UNIVERSITY OF NOTTINGHAM NINGBO CHINA

# Heuristics Evaluation

## AE2HCI Group report, Coursework 2

**Group 2**
**12/9/2015**

**Module convener: Eugene Ch'ng**
**University of Nottingham, Ningbo, China**
**School of Computer Science**

***Group 2:***

| | |
|---|---|
| Alvin Hartanto | 6514055 |
| Lilyana Stanimirova Dimitrova | 6517724 |
| Shihang Wang | 6514904 |
| Yichen Pan | 6514897 |
| Meng Yuan | 6513307 |
| Hongzhi Zhang | 6514911 |


## Contents

This group report is divided into 4 parts. We start by explaining the overall evaluating process and methodology used, then we proceed with an extensive evaluation of each prototype involved. The report ends with a brief group conclusion.

## 1. Introduction

We have followed the general steps of heuristic evaluation suggested by Nielsen and Molich (1990). By holding multiple meetings, we managed to go through the prototypes in a great detail. The result is hereby presented in this report. The detailed evaluation process is illustrated in Appendix A.

## 2. Prototypes Evaluations

Following evaluations are based on the mobile-specific heuristic guidelines we generated particularly (Appendix. B). Additionally, the severity of each usability issue is rated according to the appointed scale (Appendix. C) and the severity is represented by a red floating point at the end of the sentence (e.g. "(3.6)"). The number is the average value of all the severity marks appointed by the evaluators for the particular issue. Additionally, individual evaluation and severity ratings of prototypes can be found in Appendix. D.

## 2.1. Evaluation of Alvin Hartanto's prototype

*Interface screenshots.*

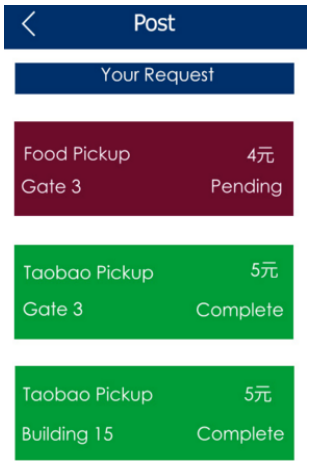


Fig. 1.1. Requests screen

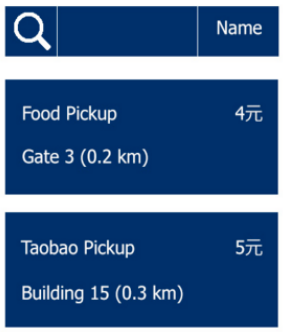


Fig. 1.2. Requests by other users

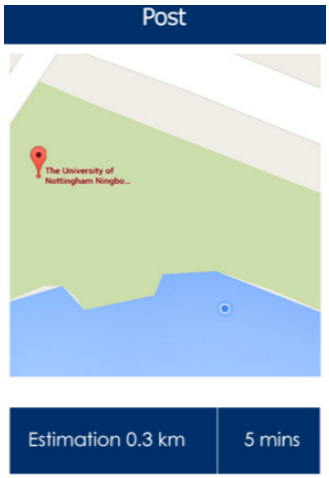


Fig. 1.3. Location screen

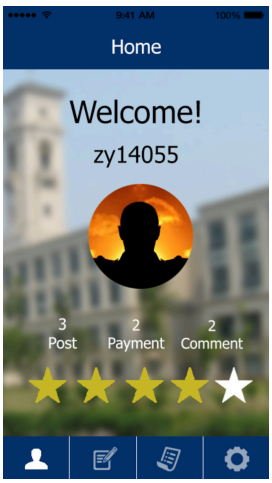


Fig.1.4. Main screen

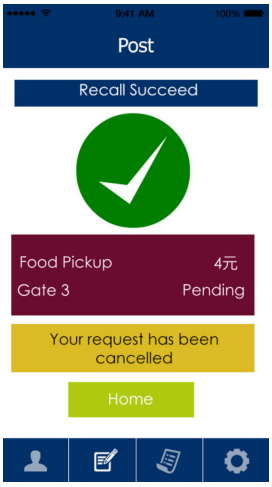


Fig. 1.5. Canceling a request

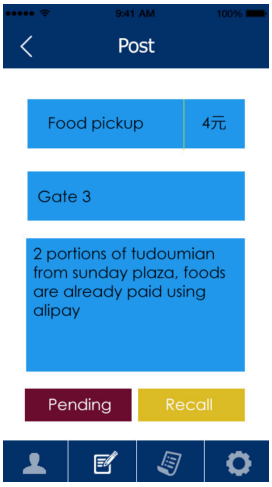


Fig. 1.6. Viewing a request


Fig. 1.7. Signing up

**Requirements checking**

Based on the user requirements list of this app, it has fulfilled all mentioned requirements.

