Final Report Of CAN301 Group Project: Project OneClick

Fangze Qiu, Bocheng Zhang, Nuo Chen, Tianyang Xie, Huaxiao Huang and Lincheng Shi
Department of Computer Science and Software Engineering
Xi'An Jiaotong Liverpool University
Email: FangzeQiu19, BochengZhang19, NuoChen19, TianyangXie19, HuaxiaoHuang19 and LinchengShi19@student.xjtlu.edu.cn

***Abstract*—Growth in the number and category of modern mobile applications has called for an efficient way to provide the user with quick and direct access to desire apps within hundreds of other apps with similar appearance and names. This report introduces the idea and implementation of a hub app: OneClick, which allows users to enhance their app-using experience through customizable functions. OneClick offers a one-step solution for making phone-call, browsing URLs and opening other apps rapidly with a customized invoking method and a floating view displaying over other apps for immediate access. The report demonstrates the design process of the code and UI of the app and presents the result of its evaluation and redesign.**

I. Introduction

A. Ideas and Inspiration

Dramatic increment in the functionality, category and number of mobile applications has caused great difficulties for users to organize, distinguish and access applications with various purposes [1]. Until 2014, it is reported that over 675,000 apps have been uploaded to Google play while every android user deals with an average number of 35 different apps on his or her cellphone [2]. Additionally, different Pandemic Control Regulations have caused citizens to locate, open and switch between multiple apps to verify their health condition. Motivated by the idea to create a user-friendly cross-app environment, **we aim to design an app implemented with practical approaches and functions that save time and energy for users who frequently engage in multiple-applications contexts.**

B. Related Works

Ideas of facilitating multi-app user experience are adequate, and many have offered inspiration and solid theoretical support for our own development. In [2], an app called *AppRush* was proposed as an adaptive solution for predicting user behaviour and providing shortcuts to possible apps that the user might prefer. While *AppRush* achieves a good prediction rate in tests, its system does not support customization. It may not perform well since users' wishes to access applications can be abrupt and subjective. Another paradigm is the *ShortCut* app from Apple Inc., which is capable of achieving tasks with high complexity [3]. While *ShortCut* is powerful and mature, its learning cost is relatively high since its operating style is similar to a programming language and has complex running logic.

C. Our Solution: OneClick

In this report, we present OneClick on the Android platform as a lightweight solution for quickly accessing normal actions including dialling, browsing and opening applications. OneClick works not as an individual package with isolated and complete functions, but as a Hub-like application that facilitates daily actions performed by users by connecting the pool of functions and apps together.

II. App Demonstration

To help explain the functionalities and coding of the app, a general demonstration of the application will be presented first. The main function of OneClick is direct and simple: The user can create shortcuts toward the desired action with the triggering method they have chosen and manage their shortcut library on the main page. As the central station for the app, the main page is organized as shown in Figure 1.

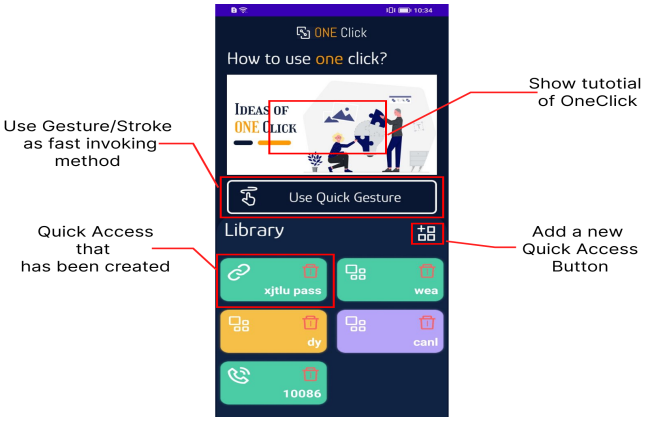

Fig. 1. Design and Function of Main page

The whole user flow can be divided into two phases: creating and triggering, where creating refers to the process of generating quick access (We name one access as a **pair** because it links the invoker and the action as a pair) and triggering indicate the process of using the pair through selectable methods.

A. *Creating*

There are in total four stages to create a pair (See Figure 2):

Fig. 2. Four Stages of Creating a pair

In stage 1, one can choose whether to use a stroke gesture with a maximum count of nine dots as an alternative way to invoke a certain action selected in stage 2. The option includes dialling, browsing or opening apps, and the user has to provide the necessary information (the phone number, the URL or one specific app to be opened with) to initiate the jump successfully. In the last stage, the pair will have to be named in order to finish the construction.

B. *Triggering*

After being created, a pair will appear on the main page in the format of a Button (Namely a Pair Button, one of the unit components of OneClick). Users can initiate the button by clicking, or inputting the quick gesture that has been registered in the phase of creating (See Figure 3).

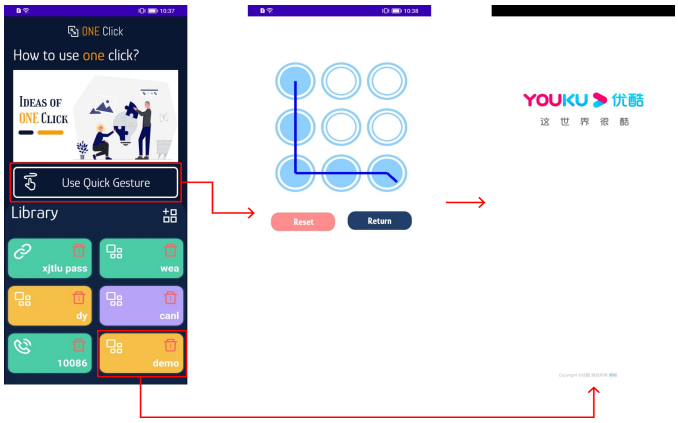

Fig. 3. How to Invoke a Pair

C. *Locating OneClick*

To free users from the dilemma of identifying between different applications when using OneClick, the app will maintain float after focus is lost (See Figure 4).

