to adjust the work and let me take in charge of this part.

3.5 Moment - Xingren Wang

3.5.1 User Requirements

Users can use our project to read, post and delete their daily status. They also can send or delete comments and add or cancel like to friends' moments. Users can know the category that each thread belonging to.

When posting moments and comments, namely posting information on our project, the user can send a plain text with character limitation as to the content

of moment or comment. Users can know the number of likes and comments from the screen.

3.5.2 Design

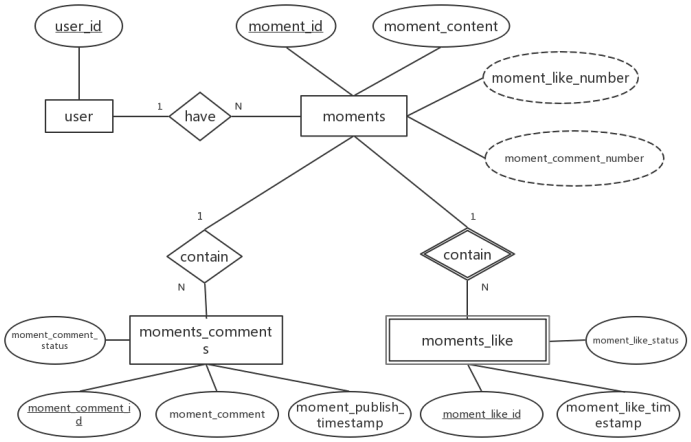


Figure 4: database for moment

Based on the user requirement, the Moments has the following functions:

- Get moments
- Post moments
- Delete moments
- Get comments from friends
- Post comments
- Delete comment
- Set the number of moments like
- Set the number of moment's comment

See appendix for more details of each interface

The order of function I create post moments, post comments, get moments, get comments, count number. For post moment and comment, it acquires parameter from the front-end then inserts into the database. For getting moments and comments, select all the information from the database as a list and then pass it to front-end. To ensure the integrity of the database, there is an attribute called `status`. `status = 1` means has been deleted while `status = 0` means to publish.

3.5.3 Implementation

MySQL will be used to create a database which contains the number of moments and comments, the content of each of them, the number of like and so on. Besides using MySQL, functions still need to use PHP to achieve cloud storing and connect to the APP. In our final version, the SQL statements are encapsulated by using the frame, and each class relates to a table in the database. All the returned value is encapsulated in a class in this version.

As for each function, it restricts the frequency of posting by counting differ between the latest related operation and the current time. restricts the frequency means to send or delete one moment content or comment per 60 seconds, for adding or deleting like, the limit is per 5 seconds. The reason to restrict it is to avoid operation failure leads to the lost message due to the problem from the network and prevent some malicious behaviour. It also can show 20 moments per page by using MySQL statement called order and limit and at the end of each message, it can show the number of people who like it and the number of people who write a comment and the content of each comment by counting column statement in SQL.

3.5.4 Testing - Compute time difference

- **Description**: Need to count time differ to make sure update message completely and avoid frequent operation.
- **Testing Case**: When counting the time difference between the current time and the latest operation time, the simpler way is using subtraction to make timestamps difference. The feature of it is like 'timestamp minus timestamp'. It is feasible in MySQL.
- **Expected Result**: The result I want is an integer number.
- **Realistic Result**: It cannot be processed in PHP when one of the timestamps is selected from the database. The result from PHP is neither an integer nor true.
- **Causes**: It is set in PHP.
- **Improvements**: Typecast the timestamp from the database from timestamp to integer then subtraction it. The result so far is the result is correct but it is still not an integer so I have to type cast it again to get integer type.

3.5.5 Testing - Update number

- **Description**: When delete like or comment, need to update the time of like or comment rather than the time of moment.
- **Expected Result**:The result I want is only to update the number of like and the time users publish or delete like, which means the rest of things related to like such as moment and comment cannot be changed.

