# Introduction

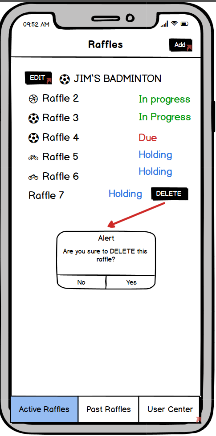
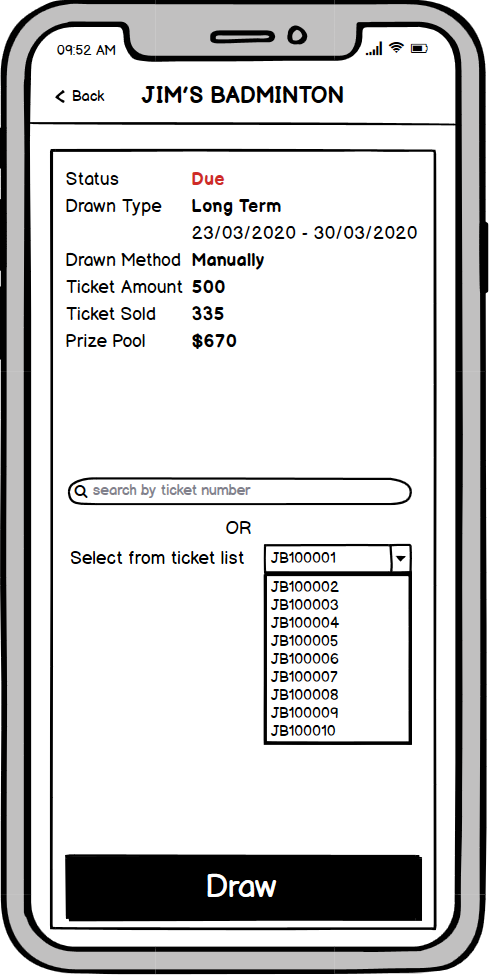
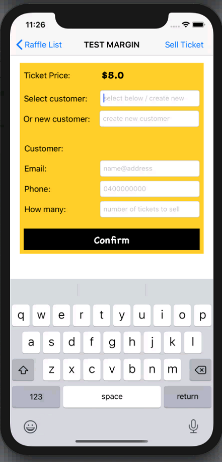
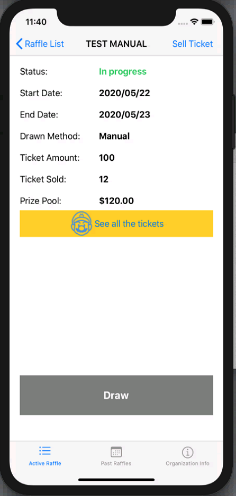
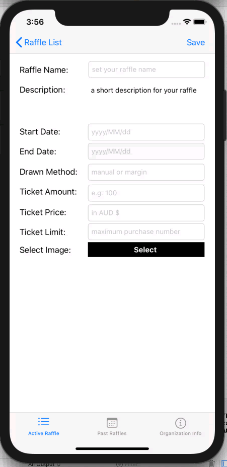
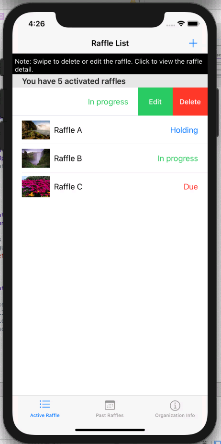
This report demonstrates the usability test on the newly-designed application named RaffleApp applying the concurrent think-aloud protocol in order to gain a profound understanding about how users will interact with this application in real-time and how to improve the overall experience based on the result. The report begins with description of the final version by explaining the functionality that aligns with the user-centered design principles. Then, the test methodology will be specified with detailed justification, following by the clarification of user feedbacks at each stage as well as how this raffle application has been refined iteratively based on results.

