CAN301 Final Report

Weiqiao Ai, Fei Hong, Zirui Zhou, Yueran Cao, Yetong Wang and Haohsiang Yu
Department of Computer Science and Software Engineering
Xi'An Jiaotong Liverpool University
{Weiqiao Ai, Fei Hong, Zirui Zhou, Yueran Cao, Yetong Wang, Haohsiang Yu}@xjtlu.edu.cn

Abstract

XJTLU Public Map, as the application developed and designed by our group this semester, can help XJTLU students and staff make their daily campus life more convenient. Popular applications currently used by most people, such as Dian Ping and Gaode Map, can help users efficiently obtain information about their surrounding resources. Our application design also adopts such inspiration, that is, it can help Xjtluers quickly obtain the details of surrounding resources, such as supermarkets, restaurants, and printers. These details include a brief description, available time, rating scores, and a list of items offered by each location. Furthermore, users can also choose two browsing methods according to their preferences. One is that users can select the place they want to browse through the resource list provided on the Home page and enter the details page to view it. The other is that users can select the surrounding resources through the Map page in combination with their positioning on the campus and enter the corresponding details page to view. If users encounter problems or want to make suggestions after using our application, they can contact us by clicking the icon in the upper right corner.

1. Introduction

Popular applications currently used by people, such as Dian Ping and Gaode Map, can help them more easily understand and obtain the details of their surrounding resources, such as restaurants and supermarkets. These functions will help users save time and improve the convenience of daily life, especially for users unfamiliar with the surrounding environment, the emergence of these applications can effectively help them adapt to the new life. However, due to the excessive resources stored in these applications, the location errors and the problems that some resources are not displayed on the list have been potentially affecting the experience of some users, especially those who use these applications in non-commercial areas. In many non-centralized consumption areas on the map, due to the impact of commerce, application developers have not updated new resources in time, such as newly opened stores. When XJTLU students and staff use these applications, they will also encounter such problems, which are not friendly to people who have just come to the university to study or work. In order to help Xjtluers have a more convenient campus life, our team decided to develop a similar application, that is, to imitate the functions of these popular applications and narrow the scope to the campus, to ensure that users can enjoy a more convenient service in XJTLU. In addition, our team also provided some unique resources at the university, including printers, vending machines, coffee machines, etc. This design also effectively helps users reduce the waste of time.

There are three design implementations of our application that can arouse the interest of users in XJTLU. First of all, we made a precise positioning for all the resources in the school through the visit records. In addition, we also listed the specific locations of these locations, including the number of floors, and the teaching buildings in the corresponding detail pages. Secondly, we provide users with two ways to browse these places. One is to select and view through the list on the Home page, and the other is to view the surrounding resources through the map provided by the Map page in combination with their location. Furthermore, in both ways, we provide users with the classification function, which will help them quickly select the resources they need. Finally, we carefully designed the application, including icons, animation, and operation logic, which will help users quickly become familiar with the way of use. The diversity of application operation options, the vividness of application design, and the convenience brought to users can attract the interest of Xjtluers.

2. App Design

2.1 The system architecture design of our app

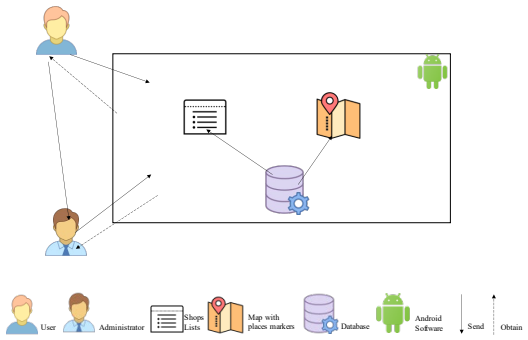

Figure 1. System architecture diagram of our app

As the details shown in Figure 1, the following entities have been defined to participate in the proposed architecture:

- User: The User is considered as the main participant and the final stakeholder of this app who use this particular app to acquire different kinds of information. More

specifically, the users here are set as the target users of the application, that is, students, teachers, and other people who have strong connections related to the XJTLU campus.

- Administrator: This role is generally regarded as the developer of this software who would also maintain the system to ensure that could run successfully. Moreover, this role can also update the system by adding new information and solving problems based on users' feedback.

- Shops List: Shops List is one of the main features in our proposed system which use a list form to provide different shop information related to our campus. Moreover, we design printers and other small features to help target users look up detain information.

- Map with places and markers: The Map is another feature in our system which provides vivid effects for users to find different shops' locations. Moreover, by designing different styles of markers, people would know the classification of different shops.

- Database: The Database is a core feature of our proposed system since the majority of data has been stored in databases. With the help designed function, our app could acquire data from the database and show it in our app which means, if data with the correct form is added to the database, this would be shown in both list and map.