- **Realistic Result**: When updating the number of likes, the moment timestamp(`moment_timestamp`) corresponding to the time the users update moment like(`moment_like_timestamp`) has been changed.
- **Causes**: The reason for it is when the type of an attribute is a timestamp and its set is NOT NULL with no default value, it will automatically add an extra condition called `ON UPDATE CURRENT_TIMESTAMP`. In my database, both of `moment_timestamp` and `moment_like_timestamp` satisfied this condition. This means when updating the number of likes, `moment_timestamp` will update automatically.
- **Improvements**:The solution of it is to remove it manually. The statement of it has been shown in the figure.

3.5.6 Testing- Count number

- **Description**: The number of comment and like need to be counted. [Realistic Result] the number may not be correct.
- **Testing Case**: The number may not be correct.
- **Expected Result**:
- **Realistic Result**: Hopping to get correct number of it.
- **Causes**: To reduce the time complexity, the number of like and the comment will add 1 by using `num = num + 1` when updating the number of like and comment. It may have conflict when two or more people do the same calculation at the same time, which means the operation `num = num + 1` may make mistake and then get the wrong number of likes and comments. This error can not be fixed automatically.
- **Improvements**: We decide to use `COUNT ( )` statement to calculate to guarantee the accuracy of the figure although the complexity of time is O(n).

3.6 Forum - Xingren Wang

3.6.1 User Requirements

Users can use our project to read, post and delete their academic information. They also can send or delete the reply. In the forum, the categories help users to filtrate the information they want.

When posting a thread and reply, the user can send a plain text without a picture and limit the character number of texts as the content of thread or reply. Users can know the number of replies from the screen.

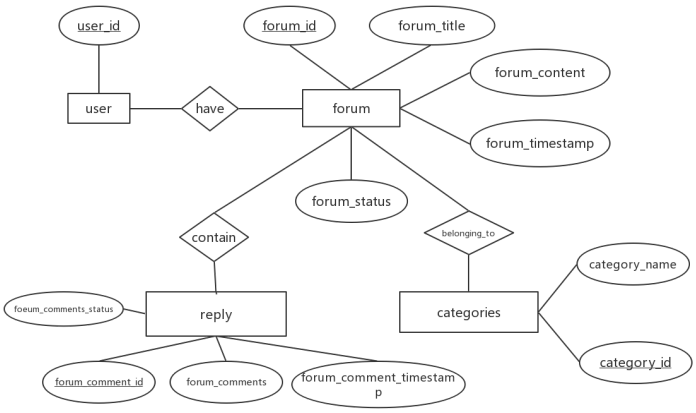


Figure 5: database for forum

3.6.2 Design

Based on the user requirement, the Forum has the following function:

- Get thread title
- Get thread and reply
- Get categories
- Post category
- Post thread and reply
- Delete thread
- Delete reply
- Set the number reply

See appendix for more details of each interface

Users can get a list of thread titles with particular The information of thread title, thread content, the category that has existed and replies, that select from the user inserts into the database are being allowed. For getting thread and reply, select all the information from the database as a list and then show it to the users. To ensure the integrity of the database, it is similar to Moments, there is an attribute called 'status'. 'status = 1' means has been deleted while 'status = 0' means to publish.

3.6.3 Implementation

MySQL will be used to create a database that contains the number of replies, the title of each thread, the list of categories and so on. Besides using MySQL, this function still needs to use PHP to achieve cloud storing and front-end and back-end interactive.

In our final version, the SQL statements are encapsulated by using the frame, and each class relates to a table in the database. All the returned value is encapsulated in a class in this version.

It is important to mention that only administrators can post categories to make sure the normalisation of classification facilitates users to find the required threads.

As for each function, the way and the function to restrict the frequency of posting are similar to Moments. Frequency of operating one thread or reply is also once per minute, for the operation of like, the limit is still per 5 seconds. The reason for it also is the same as Moments.

It also can show 20 thread titles in the same category per page by using the MySQL statement called order and limit. At the end of each message, it can show the number of people who write a reply.