# Description of RaffleApp

RaffleApp is a native raffle application on IOS platform, which designs specifically for raffle organizers to host raffles, manage their client information and conduct drawing in an easier and more efficient way.

## Interface and functionality

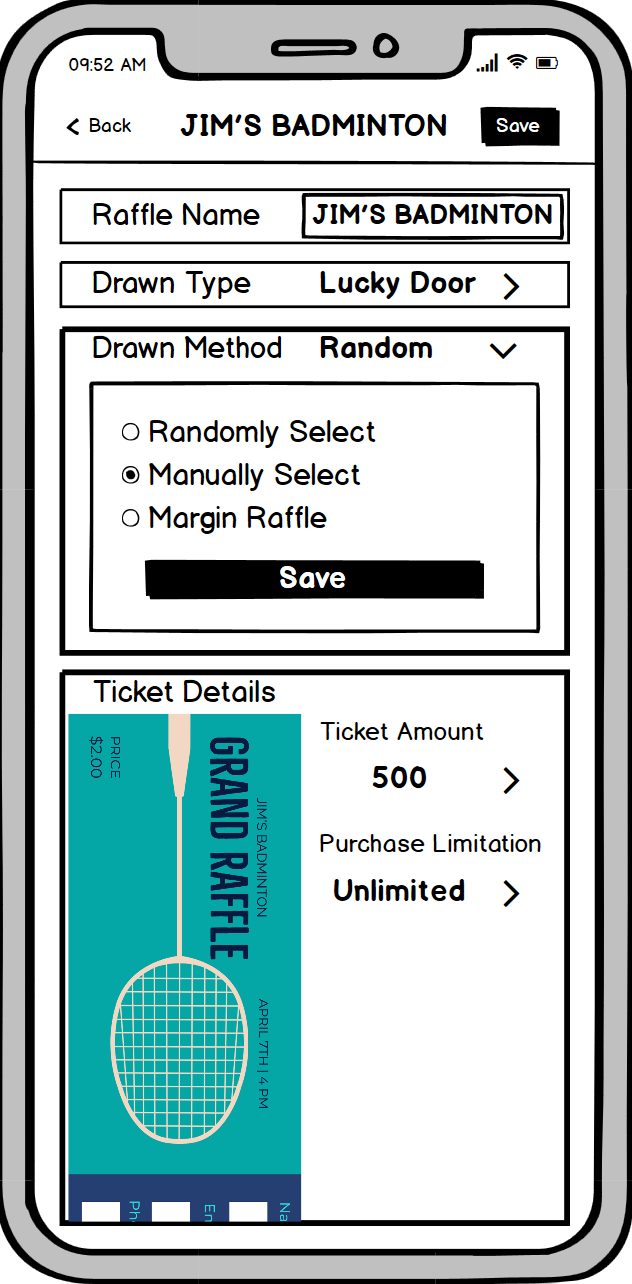
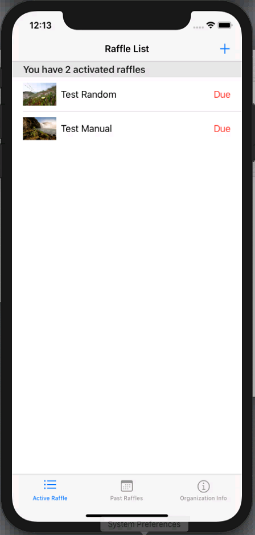
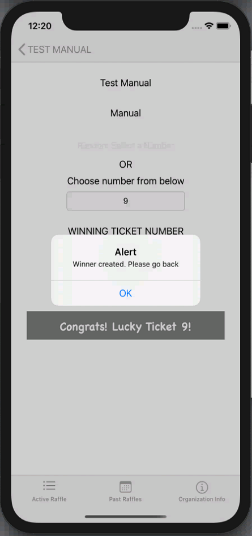
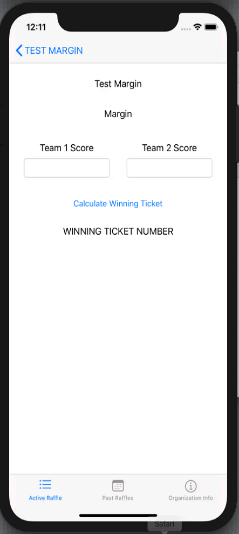
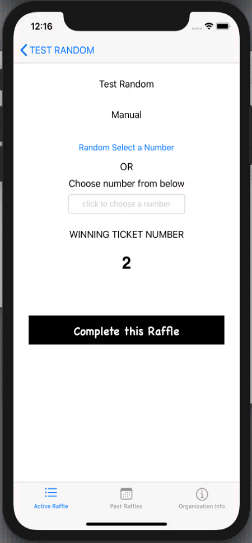
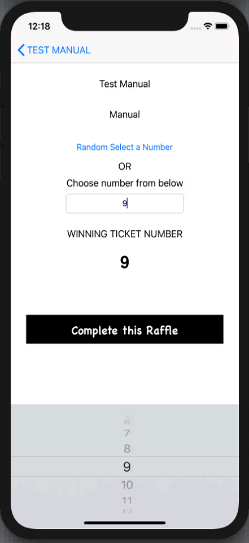
RaffleApp helps management of raffles and generation of winners. Figure 1 shows lo-fi and final interfaces for preparing raffles. The lo-fi prototype are different in terms of the position of buttons. It is originally designed that users can left swipe to delete and right swipe to edit. But in final version, if is simplified that users can edit or delete by one left swipe for enhancing efficiency. The ticket list is also different from final version which adds a button leading to a separate view of the whole list so that users can see entries with clear details rather than a drop box of name. Users can view the list of active raffles with the state that is either holding, in progress, or due. The ‘holding’ state used blue font, which implies the raffle has not started. The ‘in progress’ state is in green font. Any raffles that is due but has not been drawn yet will be highlighted in red color that acts as a reminder for users. The application also supports deletion of raffles that have not sold ticket and edit of raffles that have not been closed.