Fig. 4. Floating Window over Other Apps

To protect privacy and follow the regulation of Android application development, users will be informed that the app is running in the background through notifications and can shut OneClick down at any time. No unauthorised background behaviour will be made.

III. App Design

Ideas of OneClick were developed according to three major requests:

1. OneClick can be easily accessed without the process of locating it on the main screen.
2. OneClick works as a hub connecting other apps and some major functions of the system. It will generate short cut and visible interfaces for users to quickly jump from

OneClick to another desired systematic destination.
3. OneClick provides an effective method for users to manage frequently used apps or activities.

The sub-systems and functions of OneClick are all designed under these instructive ideas. The illustration of OneClick's design will follow the order below: First the conceptual model, system structure and the working rationale of the app, then its UI design based on cognitive science and UX principles. The usage of external sources and our adaptation will be listed at the end of Section III, along with the contribution table of our group members.

A. Conceptual Model

The core concept of OneClick is to build and sustain a database system, allowing users to dynamically add, delete, update and use customized shortcuts. Consequently, the design of the storage carrier of shortcuts (Or the items of the database) is the most significant task. To resonate with the three general design ideas proposed previously, each carrier must have a unique identifier selectively assigned by the user to indicate its importance.

The solution is achieved by the design of PairButton and corresponding sub-functions. As a unit class used to store shortcut information, it supports dynamic modifications on its universal properties, real-time invoking using in-class function and the assignment of unique identifier at creating time: a quick gesture, to demonstrate its priority (See Figure 5).

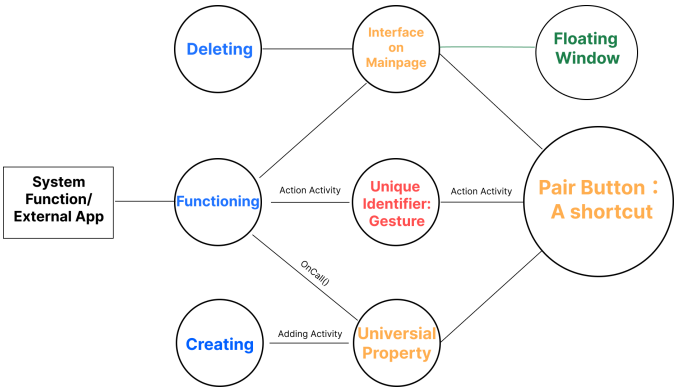

Fig. 5. Conceptual design of PairButton (Credit: Fangze Qiu)

UCD is provided as a structured model to explain the functionality as OneClick is an Object-Oriented system (See Figure 6). In the view of users, the different expansions of the system within five major function categories assist the management and use of shortcuts information and ensure that their privacy and right to know are not violated. Services like calling and background operations are notified to the user through the permission-enabling process.

B. System Structure

The project is divided into seven packages, based on the features and functionalities of the code (See Figure 7).

Fig. 6. UCD of OneClick(Credit: Lincheng Shi)

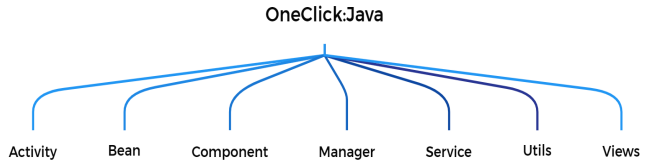

Fig. 7. Basic Structure of system architecture

1) Activity: Activity Package contains the corresponding code for 5 main pages in OneClick (See Figure 8), which support the main functions of the app:

Action Activity: Page for drawing a gesture to trigger the next move.

Adding Activity: Series of pages used for creating quick access.

Guide&Introduction Activity: Two pages that demonstrate the basic knowledge and tutorial of the app.

Main Activity: landing page of OneClick, providing users with all major functions.

Fig. 8. Activity Package

2) Component: Components in OneClick are unit classes that are used multiple times across different pages (See Figure 9):

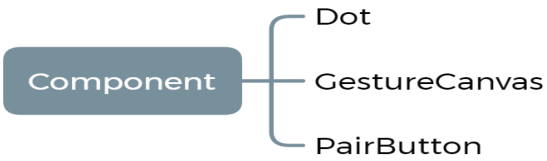

Fig. 9. Component

Gesture Canvas: A series of nine points arranged in order to receive input stroke gestures from the user and manage necessary functions regulating the behaviour of a gesture canvas, such as resetting points.
Dot: Refers to each dot in a GestureCanvas.
Pair Button: The class of pair that contains the basic information of quick access. Including the name, type, corresponding gesture, the path to invoke action in systematic language and randomly generated appearance data of the button that appeared on the main page.

3) Manager: Managers are non-display classes related to the background behaviour of the app (See Figure 10).

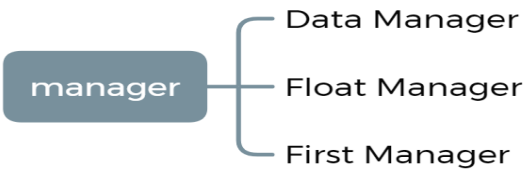

Fig. 10. Manager

Data Manager: A class used to normalize, localize and extract persistent data in JSON format.
Float Manager: Regulate floating window behavior.
First Manager: Extract local data to judge whether this is the first time the user has opened the app.

4) Service, Util, Bean and View: Other parts of the code helping the operation of the app include Service, Util, Bean and View (See Figure 11).

Fig. 11. Other packages

App Bean: A class with functions to store and use the information of external applications for displaying and jumping.
ScreenSize Utils: Contains Tools function to capture the size of the screen of the current device.
BansidingViewPage: An adapted ViewPager that banned users from sliding themselves. Used to contain the three steps when creating a new quick access.
ForegoundServier: Class to keep the app alive while not being focused by the user.

C. Working Ratinale

The overall working rationale can be concluded as the storing, updating, and invoking of pairs (See Figure 12).