- Android Software: This role is the main platform that our app runs. The overall design of the app is based on the requirement of android and could show the feature of the android system.

- Send and Obtain: Those two arrows shown in above Figure 1 represent interactions among different entities. Detailed content will be shown in the workflow chart.

2.2 The project workflow of our app

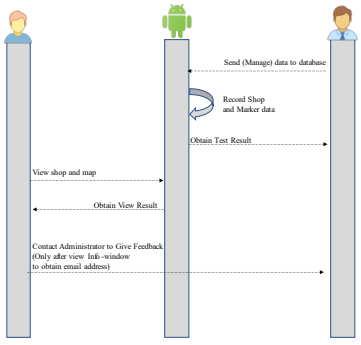

Figure 2. Project workflow diagram of our app

Figure 2 represents the workflow of our proposed apps which focuses on starting from the initial steps. First of all, administrators would use data to manage related databases to provide results in software. Subsequently, the database in our designed Android-based system will hold those various types of data. Moreover, the administrator will only have to open the app to get the test results. As for the user, they can first open the app and decide to view the required information. The software would then provide results based on the user's actions. Finally, there is an alternative feature that allows users to contact the administrator to give feedback or additional information by obtaining an email address from this software.

2.3 Application design of Java Classes

The Java Classes of the application are classified into front-end and back-end, which focus on user interaction and information storage respectively, and two UML images will be illustrated below to show the general Java Class relationship in the front and back end. The front-end is for the Android system in this project. And more details concerned about it will be demonstrated later.

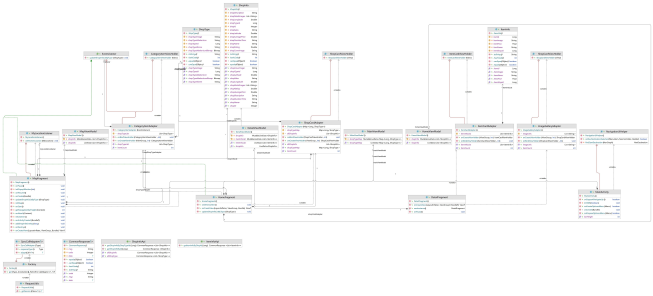

Figure 3. The relationship among each Java Class in the front-end

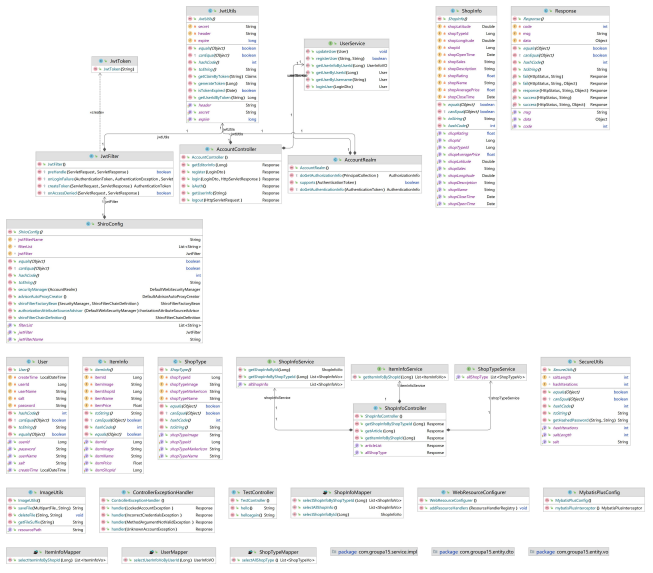

Figure 4. The relationship among each Java Class in the back-end

2.3.1 Front-end design

The Java Classes in this part contribute primarily to achieving the main functionalities in the 'Home' and 'Map'

page, which include shop information classification, shop navigation, and other incidental functionalities.

1) Main activity and main view model

The *MainActivity* class includes the navigation bar and permission check SDK initializer with related animation effects and system management lifecycle.

The *MainViewModel* is a class that monitors and updates data in the application, though this class is utilized mainly to store the corresponding data in it.

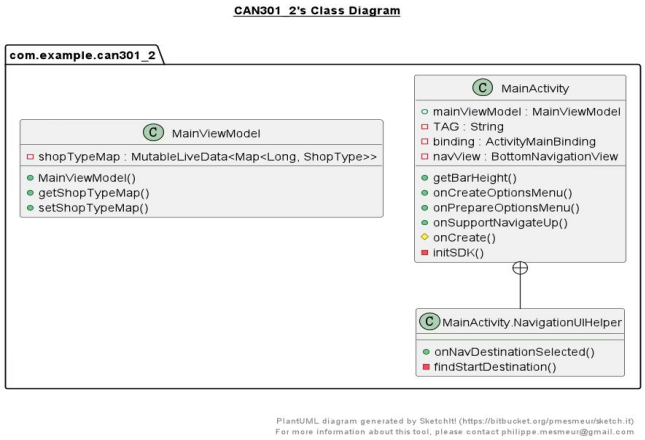

Figure 5. The class diagram of the Main

2) API

The *API* Class in our application aims primarily at acquiring the data from the interface of the back end remotely.