**Figure 1. Process of managing raffles (lo-fi prototype & final version).** *Source: own-creation.*

By a simple tap on the ‘+’ button at the top right, users can create a new raffle. Input of relevant details and image enables the customized specification by users. After the raffle being created successfully, users can view the details by tapping it. Users can add ticket entries to this existing raffle by tapping on the ‘Sell tickets’, which is similar to the raffle creation. In the ticket view, the predefined price is labelled automatically at the top. Users can input the individual information such as full names, emails, phone numbers and the number of purchased tickets. The list of sold tickets is available after entering the ticket entries, and users can view the detail by tapping on ‘List all ticket’.

When raffles are due, users can begin generating the winner by random select, manual assigned, and margin select, as is illustrated in Figure 2. The main difference is that the lo-fi design has three draw methods in the same page, while views in final version are different according to the identified draw method. The final version also eliminates the description of tickets that simplify the interface and lay more focus on the draw. For manual raffle, users can either select the number from sold tickets, or using automatic generation of random number. For margin raffle. When starting the margin raffle, Score of two teams is required to be identified so that the application will do the calculation for selecting the winning number. It is noteworthy that, unlike other two drawing methods, margin raffle assigns random number to clients to ensure the fairness of competitions, thus it may not have the winner who purchased the exact number. The completed raffle after drawing will be sent to past raffles.