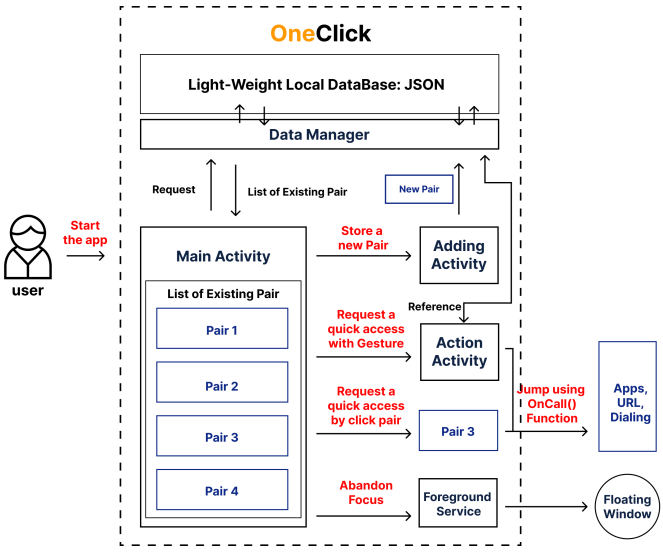

Fig. 12. Working Logic

When initialized or resumed, Main Activity will request a data synchronization of the existing list of pairs to ensure that

no newly-added pair will be omitted. Each pair will contain an internal *OnCall()* function and the necessary information to initiate the jump when clicked. If the user chooses to input a gesture as an invoker, functions in Action Activity will refer the inputted gesture to the stored gestures obtained by Data Manager to see if there is a match, then start the jump if a match is found. Once the focus is lost, the foreground service will be automatically started to create a floating window over other apps for convenient usage.

The simplified UML graph (See Figure 13) demonstrates the general supporting and invoking relationships between major packages. The detailed version of UML is attached in the appendix.

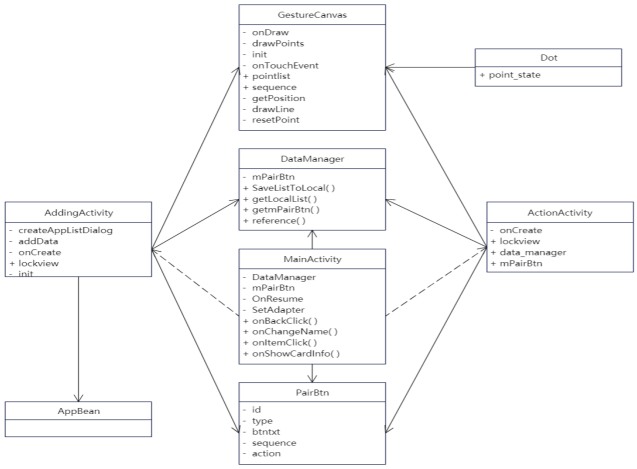

Fig. 13. Basic Structure of system architecture in UML (Credit: Lincheng Shi Detailed version is in appendix)

D. Images & UI Components

The UI design of the app has iterated through two versions. We adopted the dark-theme style with simplicity and aimed to provide a smooth experience with proper affordance and visual effects.

For the initial UI version, we use a light-theme design and an energetic color palette (See Appendix A, Figure 36). It generates a clear view but unpolished feelings.

In the final version, we put more frequently accessed buttons in more visible places and adopt a dark theme with lighting effects to create a sense of vitality and refinement (See Figure 14).

External sources of UI include some open-source icons and an illustration. Downloaded from plugin iconify [4] and flexiple [5] (See Figure 15)

E. Cognitive Design: Use of Gesture

We are using stroke gestures as a shortcut because of their cognitive benefits. As a visual representation that combines both the visual system and motion system, gestures can reduce the cognitive load caused to the user and raise better attention to avoid errors [6]. Alternatively, compared to normal access like buttons and card which has duplicated shape, color and

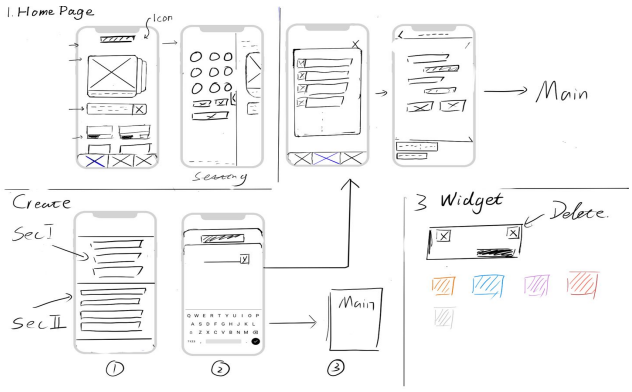

Fig. 14. UI design of the final Version: Wireframe and prototype

Fig. 15. UI External Sources

pattern, gesture has a higher level of uniqueness [7] thus creating a stronger mapping relation towards different actions.

F. External Coding Resources

Two independent functions were built based on existing code (See Figure 16).

Code	Adoption Level	Source
ScreenSize Utils	95%	Mars-xq. Getting the size of screen of android device. [Online].Available: http://t.csdn.cn/eOist
Gesture Canvas	30%	https://github.com/ihsg/PatternLocker/blob/master/

Fig. 16. Code External Sources

The ScreenSize Utils [8] is fetched from existing work. It is used to obtain the size of the screen. Instead of using the original function, it is extracted from its source code and put as an independent class for repetitive usage.

The GestureCanvas was partially built upon the works of others. We modified its operation logic to adapt it from an unlocking method to an individual component that outputs gesture sequences. We also rewrite its reset() function, OnDrawFinish() function and other detailed implementations to better cope with our needs. The drawing logic and method for judging the position of the points remain unchanged.

G. Contribution of Member

The contribution of each member is displayed in the screenshot below. Please be noted that the table does not indicate the difficulty of the coding work and the important classes are marked in red.