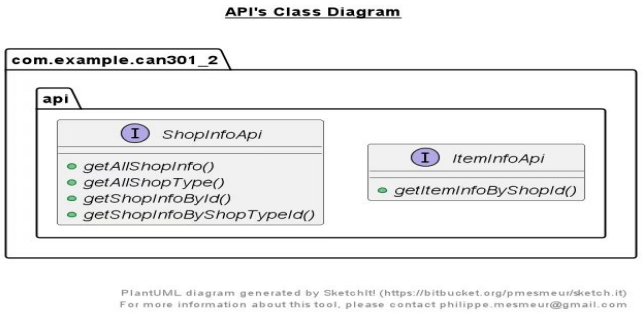

Figure 6. The class diagram of API

3) Adapter

The *Adapter* Class is established to adapt the position of the lists and items in it in the 'Home', 'Detail', and 'Map' pages.

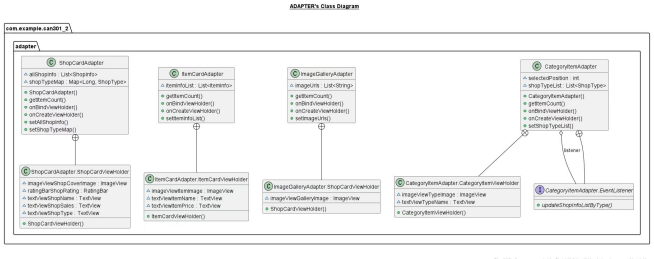

Figure 7. The class diagram of the Adapter

4) Domain

The *Domain* Class primarily focuses on resolving the data in JSON format from the interface of the back-end and declaring the type of data.

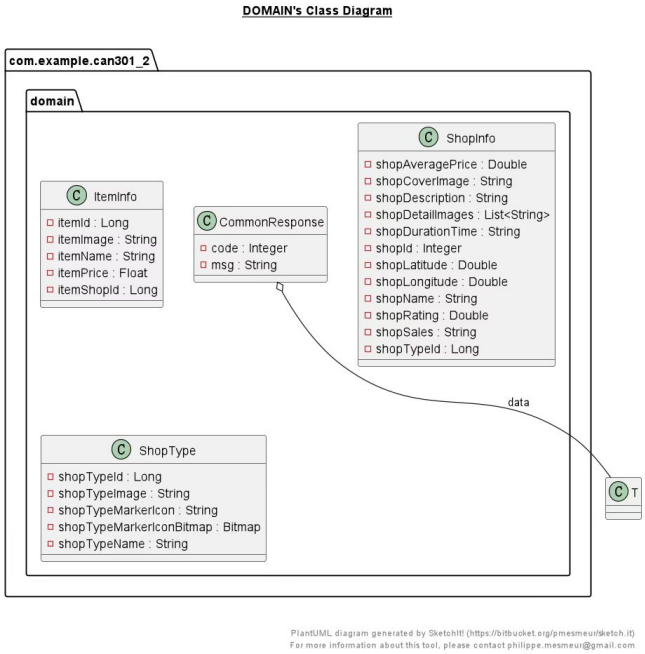

Figure 8. The class diagram of Domain

5) UI

The *Home* Class is created to achieve our main functionality in the 'Home' page, which is supposed to exhibit relevant and plain information about stores or other facilities.

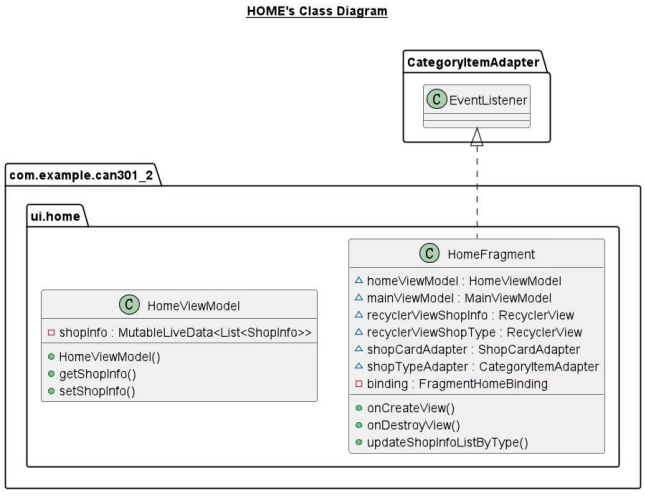

Figure 9. The class diagram of Home

Moreover, the *Detail*'s Class serves at providing customers with more information about the stores or facilities, such as their specific location, opening hours, and some specific goods on sale.