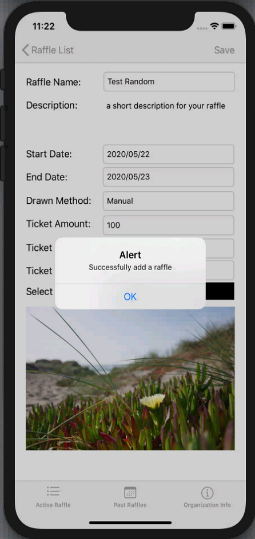
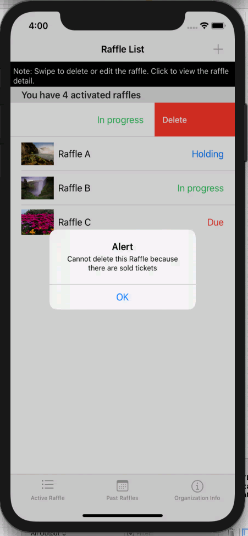
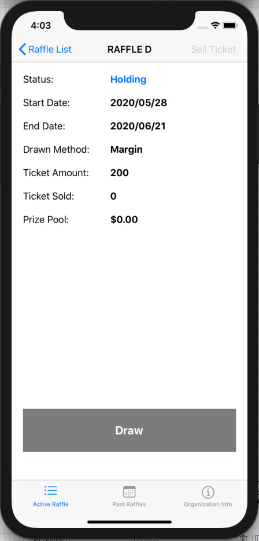
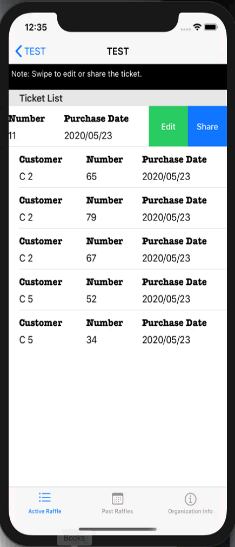
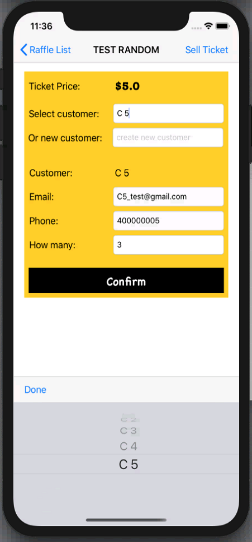
 

**Figure 2. Process of completing raffles (lo-fi prototype, random, manual, margin).** *Source: own-creation.*

## Alignment with efficiency

To improve the efficiency of users, RaffleApp helps adding client information much faster (see Figure 3). When adding tickets to existing raffles, users can directly select existing clients from the name list, and the client information can be automatically filled without having to enter same client repetitively, which is tremendously valuable for clients who purchased tickets periodically. Users can also create a new client, and the information will be automatically saved after adding the ticket which can be directly used for future. Additionally, the ticket-creation view of RaffleApp has the field for entering number of tickets. That is to say, if the client has bought multiple tickets for the same raffle, users can aggregate the amount of ticket purchased by this client and enter the number. RaffleApp generates multiple entries without having to manually enter the entries for several times.

Moreover, RaffleApp provides instructions, constraints and feedbacks to reduce the difficulty. It shows conspicuous tutorial on the top reminding users of swapping to edit and delete raffles or ticket entries. To avoid users from misuse, ‘Draw’ button is disabled until the due time and ‘Sell tickets’ button is only enabled for on-progress raffles. If users are not doing correctly, such as delete raffles that have already sold tickets, the error message shows up. Apart from alert for misuse, RaffleApp also returns the success message, for instance, confirming saved state of newly created raffles or the competition of raffles after the draw.



**Figure 3. Examples of how RaffleApp improves efficiency for users.**

# Methodology and usability testing

For testing the usability of RaffleApp, three rounds of prototype test were employed via concurrent think-aloud protocol, including two rounds of Balsamiq-based test and one round of coded-version test. The purpose is to ensuring that the design actually matches the end-user’s need by comprehending their mental model (Lindsay 2020).

## Specification of test decision

The usability test applies the think-aloud protocol which is the direct approach to users’ thinking. Participants are asked to vocalize their feedback and opinion of the task they are performing on the given interface of RaffleApp. Inquiry methods such as interviews were not used during the test due to the social restrictions under COVID-19. And we discarded the inspection-based method due to the limited time and difficulty to find experts at this time.