Name	Fangze Qiu	Tianyang Xie	Nuo Chen	Bocheng Zhang	Lincheng Shi	Huaxiao Huang
ID	1929003	1929930	1929999	1929892	1927978	1927142
Report Writing						
UI Design						
UI Implementation(.xml)						
Action Activity.java						
Adding Activity.java						
Guide & Introduction Activity.java						
App Bean.java						
Dot.java						
GestureCanvas.java						
PairBtn.java						
DataManager.java						
FirstManager.java						
FloatManager.java						
ForegroundService.java						
ScreenSizeUtils.java						
BanslidingViewPage.java						
MainActivity.java						
Debug&Testing						
Percentage	19%	17%	16%	16%	16%	16%

Fig. 17. Screenshot of Contribution Table

IV. Evaluation

In this section, the analysis and solution for technical difficulties encountered in the project will be discussed, and the result of the usability test will also be presented. Additionally, the comparison between OneClick and one similar product: "Short Cut" from Apple Inc. will be made to illustrate the usage and practical meaning of our application.

A. Technical Challenges and Problem Solving

There are a total of three technical challenges in terms of the development: First, how to efficiently register the necessary information for creating and invoking a shortcut; Second, when an increasing number of shortcuts are added to the app, how to equip the user with a tool that helps to locate the right shortcut with ease; Third, OneClick is also an app that lay on the main screen, how to make it more visible compared to regular apps so that the user can spot and access OneClick effortlessly.

For the first obstacle, our solution is to create a PairButton instance containing all the required information as variables (See Figure 18) and install the function for calling, visiting websites and opening the desired app inside a single PairButton class. And use Data Manager as a macro approach for storing & managing clusters of pair buttons.

The *btntype* variable will indicate the behaviour of the shortcut after invoking. When a PairButton is initiated from outside its class using *OnCall()* function (See Figure 19), the system will decide the action taken based on the *btntype*.

```
public class PairBtn implements Serializable {
    public int id;
    public int type;
    public int btntype;
    public int backbtn;
    public String btntxt;
    public int del;
    public List<Integer> sequence;
    public String action;
```

Fig. 18. Variables of PairButton

```
public void onCall(Context context, Activity act){
    if(btntype==0){
        String url=this.action;
        if (url.startsWith("www."))
            url = "http://" + url;
        if (!url.startsWith("http://") && !url.startsWith("https://")) {
            Toast.makeText(context, text: "Wrong URL", Toast.LENGTH_SHORT).show();
        }
        Uri uri=Uri.parse(url);
        Intent intent=new Intent(Intent.ACTION_VIEW,uri);
        context.startActivity(intent);
    }else if(btntype==1){
        String pkname=this.action;
        Intent intent=new Intent();
        PackageManager packageManager=context.getPackageManager();
        intent=packageManager.getLaunchIntentForPackage(pkname);
        context.startActivity(intent);
    }else if(btntype==2){
        String phonenum=this.action;
        if (ActivityCompat.checkSelfPermission(context, Manifest.permission.CALL_PHONE) !=
                PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(act, new String[]{
                    Manifest.permission.CALL_PHONE}, requestCode: 1);
        } else {
            Intent intent = new Intent(Intent.ACTION_CALL);
            Uri data = Uri.parse("tel:" + phonenum);
            intent.setData(data);
            context.startActivity(intent);
        }
    }else{
        Intent intent=new Intent(context, ActionActivity.class);
        context.startActivity(intent);
    }
```

Fig. 19. Logic of OnCall()

The solution for distinguishing important pairs from less stressed ones is the implementation of QuickGesture functions, supported by GestureCanvas along with other classes. The reason for using gestures but no other method is explained in Section 3.4.

In order to grant OneClick a quick and exclusive accessing experience compare to other applications, we have designed and tested several approaches. At first, we designed a background listener that records and react to a special volume button event. When the up and down volume buttons are pushed at the same time, the listener will invoke OneClick. However, this method performed terribly since the volume buttons have multiple other functions on a systematic level like taking screenshots or making adjustments to volume. While pushing both buttons simultaneously does not trigger and specific event, it is hard to define the boundary between adjusting volume quickly using both buttons and invoking the app. Later, we decided to use a small size floating window as a port for entering OneClick. The app will be kept alive when losing focus through sending regular notifications, which

is the method approved by the official document of Android development published by google.

B. Usability Test Result

In the usability test, we first evaluated the performance of two versions of UI and the general capability of the application through a questionnaire with 14 participants.

The result indicates that the UI of Version II has generally better performance, in terms of its visual appearance and usability (See Figure 20, Figure 21).