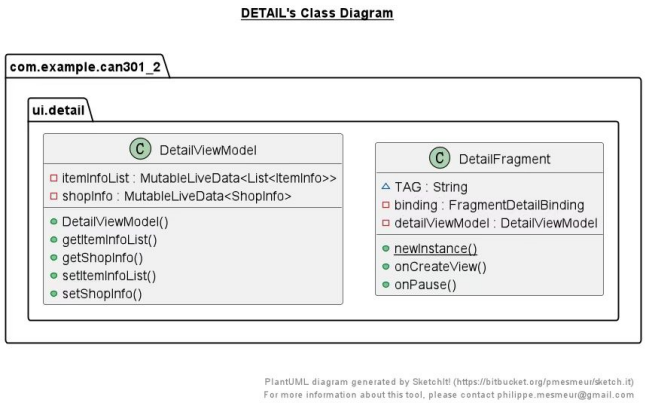

Figure 10. The class diagram of Detail

And the other main functionality in the application is the navigation system, which is served and achieved by the *ui.map* Java Class, which includes the classification and localization of each store or facility on the 'Home' page. Moreover, it could sign the stores and facilities around our school on the 'Map' page with specialized icons designed by our team members. The classification function can show or hide the icons based on the users' requirements.

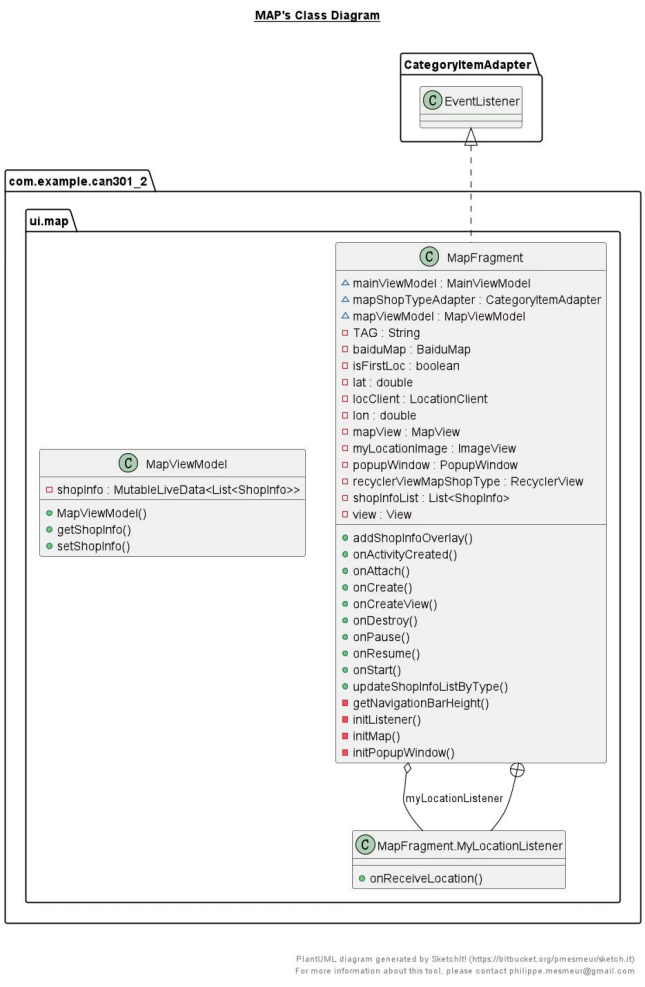

Figure 11. The class diagram of Map

6) Utils

The *Utils* Class containing two sub-classes called *'SyncCallAdapter'* and *'RequestUtils'* has been established to provide a standard interface to synchronize requests in the JSON format, which could be executed after declaring and building the specific service by adding a customized adapter and a Gson convertor.