There are two widely-adopted type for think-aloud known as concurrent and retrospective method. Our think-aloud protocol uses the concurrent verbal accounts because we want to acquire more information from participants. Obviously, retrospective method has its advantage comparing with its counterparts that it is proved to be easier to run (Brian & Catharine 2014) and participants may perform better when they are quiet (Prokop 2020). However, it is blamed for leading to less data (Wabil et al. 2010) since it is limited by the reported feedback from users (Koch et al. 2019). Moreover, thanks to the direct understanding about where they got stuck (Michel & Ivana 2020), we realized that the concurrent method helps linking users’ feedback with the particular button or selection boxes that makes refinement and fixation much easier. Admittedly, we found bias related to concurrent method since users appeared nervous when being observed and they felt shy to criticize. Thus, we encouraged them to speak and gave positive response for their critique. One possible disadvantage is that it took us more time and energy on process planning because users had to vocalize what they are doing besides their thinking.

We adopted the qualitative method for collecting and analyze the user response. Observation on their facial expression and recording of their words are combined for data collection after permission was offered. Although qualitative method showed more bias than quantitative method, we thought that these human-led biases were also valuable for designing since we want feedbacks with various aspects to fit the largest user group.

## Participants

Formative method was used at three stages of programming and iterative adjustment to users’ requirement was done after each round. It helps refining RaffleApp with less cost and energy (Bahrami 2019) compared to testing when everything has finished. Twelve people were planned to be invited and divided into three groups for three rounds of test respectively. Four is the appropriate size each group for getting enough feedback considering of the given time as well as budget, and studies proved more participants tends to give repetitive information (Jakob 2000). Fresh users are needed for each round of test. We abandon the alternative option of having same people do multiple rounds, in that despite the convenience and low cost, it creates bias since they already had experience with RaffleApp which is unrepresentative of end-users.

We decided to recruit participants from people that we knew. We did not apply the traditional recruiting such as emails or Facebooks. The main reason is that people tend to be reluctant to have connection with strangers due to the COVID-19. Another issue is with the security of personal credentials. Because the test can only be done via video conferencing, people do not want to risk leakage of personal information. It is possible that participants are still photographed without permission because they cannot ensure their privacy remotely. Thus, we targeted the user from our friends. Inevitably, this narrows the user group which has the potential bias of sharing similar knowledge when using the application. In that case, we try to find people with distinctive age or background to represent users.