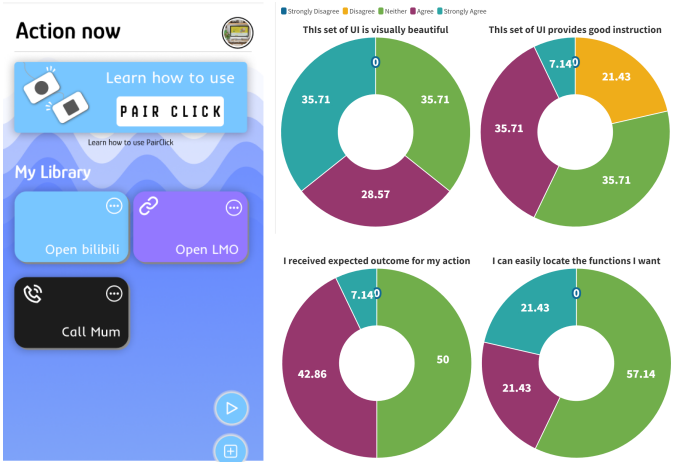

Fig. 20. Evaluation Result of UI Version I

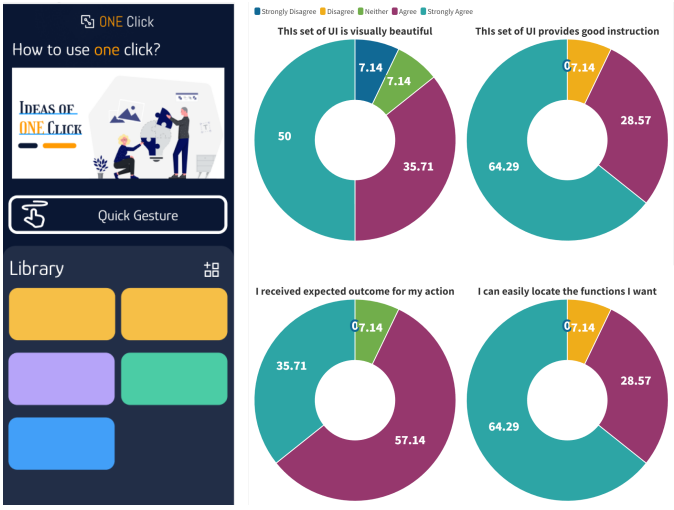

Fig. 21. Evaluation Result of UI Version II

As shown in the visualization, while the second set of UI is attractive to most participants, a few people have also stated a dislike to this certain style. Exception for subjective visual preference, the usability of the second set of UI outperforms the initial version in providing functional instructions and giving feedback (See Figure 22).

In the next stage of evaluation, 14 participants were given a task to create a shortcut freely and questions to assess the effectiveness and capability of the design of OneClick. The

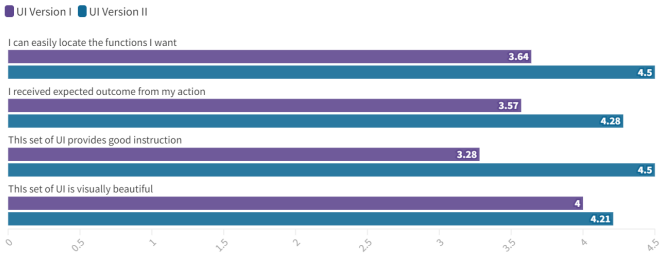

Fig. 22. Evaluation Result of UI Version I&II

time consumed to create a shortcut (See Figure 24), personal rating on the practicability of OneClick, their satisfaction with functional width and depth, as well as the willingness to continuously use the app were recorded and analyzed (See Figure 23). In the meantime, the whole process was recorded to spot any breakpoints when using the app.

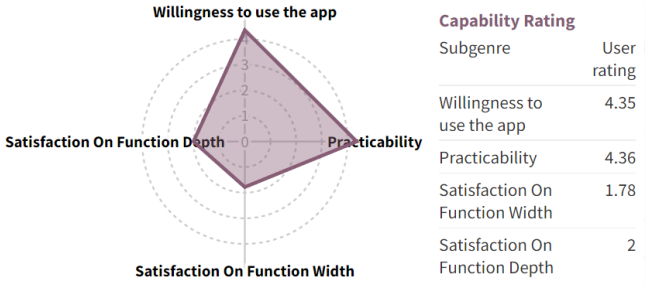

Fig. 23. Evaluation Result of Capability

To provide reference to the learning cost in time of using OneClick, the average time of creating a shortcut in OneClick by our designers is marked in gold and listed together with the result (See Figure 24). Among all the participants, one took over 10 minutes to create a shortcut of its own (Marked in red). The data is considered to be generated due to erroneous recording and is treated as an outlier.

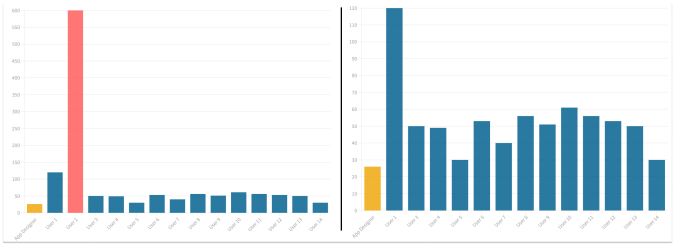

Fig. 24. Time consumed to create a shortcut

In addition to the evaluation of the user experience, the memory usage of OneClick is also examined as it is an app that requires to be operated constantly in the background. We kept the program running for three hours and record its average

memory consumption. It is found that the average consumption was about 80Mb and OneClick did not cause any lagging or crashing to the system operation no matter its focus status. Compared to an input method editor, OneClick consumed less average memory (See Figure 25).

Fig. 25. Memory usage of OneClick compared to a regular app

C. Usability Test Analysis

By analyzing the user flow of participants who took longer than average to create a shortcut, we found that the main problems affecting efficiency occurred during the creation.

Firstly, when creating a shortcut, some users assume that the system will automatically jump to the next stage once the selection for the current one is done. However, the jump is initiated through a "Next Page" button and is designed to be controlled manually in case further modifications are made (See Figure 26). To fix that, a Toast (Popup window) will appear to instruct the user to click the "Next Page".

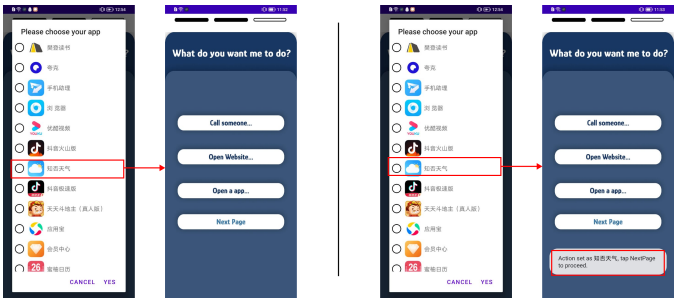

Fig. 26. Previously at creation stage (Left) and improvement (Right)

Secondly, it is found that when selecting an application as the final destination for a shortcut, some apps at the systematic level will be listed as well. Those apps can not be accessed through regular methods and will lead to a crash once a shortcut for opening it is invoked. Consequently, a filter is created to filter out those systematic applications and extract external apps only (See Figure 27). While this method works, it will also filter out some accessible system functions like the Clock, Camera and Text. The filter can be improved if more time is given.