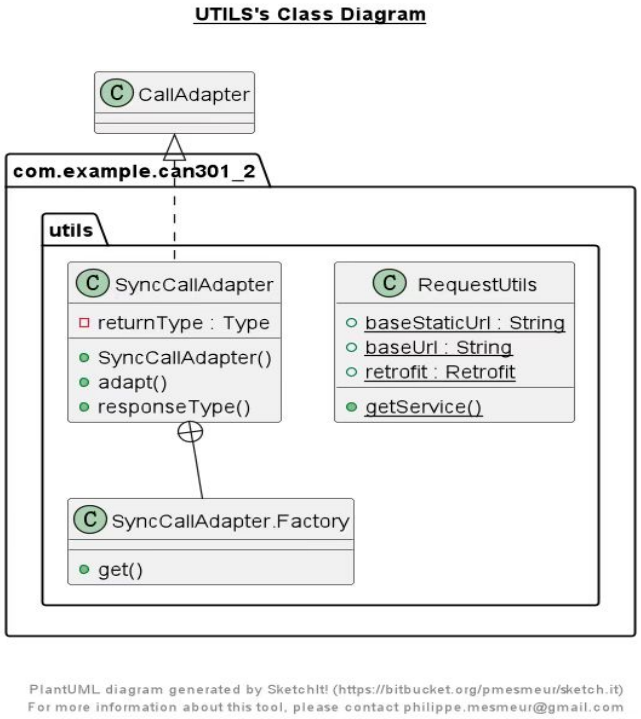

Figure 12. The class diagram of Utils

2.3.2 Back-end design

The back-end design in our application is divided into *Database* and *Controller* two modules. The Java Class in the *Database* module serves at storing corresponding data concerned about the shops and printers. And the Java Class in the *Controller* module focuses on transferring the data to the front-end. Specifically, the *shop_detail_image* and *shop_type* two classes aim at storing the specific information about the shops, like name, image, type, location, and other information, and the *item_info* is established to store specific information about the items on sale, like their name, price, and image.

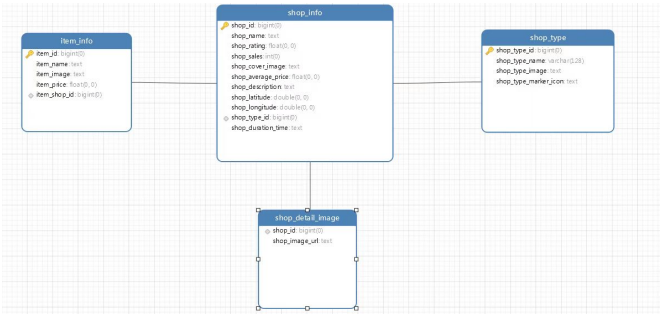

Figure 13. The relationship of used database in Back-end

2.4 Images selection of our app

Our app is a campus assistant-type map app developed for XJTLU students and staff, which integrates campus-related stores, supermarkets, printers, vending machines, and drinking places around the campus. There are three main types of graphic materials used in this app. Among them, these image resources are obtained by querying the appropriate icon on the image website ICONS8 [1] and applying them to our application design.

The first category is web material, such as pixel icons, which are listed on top of the whole app. These materials represent the labels of different types of stores, thus helping users to find them more easily.

Figure 14. Example of icon images

The second category is live images, which were collected by our team members themselves, such as stores on campus, printers, and supermarkets, which are mainly used as displays in the store list.

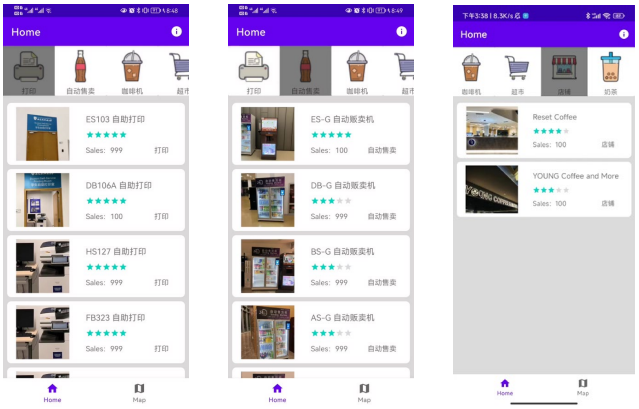

Figure 15. Example of shop images

The third type of image is the product descriptions from real stores, which we downloaded and used as products in our app, restoring the contents of the stores to make it easier for users to understand them.

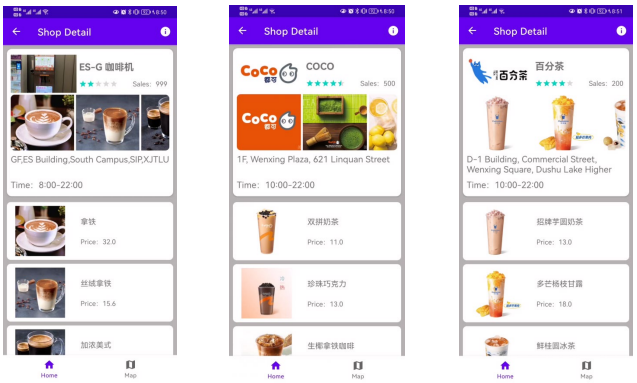

Figure 16. Example of product images

2.5 Other android resources

1) Baidu Android Map SDK

The Map is an important function in our proposed app. Moreover, it is found that if we want to use location information in our app, various Map SDKs could be used to tackle related problems since it is considered only a legal way to show the real map and use location data in China. Therefore, Baidu Android Map SDK has been imported into our system to show shop locations vividly and interact with the map based on locate and click functions with the help of related API documents [2].