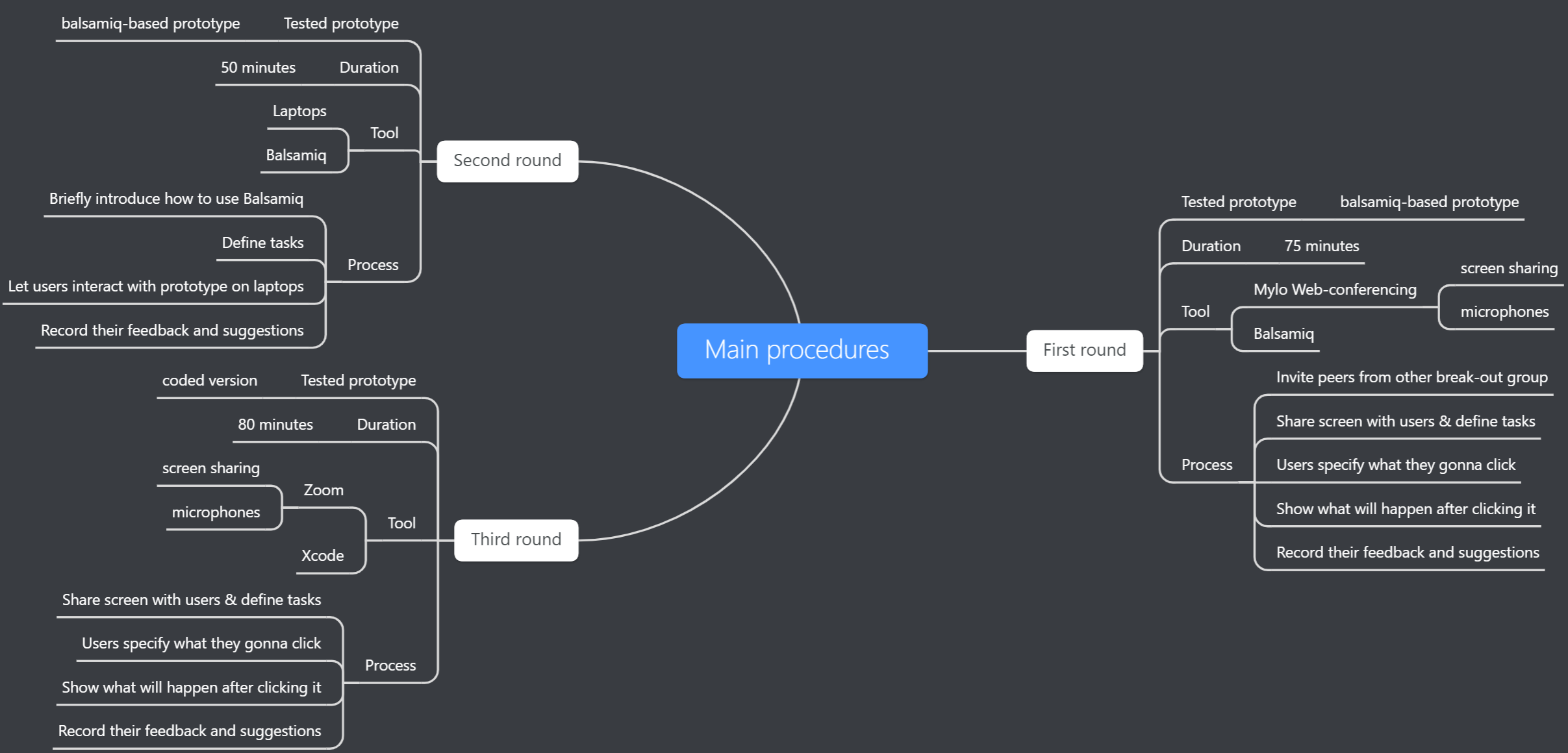
For the first round, four classmates from the same tutorial were invited to our test. They are recruited by swiping group members to test mutually on each other’s prototype. It is beneficial for exchanging ideas and innovating UI designs. However, it also has bias considering that these participants are from the same major who possess strong IT literacy. More influentially, they are designers of similar applications. In other words, they have the relative knowledge before using the application such as basic functionality. They may also know the coming task since they have the experience designing the similar function. All of these had side-effect on the test result.

For the second round, four room mates were involved who are easiest to be directly invited because of sharing same accommodation. Unlike the previous round, these peers have distinct background from the education and maritime engineering major, and they aged from 20 – 37. It is better than the last test since they have little background knowledge about the application itself and they have gap between ages, which are closer to real users. Admittedly, the bias is that they do not have experience relevant to raffles, let alone hosting the raffle. Thus, the majority of their confusion located in understanding the elements and process of raffles, rather than using the application.

For the third round, we are lucky to have four participants from City Baptist Church who have hosted many raffle events locally. They are invited via phone calls by one of the team members who has been volunteered for their charity. These participants are desirable comparing to previous two rounds, since they are representatives of our end-users. The only bias is their lack of experience using application.

## Test procedure

Figure 4 summarizes main procedures for each round of testing. All test involves a brief introduction of application purpose and the sincere gratitude in the end. Positive response and supporting encouragement were highlighted throughout all tests. Leading were avoided carefully. The first round of test is based on the selected rudimentary prototype from previous assignment. We invite peers from other break-out group during the tutorial and communicating through web-conferencing. Screen sharing and microphones were used to define tasks and give feedbacks. After defining each task, users were asked about what they want to do and we showed them what will happen after doing it through screen sharing. Their hesitation and suggestions are recorded in note-sheet.