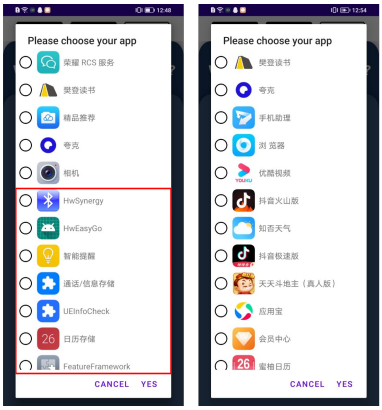

Fig. 27. Filtering out system application

From the previous analysis and evaluation, the overall usability and functional effectiveness of OneClick are proven to be valid. On one hand, as a hub app that does not have any independent function, OneClick as a whole is easy to learn and used and successfully serve its purpose of connecting certain systematic function to facilitate the daily actions of its user. On the other hand, the available options provided by the current system are considered to be limited and insufficient, as many have demanded an extension of the functional width and depth. Ideas are elaborated on the individual reports of our group members.

D. Design Alternatives - Analysis from Current Similar Product "Short Cut" from iOS

One similar shortcut product, and one of the inspirations of OneClick, is the "Short Cut" on the iOS platform (See Figure 28).

	OneClick	ShortCut
Platform:	Android	iOS
Learning Cost:	Low	High
Quick Dialing	✓	✓
Quick app access:	✓	✓
Quick browsering:	✓	✓
Floating Window:	✓	✗
Deep level programming:	✗	✓
Support External Import:	✗	✓

Fig. 28. Comparsion Between OneClick and ShortCut

As an application with system-level support, ShortCut outperforms OneClick in flexibility and interaction depth. However, OneClick is lighter, easier to learn and has higher accessibility for its floating windows function.

V. Conclusion

In conclusion, the report has illustrated the design and evaluation of an app called OneClick, which is a lightweight customized shortcut generator that can be directly accessed on the screen with a floating window whenever needed. The app was designed to achieve three instructive goals.

1. OneClick can be easily accessed without the process of locating it on the main screen.
2. OneClick works as a hub connecting other apps and some major functions of the system. It will generate short cut and visible interfaces for users to quickly jump from OneClick to another desired systematic destination.
3. OneClick provides an effective method for users to manage frequently used apps or activities.

The current functional and UI design of OneClick has fulfilled the three main goals proposed using the idea of OOP and user-centric approaches. It uses a floating window to provide access at any time, a shortcut managing system based on the class called pairbutton to create and invoke shortcuts and adopted a gesture-based function for users to differentiate the priority of shortcuts. Limited by time and capability, the depth and width of interaction for existing shortcut options can be further improved.

References

[1] A. Tarute, S. Nikou, and R. Gatautis, "Mobile application driven consumer engagement," *Telematics and Informatics*, vol. 34, no. 4, pp. 145–156, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736585316307006

[2] C. Sun, J. Zheng, H. Yao, Y. Wang, and D. F. Hsu, "Apprush: Using dynamic shortcuts to facilitate application launching on mobile devices," *Procedia Computer Science*, vol. 19, pp. 445–452, 2013, the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050913006686

[3] D. Thompson *et al.*, "Utilization of the ios shortcuts app to generate a surgical logbook tool: Feasibility study," *JMIR Perioperative Medicine*, vol. 4, no. 1, p. e24644, 2021.

[4] V. Trushkin. iconify. [Online]. Available: https://iconify.design/

[5] G. Kaur. flexiple. [Online]. Available: https://flexiple.com/illustrations/

[6] B. Poppinga, A. Sahami Shirazi, N. Henze, W. Heuten, and S. Boll, "Understanding shortcut gestures on mobile touch devices," in *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices amp; Services*, ser. MobileHCI '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 173–182. [Online]. Available: https://doi.org/10.1145/2628363.2628378

[7] C. Appert and S. Zhai, "Using strokes as command shortcuts: Cognitive benefits and toolkit support," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 2289–2298. [Online]. Available: https://doi.org/10.1145/1518701.1519052

[8] Mars-xq. Getting the size of screen of android device. [Online]. Available: http://t.csdn.cn/e0ist