3.6.4 Testing - RESTful API

- **Description**: We use delete, get, post, patch to achieve RESTful API so that it can Dependency Injection (DI).
- **Expected Result**: It can delete it successfully.
- **Realistic Result**: The operation of delete cannot be used on front-end but 'delete' can not be used.
- **Causes**:The function parameter part of our framework is taken from PHP's `$_REQUEST` but the definition of `$_REQUEST` is that his content is `$_GET`, `$_POST` and `$_COOKIE`. In other words, it can only accept fields from these three places, which means if we need to take parameters outside these three places, we need to go through other means.
- **Improvements**: We replace the patch with delete in all delete operation such as delete thread. If we use a patch for a function, it needs to use PHP function that is encapsulated by ourselves because PHP does not support using the patch, delete to acquire parameter.

3.6.5 Testing - Select multiple categories

- **Description**: The forum can able to select multiple categories.
- **Expected Result**: Can get multiple categories.
- **Realistic Result**: Only can get one category.

- **Improvements**: Spilled field by commas, then select information from the database by using each category. After that, all the information is grouped together and passed it to the front-end.

3.6.6 Testing - Reset avatars size

- **Description**: The avatars need to show on the screen to show who post thread and reply.
- **Testing Case**: We fetching avatar from the web and the size of it is 64x64.
- **Expected Result**: It can fetch avatars successfully.
- **Realistic Result**: Sometimes it is fast to acquire an avatar while sometime it may reply error owing to the Internet problem.
- **Causes**: Because the avatar server is oversea, so the network is unstable.
- **Improvements**: I change the size of the avatar to reduce the influence of the Internet. [Expected Result] It can fetch avatars successfully.

3.6.7 Management

Different from the rest of our group members, it is the first time for me to use Git so I spent about 2 weeks to learn what it is and how to use that. From the third to seventh weeks, the V1 was finished. From the eighth to thirteen weeks, the code of V2 was completed. Since the thirteen weeks, start to complete API and begin to achieve RESTful API. About one week ago, the V2 was complete and turn to finish project report.

From beginning to end, our group uses Git to build up our project. Using apidoc to passing information such URL link and data from the back-end to the front-end. To achieve the function of like, the front-end need to know the user weather like it before so they need a list of likes. They put forward a request and I create it and it can use it very well. In the Git, our group leader also asks questions and make improvements for us when we submit pull request.

3.7 Lost Card - Xinyun Fang

3.7.1 User Requirements

- Glance over all information that the lost or get the student card.
- Publish the necessary information on the platform if the user gets or lost a student card.
- Send a message to the user if he or she's student card has been getting.
- Update the record status if the card has been back to the owner.
- Delete the record if the author make ant mistakes.

3.7.2 Design

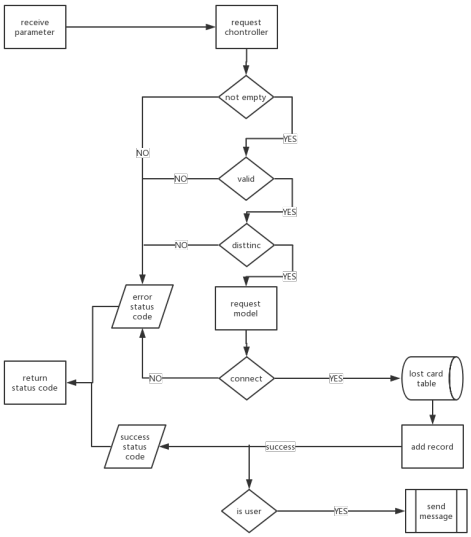


Figure 6: flow chart for add lost card

This feature should include three main functions: get the records list, add the record, and update the status of the records.

The get function returns a list of data including the lost card id, some details, author's information, and communication ways. The add function requires the lost card id, the details, and the communication way from the author and add to the database. The author can choose either the phone number that they bound or the WeChat/QQ/another phone number. This function also includes the sending message system, which can send a short message to the owner if he or she is our user. The update function is used to update the status code. Since we are not deleting any data in the database, the status code is used to decide the record is waiting, finished or deleted.

We have one table for lost cards. Since we considered that the 'lost' and 'found' need the same information, it is not required to divide them into two different tables. Therefore, the 'lost' and 'found' are in the same table which has been divided by the attribute `lost_card_type`. In addition, except the type, states and the information from the user, the database also includes the id, which is the primary key, the time when publishing the record, and the time when updating the record. The time is used to help users double-check whether the lost card is the card he or she found. It is also be used to check whether someone requires to add the data too frequent.