**Figure 4. Summary of three round of main task procedure***. Source: own-creation.*

We performed the second round of test on the modified Balsamiq prototype which has been improved based on the result from test one. This time, we found four peers sharing the accommodation. The advantage is that we can actually observe their reaction and let them interact with the application by themselves. However, we still have to follow the social distance restriction of 1.5 meters so we can only watch their behavior from far. Apart from the task defining, we have to introduce briefly how can they interact with the prototype using Balsamiq. We let participants use our laptops to emulate using application so that no transfer of files was needed and our roommates did not have to download the software. They gave permission on video taking so that we can view recordings afterwards in case we have missed something due to the distance restrictions under COVID-19.

The third test was on our coded product. Four participants were invited to the hosted Zoom group and asked to state what they want to tap for performing each task. We will do exactly the same to emulate their doing by themselves. They can view the response of application after each activity through screen sharing. We got their permission of video recording so we utilize the record function of Zoom. Three of them are elderly and not familiar with application or computer, so it took us the longest duration to complete the test.

# Result

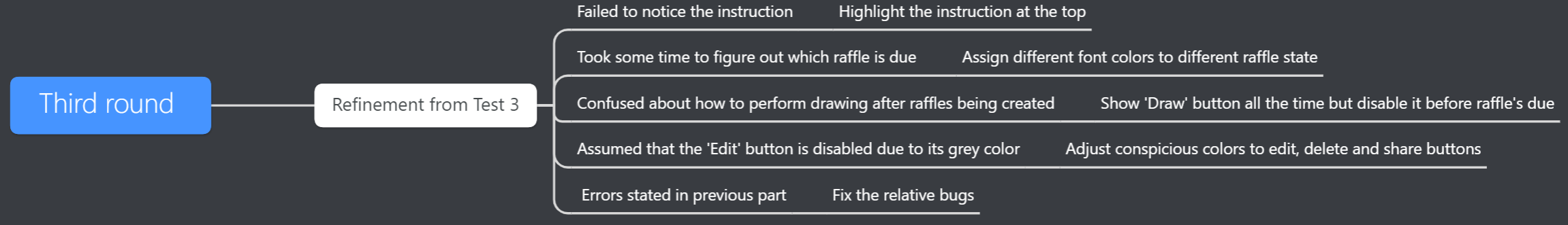
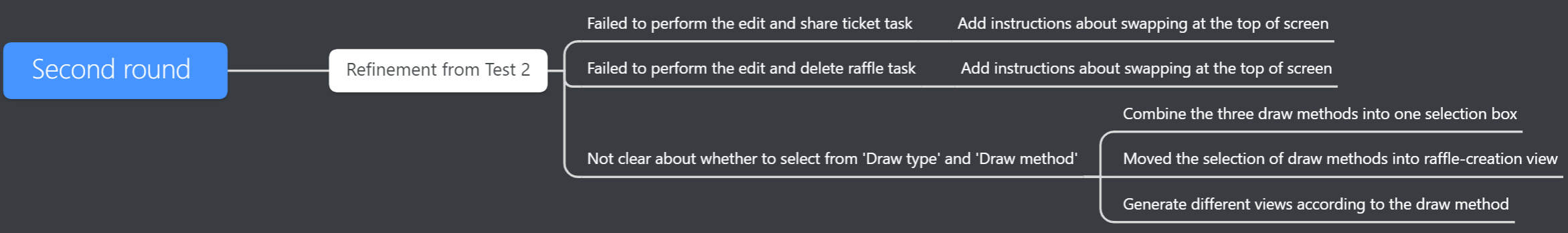
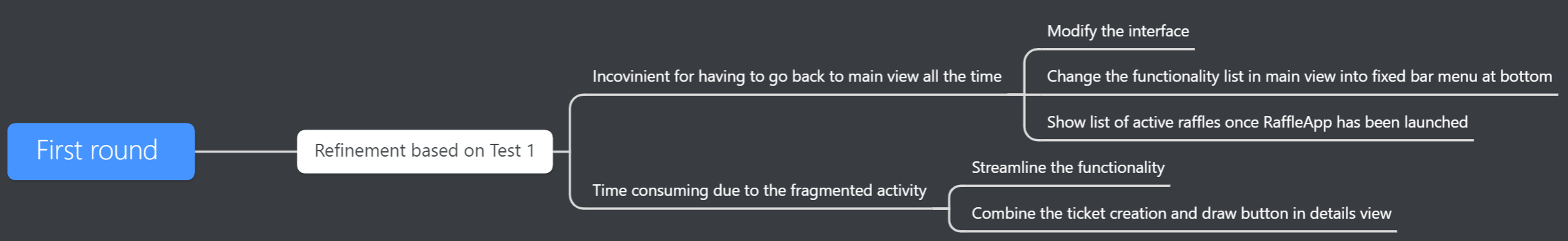
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Participants | Duration | Success tasks | Observations | User comments |
| P1 |  |  |  |  |
| P2 |  |  |  |  |
| P3 |  |  |  |  |
| P4 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Participants | Duration | Success tasks | Observations | User comments |
| P1 |  |  |  |  |
| P2 |  |  |  |  |
| P3 |  |  |  |  |
| P4 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Participants | Duration | Success tasks | Observations | User comments |
| P1 |  |  |  |  |
| P2 |  |  |  |  |
| P3 |  |  |  |  |
| P4 |  |  |  |  |