APPENDIX

Fig. 29. Icon Design Draft

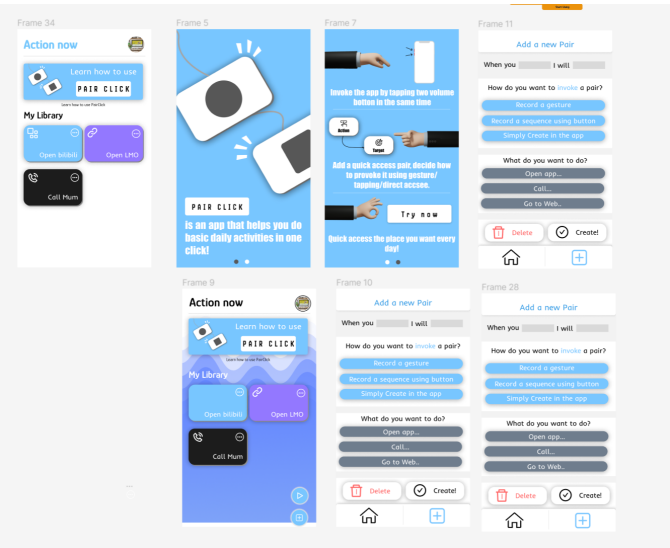

Fig. 30. Main UI design of Version 1

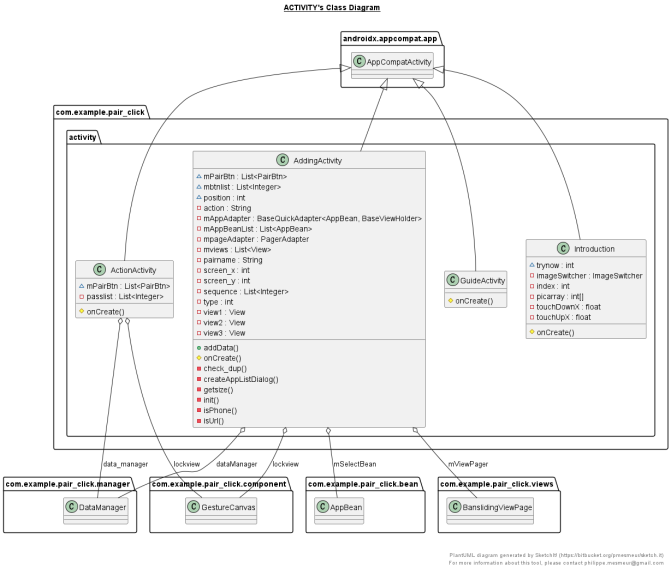

Fig. 31. UML of Activity Package

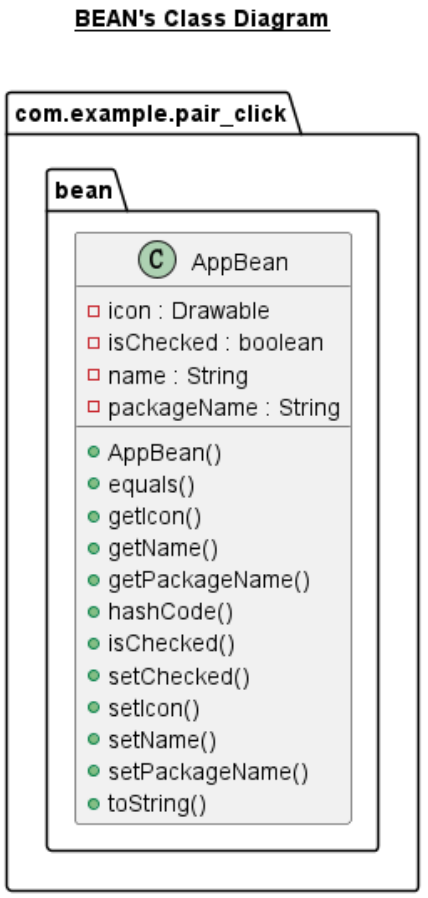

Fig. 32. UML of Bean Package