The lost student card is the key point of the record. Therefore, if there are some mistakes that the user applies to add a new record without the lost card id,

the function should return the status code 400, missing message. In addition, if a user request same information too frequency (for example, if the viewer does not change because of the bed network, the user may click the bottom again and again), the function will check if the same user requires the same student card in five minutes, the new record will not allowed to be add. In addition, only the author or the owner of the card can check the status.

3.7.3 Implementation

We use the BunnyPHP frame to implement MVC structure.

The model is used for interacting with the database. The model class includes three basic functions, add, get, update and an extra function to check if the add requires is too frequent. All of these functions are using the basic SQL function from the frame. Specifically, the check function uses constraints `TIMESTAMPDIFF` to check whether it has been requested too frequent, and the get function use `JOIN` to get the user information from another table. In addition, only when the record id and the current user id are both equals to the data in the database, the SQL statement can select this data and do the updating. For the SMS service, we have a model that engages the API from Ali.

The controller is used to communicate with the front-end. The controller model also includes getting, add, update function which corresponds with the function in the model. For transform the parameters, we use HTTP method get to the function get, post to the function add, and patch to the function update. For each function, we check whether the parameter is not empty and whether the record is valid. It returns the status code and data (if necessary).

3.7.4 Testing - Empty Parameter

- **Description**: I use the function `empty()` to check all the parameters that are empty or not. If the parameter is empty, I need to return the status code 'missing message'.
- **Expected Result**: If the parameter is empty, return true, else return false.
- **Realistic Result 1**: If I add a parameter as '0', it will be distinguished as empty, but if I add '1' it is correct.
- **Causes 1**: The integer '0' will be distinguished as empty in function `empty()`.
- **Improvements 1**: Using `is_null()` function to check empty is impossible because the frame is based on dependency injection, which will give something to the parameter which is null. Therefore, I use the `empty()` with an extra check that the parameter does not equal to zero.
- **Realistic Result 2**: Both the empty and the '0' can pass the checking, which means that it cannot distinguish the empty

- **Causes 2**: The Hypertext Transfer Protocol is not data type sensitive. Therefore, the parameters cannot be distinguished as an integer. So the PHP disposes of those parameters in form-data as String.
- **Improvements 2**: Change the integer 0 to the String '0'

3.8 Second-hand Transaction - Xinyun Fang

3.8.1 User Requirements

- Glance over all the information about the second-hand transaction
- Search what they want quickly
- Glance over the information in the category they need
- Publish the selling information on the platform.
- Update the record status if the transaction finished
- Delete the record if the author make ant mistakes

3.8.2 Design

This feature should include four main functions: get the records list, add the record, update the status of the records and search the record. It also includes an extra function that returns the categories.

The get function returns a list of data including the title, some details, seller's information, and communication ways. The add function requires the title, the details, and the communication way from the seller and add to the database. The seller can choose either the phone number that they bound or the WeChat/QQ/another phone number. The details can be empty if the seller thinks that the title is enough to describe the book or item that they want to sell. The update function is used to update the status code. Since we are not deleting any data in the database, the status code is used to decide the record is waiting, finished or deleted. The search function also returns the records list, but it only returns the records whose title or detail contains the keyword that the user wants to search. The records that contain the keyword in the title will in the front of those records that contain the keyword in detail. The function for categories should return either all the categories or the category in specific id.

Except for the states and the information from the user, the database also includes the id, which is the primary key, the time when publishing the record, and the time when updating the record. The time is used to check whether someone requires to add the data too frequent. If a user request same information too frequency (for example, if the viewer does not change because of the bed network, the user may click the bottom again and again), the function will check if the same user requires the same title in five minutes, the new record will not allowed to be add. In addition, only the seller can check the status. In addition, we create another table to store the supporting categories.

3.8.3 Implementation

We use the BunnyPHP frame to implement MVC structure.