# Discussion

This part addresses the understanding of users’ mental model based on the iterative refinement from previous test results (see Figure 5) and looked into depth about users’ vocal feedback and potential biases of each stage.



**Figure 5. Iterative refinement after each formative test based on user response.** *Source: own-creation.*

For the first round, biases occurred in that all participants are classmates who have supreme IT literacy. They have pre-understanding about the application and required tasks. Additionally, we did not capture their facial expression in that they did not open the camera. The main information comes from their hesitation time and verbal suggestion about what they like and dislike about this application. We got the least information in this stage in that they were familiar with the task performance. Valuable information mainly came from their suggestions on our interface and menu options. At this first stage, our prototype is criticized about having to go back to main view after performing each activity which is inconvenient and time-consuming. Thus, instead of having all functions listed in the main view, we changed it into fixed menu bar at bottom. We also combine some options, such as create tickets and draw raffles, into the raffle-detail view so that users can easily perform various tasks from this streamlined design.

For the second round, biases occurred because participants lacked their experience relevant to raffles, which is non-repetitive to our end-users. It took us extra time to explain the nature of raffle. They got confused about ‘Draw type’ and ‘Draw method’ when they are asked to perform the margin draw. Thus, we combined them into one selection box and moved the draw defining into the creation view. Also, to enhance the easiness, we programmed in the way that system will show different views of drawing according to their selection of draw methods. We add instructions to tell users using swap to find edit and delete buttons due to their failure to perform the edit task. It is also found that having the ‘Add’ button at the bottom will block some new raffles as the list growing bigger. So we move the ‘Add’ button to the top right which coincide with IOS users’ habits to a greater extent.

For the third round, participants that can best represent our end-user were invited. However, most of them are elderly who are unfamiliar with application. It inspired us more thinking about how to generalize it to people with little computer literacy. Based on their feedback, we added more constraints, instructions and feedback to the application, as is stated previously in this report. We also addressed the font color to highlight buttons and different state of raffles to remind users and to mitigate potential mis-interpretations.

Create customer add information every time, only select name

Welcome

Check list of customers

Unable to create continuously

Tutorial on the top

Swap too much will delete

Button draw will appear at first rather than only exist after due, customer may not know how to raffle in the future

# Feedback

## Repetitive title, messy

### Unable to return automatically

#### Due timing

##### Remind

###### What if customer update info

Method & Type

Visualized should not be shown at edit page

Abandon the left and right swap

Three rounds