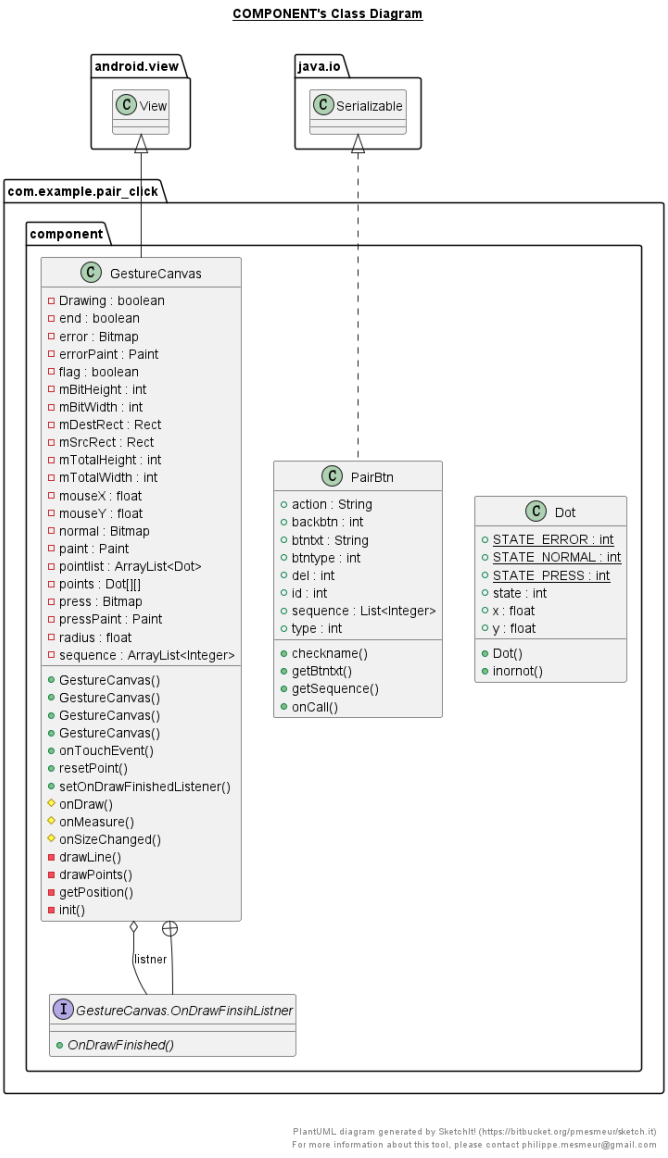

Fig. 33. UML of Component Package

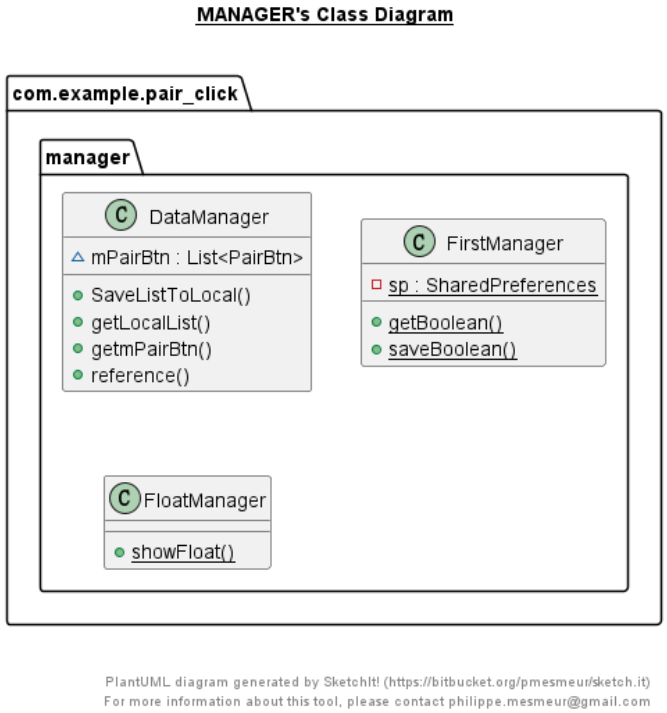

Fig. 34. UML of Manager Package

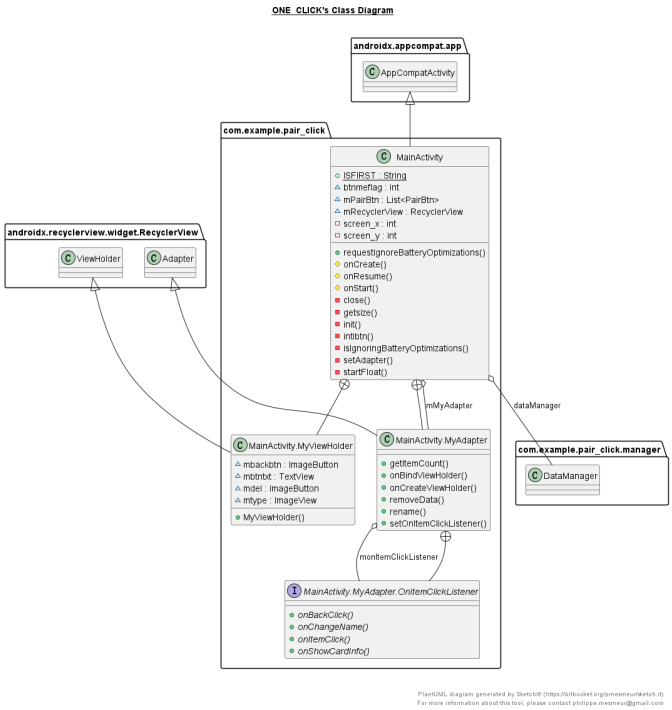

Fig. 35. UML of Main Activity

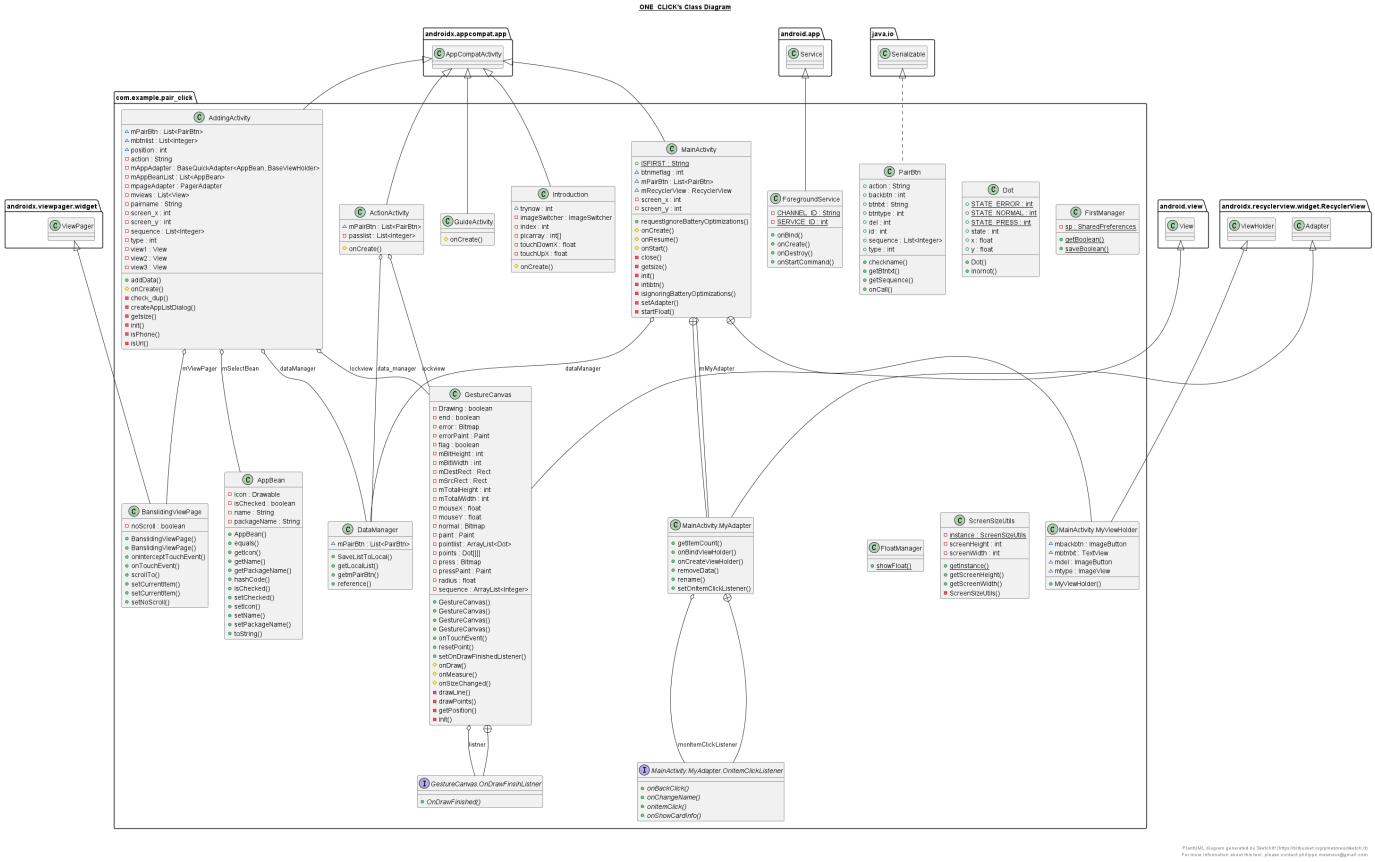

Fig. 36. UML of OneClick