The model is used for interacting with the database. The model class includes four basic functions, add, get, update, search and an extra function to check if the add requires is too frequent. All of these functions are using the basic SQL function from the frame. Specifically, the check function uses constraints `TIMESTAMPDIFF` to check whether it has been requested too frequent, and the get function uses `JOIN` to get the user information from another table. The `CASE, WHEN` has been used to order the records. In addition, only when the record id and the current user id are both equals to the data in the database, the SQL statement can select this data and do the updating.

The controller is used to communicate with the front-end. The controller model also includes getting, add, update, search function which corresponds with the function in the model. For transform the parameters, we use the HTTP method get to the function get and search, post to the function add, and patch to the function update. For each function, we check whether the parameter is not empty and whether the record is valid. It returns the status code and data (if necessary).

3.8.4 Testing - Patch Parameter

- **Description**:The update function cannot receive any parameter.
- **Testing Case**: Use `json_encode()` to return the parameters and check its value
- **Expected Result**: Receive the value.
- **Realistic Result**: The parameter is empty.
- **Causes**: The frame is based on `$_REQUEST`, so it can only receive the parameters from HTTP method post, get or cookie. If we want to use other method we need to use other functions.
- **Improvements**: We call another function from the frame to identify the parameter and decode by ourselves.

3.8.5 Testing – Bind Parameters

- **Description**: At the search function, I need to bound the same parameter twice in the SQL statement. I input one parameter and bound it twice, but it not works.
- **Testing Case**: Use debug() function from frame to return the SQL statement, execute it in MySQL. Comment a line one by one to locate the mistake.
- **Expected Result**: Get the data.

- **Realistic Result**: Return an error.
- **Causes**: In PHP, for bound parameters, the number of parameters that we input has to be equal to the number of places that we need to bind.
- **Improvements**: Input the same parameter twice with a different name.

3.8.6 Management

We use the API doc to show the parameters and return data to the front end. Also, we use git to manage the code. We have a group meeting each week.

At the group meeting, the front-end and the back-end discuss the implementation of some functions. For example, at first, the get lost and found only return the records that are lost or found. However, we plan to simply the lost and found into one page. So the front-end students ask for the whole lost and found record. After discussing, we decided that we use another integer for type to represent the whole records.

About April 15, I finish the code for v1, which including three basic functions for lost and found, lost card, second book transaction, second item transaction. At about May 5, I implement these functions in v2 and add the category in book transactions. After that, I did the upgrade function. First the search function and then the message sending.

3.9 Moment (GUI) - Yiwen Shang

3.9.1 User Requirements

- Browse current moments, include the moment content, the publishers' nickname, the avatar, how long ago was it posted, how many likes and comments each moment has.
- Make the thumbs-up and comments.
- Browse comments of each moment.
- Delete their moment if they want

3.9.2 Design

In the Graphical User Interface, the user would like to see the current moments first after they click in. The information in this stream should be concise and clear. The primary information of user concern is the content of moments, so the line spacing of text should not be too small, it should be easy for users to read.

Posters' profile picture and nickname can be a visual way to let the user know who posts it. Below publisher's nickname is post time, it displays how long ago did this moment posted, it convenience user to determine if this moment is still fresh to comment.

Figure 7: Moment

There is an icon button of ”like” set below content for the user to thumbs-up this moment. If the user would like to comment to this moment, the user can click the icon button of ”comment” set beside like icon or click the content of moment to enter the page of moment detail. In this page, there is a fixed red button in the right corner with an ”add” icon, this button is for the user to create a new comment.

The delete button is below the moment content; only the moment publisher can see this button. Comments also have this function.

3.9.3 Implementation

I am responsible for the UI.

The UI frame I use is flutter, it is a mobile UI framework developed by Google, and it can quickly build the high-quality native user interface on iOS and Android. The network development language I use is Dart, and it is object-oriented programing language. In my developing, Android Studio is an integrated development tool I use. To build this function, it mainly divided into two parts, build UI and do network request.

In flutter, everything is a widget. I set moment as a stateful widget, because this page usually needs to redraw, whenever user drop-down to refresh stream or pull up to turn the page. Under ”build” widget of the moment, the scaffold’s

body is a column. It has two states, one is to show the moment stream, another is an animation of circular process indicator. When entering this page, initialization begins, it will make a network request "get" to get data from the database. The current page shows a circular process indicator, until "get" request success and back code "201", then do "set state", the build will receive that code and through trinomial operation rule to detect whether code is "201" or not to decide if there is time to show the moment stream page. If lack this judgement, it may occur a situation that page rendering happens before getting data, it will cause an error.