Figure 17. The Map page of our app

2) Backend interface

The network interface in the Android project is implemented mainly depended by Retrofit, Gson, and Lombok, which are for the requesting, parsing, and class building separately. *"RequestUtils"* and *"SyncCallAdapter"* are constructed to provide a standard interface for common synchronized requests in the format of JSON, by adding a customed adapter and a Gson convertor, referring to treekt in the StackOverFlow [3]. The APIs and relevant data domains can be simply defined in the package "api" and "domain" by the given structures. These requests can be executed after declaring and building one specific service.

2.6 Personal contribution details of our project

The following table shows the contribution items of each member of our group to the project:

Name	Contribution items	Percentage
Weiqiao Ai	1) Front-end work about *home*, *detail* in *UI*, and *Adapter*; 2) Provide project ideas about the detailed page design of each shop; 3) Participate in writing about App design in the group report.	16.0%
Fei Hong	1) Back-end work about the *Controller*; 2) Provide project ideas about the content of the detail page design of each shop; 3) Participate in writing about App design in the group report.	16.0%
Zirui Zhou	1) Back-end work on the *Database*; 2) Provide project ideas about the application interface design; 3) Fixed most of the front-end and back-end bugs; 4) Participate in writing about App design and evaluation in the group report.	19.0%
Yueran Cao	1) Front-end work about *Utils*, *API*, and *Domain*; 2) Provide the project ideas about the classification function used on the Home page; 3) Collect image materials from ICONS8 and real campus; 4) Complete the writing and revision work of the group report. Complete the video editing work of group presentation.	16.0%
Yetong Wang	1) Front-end work about the *map* in *UI*, *MainActivity*, and *MainViewModel*; 2) Provide project ideas about the use of the Map function; 3) Fixed the bug about the Map function; 4) Participate in writing about App design and evaluation in the group report.	16.0%
Haohsiang Yu	1) Front-end work about the *map* in *UI*, *MainActivity*, and *MainViewModel*; 2) Provide project ideas about the classification function used in the Map page; 3) Fixed the bug about the Map function; 4) Participate in writing about evaluation in the group report.	17.0%

Table 1. The personal contribution table of our project

3. Evaluation

3.1 Problem with map positioning

Show the current location and university map have been considered as one of the major problems in app development based on the aim and usage of this app. More specifically, it is first found that after the location listener set and records the current location data the current location map turns to Africa with both longitude and latitude returning 0. After looking API document, it first gives the possible solutions that Map SDK should be initialized in the *onCreate()* function and give related dynamic permissions to request acquire location, Internet, Read Phone State, and so on for android versions over 6.0 [4]. Therefore, a request permission function has been designed and the problem has been tackled. Regarding the limitation of this solution, it is found that we only assume users would open GPS and location permission while they using this app. If they manually closed or refused to give permission, users could

not use map functions. Thus, Intent functions or detail notifications could be designed to handle this problem in the future. The second problem we encountered is that after clicking added location button, the current location would have the same problem as above. After a detailed look at Baidu Map SDK API document, it shows that the API key we applied has problems since we should both generate debug and release the key with the proper package name. Therefore, we reapplied and the details are shown in Figure 18. After that, it is found that we could use the location service and location button to go back current position after drag maps to view other places. However, the specific location is wrong. For example, we are currently on the XJTLU campus but the position on the map shows we are at Suzhou University. After detailed debugs code, a stupid mistake has been made since the default coordinates form of Baidu Map is BD09LL, we write it into BD0911 which looks the same in the Android Studio since the uppercase letter is not strictly required for writing. After modifying the above problems, map functions could be used and additional functions could be added. In terms of limitations related to this solution, it is found that the applied SDK only successfully runs based on the package name shown in Figure 18, which means we could not change app names. If app names changed, map functions could not be used. If we want to change app names, we should reapply a key and it may cost a long time to debug potential problems.

Figure 18. Baidu Development Platform Command Board

3.2 Problem with content design on the Map page

In the map fragment, one of the major challenges apart from the location function is how to add various map objects to the map and display them according to different categories.

First of all, to implement the default interface for shop icon display, we use the marker to display the icon on the layer. When initializing the *overlayOption*, the latitude and longitude in *shopInfo* and the icon style corresponding to the *shopType* are called. Moreover, to solve the problem that the marker itself cannot carry data, each of them uses a bundle to store its shop id and calls the *shopInfo* with the corresponding id when the popup window is displayed.

Subsequently, we are going to place the popup window right above the navigation bar and add a popup animation. To accomplish these requirements, we get the height of the popup window itself and get the height of the *NavigationBar* in *MainActivity* and use them as offset values in *showAtLocation*. In addition, we use *TranslateAnimation* to add the animation of the popup window.