3.9.4 Testing - Get Moment List

- **Description**: The information stream I need to display in UI is stored in the database. Therefore I need to get data from the database, then store it in a local file, set it as global variables in a suitable data structure, when I need to use it in UI, the good data structure will make it easy to implement. In dart language, there is no array, so I store those data in List. limitation of the number of moments in a page we set 20, so I set lists in static, with fixed length 20. After getting data from the HTTP network response, I use add method to store data into lists, and use index i to reference data to their place.
- **Testing Case**: Get lists of information stream from the database using the get method of network required.
- **Expected Result**: My expectation is in a page there are 20 cards, in each card, there is information about that moment, include publisher's name, head portrait, time stamp, main content, number of likes and comments. When I change page, it should back to the hand of moment stream, and all moments in this stream should be the next 20 moments.
- **Realistic Result**: The realistic situation is when I require for network response in getting, it reports errors "'data!=null': is not true."
- **Causes**: The reason for this error is because, in get method, it requires 20 numbers of data to store in lists, but there are less than 20 data that get from the database, so some of the data stores in it can be null and it is invalid.
- **Improvements**: My solution is to change global variables lists from static to dynamic, the length is unfixed, it depends on the length of data it gets from the database, therefore all data store in the list will not be null.

3.9.5 Testing - Delete Moment & Comment

- **Description**: There is a delete function of moment and moment comment.

- **Testing Case**: To delete a moment or comment, user click a button and it will call delete method then refresh UI. There is a network request statement was “await dio.delete (url, data:fomData)”.
- **Expected Result**: My expectation is delete code back with data[”code”]=201.
- **Realistic Result**: There was a compile error in this statement, data[”code”]=null.
- **Causes**: Because in the package of Dart, there are no data can be transferred in form data of delete sentence. However, if those data wrote in URL to transfer, usertoken might be leaked, information security will be threatened.
- **Improvements**: Solution is to ensure information security, change delete sentence to patch sentence, it can support form data transport.

3.10 Lost and Found (GUI) - Yiwen Shang

3.10.1 User Requirements

- Browse current Lost and Found notice stream.
- Post new Lost and Found notice if they lost or found something.
- Delete their notice if they like.
- Update their notice states as finish if lost property is found by someone or found property is claim by someone.

3.10.2 Design

Lost and Found is a function for the user to post Lost and Found notices, easy for users to find or return lost property.

There are two lists, one is particular list for a student card, one is for other items, the user can view it by sliding left and right.

On the lower right button is a red button with an add icon, it is a button to create a new notice. Double click it can directly enter creating a page of ”lost”, if press white button, it will enter creating a page of ”found”. This design fits the inertia of the finger. In the creating page, after file the title and content, the user needs to choose contact type, can be the phone, WeChat or QQ, if the user chooses ”Registration Phone number”, whatever they fill in the contact information field, it will always display their registered phone number. In order to prevent malicious operations, the publication of lost and found is limited to one time per person per day.

If the lost item is returned or found property is claimed or long time no one contact, the publisher can press the button below to reset the status of this notice. After reset, that notice will change color, and the button will disappear. Deleted notice will not in the notice list when next time gets the list.