Finally, to implement the classification function in the map fragment, we added a *recyclerView* to the top of the fragment and added *shopType* to it. When the item on the *recyclerView* is clicked, all icons on the map will be cleared and shops of all clicked types will be added. If the same type is clicked twice, then the classification will be cancelled and all icons will be displayed. In order to complete this click logic, we add a parameter called *typeId* in *AddShopInfoOverlay*. Because the type ids passed into *shopType* are all positive, type id "-1" is used in the default display condition.

3.3 Problem with UI thread and background thread

Considering the abundant network requests to backend APIs, the relevant requests should be insulated from the main UI thread and run in a new thread. Since an exception of *"android.os.NetworkOnMainThreadException"* will be thrown when launching a request in the fragments, it is not reasonable to bypass this exception, such as awaiting an async task, because it would freeze the UI and cause noticeable interface lag. In the early version, our team applies a strategy by utilizing the *"AsyncTask"* to execute the requests to update and maintain the relative *"ViewModel"* for data. However, this type of class is redundant and inflexible in parameter passing, task building, and execution sequence, not to mention that this class is deprecated in the latest Android development. According to Kashfa Khan in StackOverflow, *"Executors.newSingleThreadExecutor()"* and *"Handler(Looper.getMainLooper())"* are introduced to supersede the deprecated Android Async API [5]. To be specific, this code is closer to the functional programming and flexibly defined among the contexts, instead of inheriting an *"AsyncTask"* class. Furthermore, it can clearly distinguish the order of execution of codes at the UI level or background level. For example, the views can be rendered after the request and assignment of the required data, which is depicted in the Figure 20.

```
ExecutorService executor = Executors.newSingleThreadExecutor();
Handler handler = new Handler(Looper.getMainLooper());

executor.execute(() -> {
    //Background work here
    handler.post(() -> {
        //UI Thread work here
    });
});
```

Figure 19: The basic framework of the async task realized by the executor and handler.

```
ExecutorService executor = Executors.newSingleThreadExecutor();
Handler handler = new Handler(Looper.getMainLooper());

executor.execute(() -> {
    // Get the shop information by given shop ID.
    ShopInfoApi shopInfoService = RequestUtils.getService(ShopInfoApi.class);
    ShopInfo shopInfo = shopInfoService.getShopInfoById(shopId).getData();

    // Get the list of item information by given shop ID.
    ItemInfoApi itemInfoService = RequestUtils.getService(ItemInfoApi.class);
    List<ItemInfo> itemInfoList = itemInfoService.getItemInfoByShopId(shopId).getData();

    handler.post(() -> {
        // Add the shop cover image into the imageView.
        Glide.with(getContext()) RequestManager
                .load( string: RequestUtils.baseStaticUrl + shopInfo.getShopCoverImage()) RequestBuilder<Drawable>
                .into(imageViewShopCover);
        // Add information into relevant views.
        textViewShopName.setText(shopInfo.getShopName());
        ratingBarShopRating.setRating(shopInfo.getShopRating().floatValue());
        textViewShopSales.setText("Sales: {shopInfo.getShopSales()}");
        textViewShopDescription.setText(shopInfo.getShopDescription());
        textViewShopTime.setText("Time: {shopInfo.getShopDurationTime()}");
        // Fill the recycle list of detail images and items.
        imageGalleryAdapter.setImageUrls(shopInfo.getShopDetailImages());
        imageGalleryAdapter.notifyDataSetChanged();
        itemCardAdapter.setItemInfoList(itemInfoList);
        itemCardAdapter.notifyDataSetChanged();
    });
});
```

Figure 20: An example of the utilization of such a solution in the shop detail page.

3.4 Problem with user interface concern

When designing the classification function of the Home page, our team has two different design ideas. The first idea is to create a classification interface similar to Dian Ping so that users can click different categories to enter different kinds of list pages. Another idea is that users can select their desired categories by sliding the horizontal slider above the Home page, and the system will feed back the results to the bottom of the slider. For the evaluation of the first idea, its advantage is that users can directly see the classification of all resources on the campus after entering the Home page of the application, which facilitates them to quickly understand the richness of resource types on the campus. The disadvantage is that users must click the corresponding category before the corresponding content list appears, this operation is complex. For the evaluation of the second idea, its advantage is that users can directly see the specific content of the resource under the slider bar, which greatly reduces the complexity of the operation. The disadvantage is that users cannot directly view all the resource classifications on the campus, and they may ignore and forget something they want through the sliding operation. After the group discussion, we combined the user experience. To make the application more user-friendly, we chose the second idea. Because it improves the user's efficiency and makes the content more intuitive. In addition, we also combined the analysis of the target users of the application, and we concluded that the target users of the application have a certain understanding of the university campus, so the impact of the limitations of the second idea can be ignored. The design of the second idea is shown in the figure below.