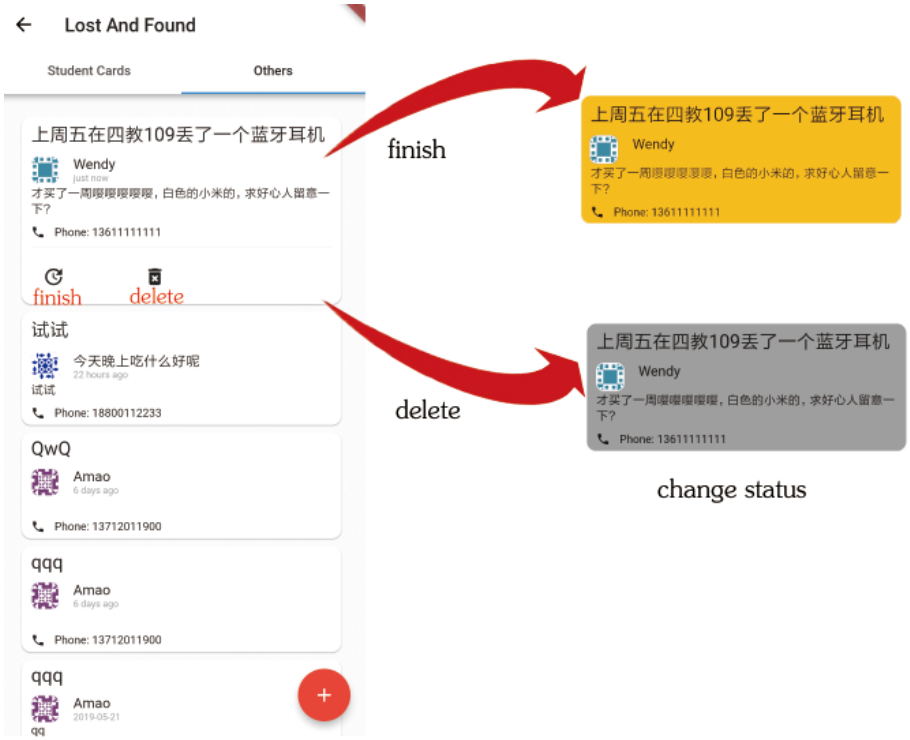


Figure 8: Lost and Found

3.10.3 Implementation

I set lost and found as a stateful widget because whenever it turns page or change status of notices, it needs to reset state. Lost and Found page's body is tab bar view, under Lost and Found, there are two subtypes, "Student Cards" and "Others", they both have individual notice list and they share the same app bar, in this app bar, it has two bottons, it controls which type of Lost and Found can be shown. Other aspects are similar as the moment.

3.10.4 Testing - Page navigation question

- **Description**: When finish editing new notice of Lost and Found Student Card and press publish button, there is a page redirection.
- **Testing Case**: The testing case is creating a new notice and publishing it successfully.
- **Expected Result**: The expected result is page redirect to the Student Card notice list page with the new notice which just created by the user.
- **Realistic Result**: The realistic testing result is it indeed back to that page, but compare with previous notice stream page, it doesn't have an app bar.
- **Causes**: The reason is that app bar is in the scaffold of Lost and Found, but not in any of its subtype's page.
- **Improvements**: Solution is redirect page to Lost and Found page but not Student Card notice stream page. Moreover, there is a value need to transmit between page to page, to tell the index of Lost and Found page which tab is shown now. For example. if from "Other" 's create page, it should transmit data "1" to Lost and Found page, let it show tab with index 1, which is Others notice stream page.

3.10.5 Testing - Contact type question

- **Description**: In every Lost & Found notice card, information includes the poster's contact information. When the user creates a new notice, they need to choose which contact way they would like to leave, then input contact information. The notice will display which kind of contact they chose, "Phone" or "WeChat" or "QQ", then followed with contact information.
- **Testing Case**: The testing case is the user chosen "Registration Phone number" as contact way but input another phone number.
- **Expected Result**: My expectation is user's contact information display correctly.

- **Realistic Result**: It will display information that user input, store it in the database as a registration Phone number. Although it would not bother others to contact this person, it is a wrong data, this problem will affect the consistency of information.
- **Causes**: No controller of input data on restricts of it.
- **Improvements**: The solution is after getting data from the database, do a trinomial operation to judge if the contact way is "Registration Phone number". If it is , when store data as user registration phone number in the global variables list of contact information, therefore whatever user input in contact information text filed, even they did not input anything, as long as the contact way they chose is "Registration Phone number", contact information display in notice will still be what it should be.

3.10.6 Time Management

On the first two weeks, I learned some basic knowledge about PHP and be a member of the back-end, I am responsible for build the user account system, include functions of user login and user registration.