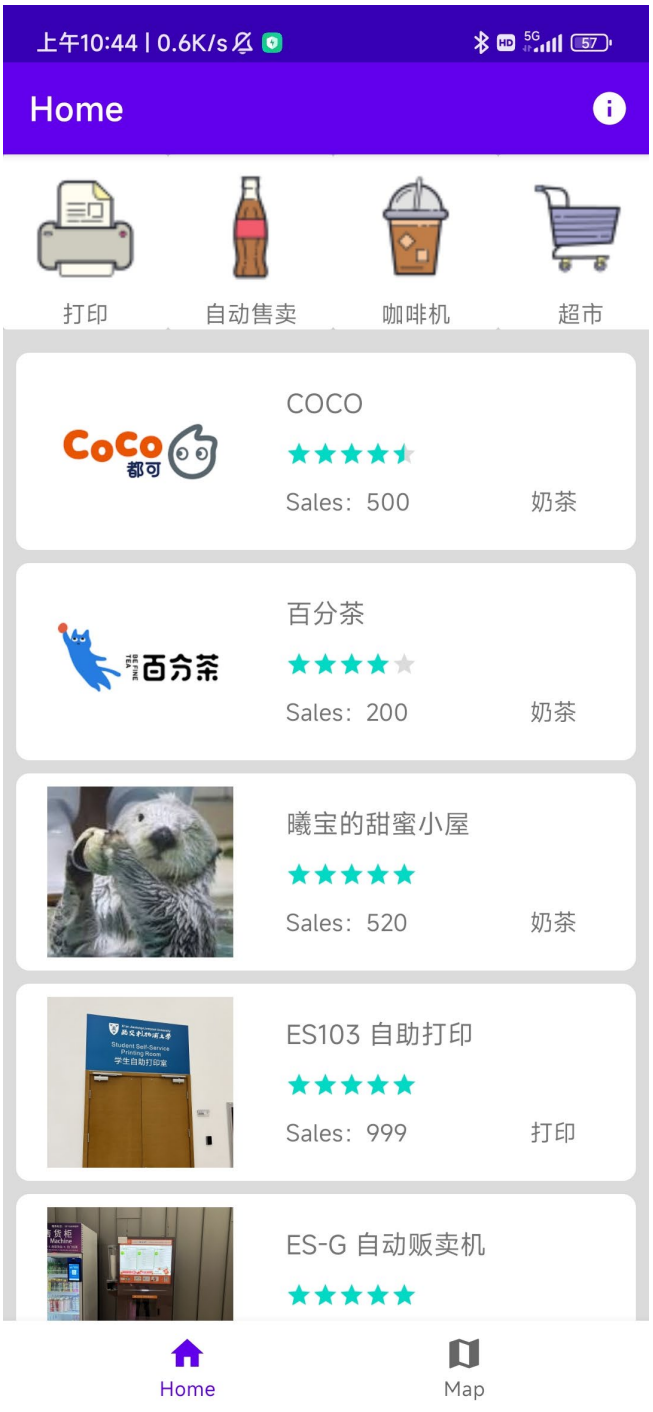

Figure 21. The design of the second idea

4. Conclusion

In conclusion, our group designed and developed an application called XJTLU Public Map, which can effectively help users in school to conveniently obtain resources in

school. Our application not only lists all available resources within the campus, such as supermarkets and printers, but also ensures the accuracy of the location of each provided resource. In addition, the diversity of application operation options, the vividness of application design, and the convenience brought to users can attract users' interests. In the future development work, we will continue to enrich and improve the design of applications, including the development of more interactive functions and the correction of existing bugs, such as the black screen problem caused by the import of Baidu Android Map SDK and the application flashback problem caused by turning off location or network permission. Furthermore, we will try to let users participate more in the application and further improve the user experience by providing users with functions such as ratings and comments. At the same time, we will also continue to collect improvement suggestions from users in the mailbox and specify corresponding measures. For the actual use scope of the application, we will try to expand the functions around the university, such as the dormitory area and business area around XJTLU, so that users can get more convenience.

REFERENCES

[1] ICONS8. Icons, illustrations, photos, music, and design tools. Accessed: Nov. 2022. [Online]. Available: https://icons8.com/

[2] Baidu. Android Map SDK. Accessed: Nov. 2022. [Online]. Available: https://lbsyun.baidu.com/index.php?title=androidsdk

[3] How to make request without Call object. Stack Overflow. Accessed: Nov. 2022. [Online]. Available: java - Retrofit 2 - How to make request without Call object - Stack Overflow

[4] Android 6.0 Changes. Accessed: Nov. 2022. [Online]. Available: https://developer.android.com/about/versions/marshmallow/android-6.0-changes

[5] The AsyncTask API is deprecated in Android 11. What are the alternatives? Stack Overflow. Accessed: Nov. 2022. [Online]. Available: java - The AsyncTask API is deprecated in Android 11. What are the alternatives? - Stack Overflow