In early April, I moved up to the front-end. Before that, the front-end and flutter were total new things for me, so I spent almost two weeks to understand how flutter works. Then I tride to imitate other's code while creating my own's.
April:

Second week:

- Add a drawer to the homepage. (Redesigned and covered)
- Build a toolbox page. (Redesigned and covered)
- Build a moment page, created some local data, and displayed it in the form of moments.

Third week

- Build the Lost and Found page, include regular and student cards.
- Build Lost and Found information flow pages and creating new notice page.

Fourth week

- Add gesture detector in lost and found to switch pages.
- Successful get and post information between front-end and back-end.
- Building app function browse pages.

May: First week

- Build moment page, include Information flow pages and creating a new moment page.

- Solve some problem of Lost and Found.
- Get moment information from back-end.

Second week

- Post a new moment from creating the page.
- Realize the update function of Lost and Found.
- Add a dynamic routing to the index page, so that other pages can pass a parameter to it, can control a specific page of the index that others page to jump to.
- Add some simple function to the toolbox function, include calendar and phone book.

Third week

- Solve moment page turning problem.
- Realize the update function of Lost and Found.
- Add a dynamic routing to the index page, so that other pages can pass a parameter to it, can control a specific page of the index that others page to jump to.
- Add some simple function to the toolbox page, included calendar and phone book.
- Build comment function of moments, include get and post request.
- Fix text field and keyboard conflict problem.
- Add limited condition jump out dialogs.
- Fix update problem of the moment, can get the last moment after post a new one.
- Get the moment time difference to display.

Fourth week

- Realize moment comment delete function
- Update URL.

Fifth week

- Add a restriction of Lost and Found posting.
- Add Up-Pull Refresh Down-Pull Page Flipping function to moment.

3.10.7 Team Management

In our team, there is a team leader who charges for almost everything, I reporte my process to him every day. If I met some unsolvable problem, I would let him know what happened and together deciding whether this function need modify or delete. He builds an API network for us to test if back-end functionality is correctly implemented, it has provided me with great convenience. Sometimes he gives me a lot of requirements, includes UI design and displaying effect, I always try my best to meet them.

Another front-end member learns flutter and H5 much earlier than I did. He helped me a lot on installing the Android studio and building a flutter environment. In earlier, he also answered some questions about flutter for me in the course of my study. Then I can learn it independently. He is a very good encourager. The division of work between us has been a bit vague, because I cannot predict the results of my front-end learning, if I met too much unsolvable problem, then I need let him take over my work. Fortunately, this did not happen, I finished two main functions successfully. We sometimes reuse each other's code and we help each other to solve issues, it improved the efficiency of identifying key issues.

My team management with back-end is mostly about front-end and back-end interaction. If they didn't tell me in advance for URL changing or back-end was editing while I was testing front-end, I would make a lot of futile efforts. When I implementing like the function of the moment, I asked back-end giving me a list of user names for who had pressed "like" of each moment, it was a beyond expectations requirement to back-end but it got a good result, it was a successful interacting.

In this project, we had a group meeting at least once a week, usually it was two or three times a week, after the team leader made a summary and checked our process, we would set together doing our own work. Sometimes we went through ten hours of collective programming, and Git is an essential tool for teamwork.

It was a good team work experience.

Appendices

A API Reference

- /
  - Articles
    - Post new article
    - Request article content
    - Request article list
    - Get banner image
  - Book Transaction
    - Add new record
    - Check if the transaction finished
    - Request Book Transaction list
    - Request category details
    - Search book
  - Forum
    - Add category
    - Add comment
    - Add post
    - ...
  - Lost Found
    - Add new record
    - Check if the transaction finished
    - ...
  - Moments
    - Add new Moment
    - Add new moment comment
    - Add new record
    - ...
  - News
    - Request news category
    - Request news department
    - Request news details
    - ...
  - User
    - Change password
    - Check ID card
    - Check email
    - ...

For more details, see `https://dev.iecho.cc/doc`.

B Database Design Document

- /
  - articles
  - books_categories
  - books_transaction
  - forums
  - forums_comments
  - items_transaction
  - lost_found
  - lost_student_card
  - moments
  - moments_comments
  - moments_like
  - news
  - tokens
  - users

For more details, see `https://github.com/nonPointer/eBJUT/wiki`.