**1. Visibility of system status**

1.1. This screen lists all the request a user has made (Fig. 1.1). Although the posts are color-coded according to their status (green for completed, purple for pending), there is no option to sort the list and show all the posts of a certain type. On the screen which shows requests made by other people (Fig. 1.2), the system does not provide any color-coding but it has a way of filtering them. The filtering button,

however, does not follow appropriate conventions so user will not intuitively know it is an active area. **(3.0)**

1.2. The system prioritizes the list's elements in a convenient way. It is more plausible for the user to be searching for the pending tasks, so they are put on top while the completed ones go at the bottom of the list. (Fig. 1.1).

1.4. The interface does not have known to be hard to find active areas (such as swiping in a specific direction). On the other hand, there are a few options that are not visibly presented as clickable areas. As shown in Figure 1.4, 'Post', 'Payment' and 'Comment' are actually touchable buttons. However, the designer does not follow the conventions for indicating them as such and the user will not gather that immediately. **(3.0)**

### 2. Match between system and real world

2.1. The system uses adequate terms which would be easy to understand by the user (e.g. "request"). However, as shown in the Fig.1.4, although the designer adopted metaphor design, their concrete functionality is difficult to be understood by the user intuitively. **(2.0)**

### 3. User control and freedom

3.1. The app does not provide an easy way to leave unwanted state without extended interaction or effort. There is no "Go back" or "Cancel" buttons. **(3.7)**

3.3. The system does not provide customization.

3.5. There are no marked exits, neither from a task ("Cancel" option) nor from the app itself. **(3.0)**

3.6. The system does not ask for a confirmation about payment. **(4.0)**

### 4. Consistency and standards

4.2. The app has two contrasting main colors and three others who are occasionally used. However, some screens have too many colors used which makes it look chaotic and distracting (Fig. 1.5). **(2.0)**

4.3. The app follows different conventions for buttons and label design. On Fig.1.6, the rectangle on the left - 'Pending' only indicates the state of the request while the one on the right is an actual touchable button, but they look the same. The user will not be able to intuitively understand the difference. **(3.0)**

### 5. Error prevention

5.1. After creating a request, the user is able to accept or deny willing participants. However, when a user wants to take up on a request, the app does not inform him whether or not he/she has been accepted. **(2.8)**

5.2. The app does not provide any warnings before doing high-risk tasks (e.g. payments). **(3.6)**

5.3. There is no indication whether or not the app adequately explains the errors to its users. **(2.0)**

### 6. Recognition not recall

6.2. The interface mostly follows design conventions with some exceptions. As a standard, the 'Cancel' option is usually placed on the left side and 'Confirm' is placed on the right side. However, the layout shown in the Fig. 1.7, goes against this convention. **(2.0)**

### 7. Flexibility and efficiency of use

7.1. The app provides a menu visible at the bottom of the screen at all times which makes it easy to navigate.

7.3. The app is relatively easy to learn even though it has no help provided.

7.4. Since there is no help or tips provided, less tech-savvy people might not find the app just as easy. **(3.0)**

## 8. Aesthetic and minimalist design

8.2. The interface has a clear contrast although the color-coding does not follow conventions (purple for "pending", yellow for "accept" instead of red and green).

## 9. Help user recognize, diagnose and recover from errors

There is no indication whether or not the app supports users during errors (No visible warnings, explanations or suggestions).

## 2.2. Evaluation of Lilyana Dimitrova's prototype

*Interface screenshots.*


Fig. 2.1. Public Board

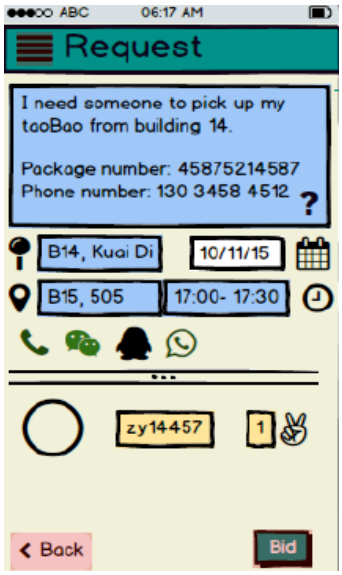


Fig. 2.2. Request View

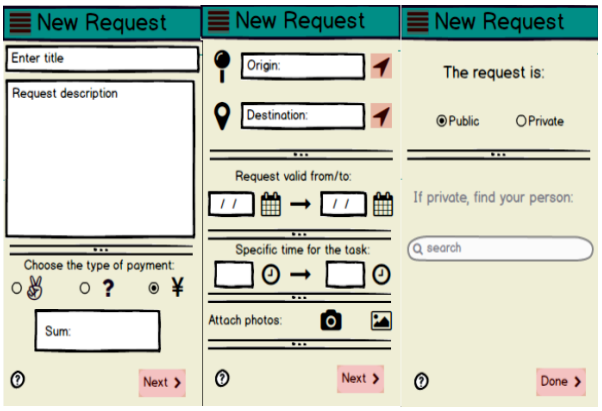


Fig. 2.3. Request making process

### Requirements checking

Based on the user requirements list of this app, it has fulfilled all mentioned requirements.

### 1. Visibility

1.1. The "Public Board" holds all the requests made by the users (Fig. 2.1). Each entry is constrained in a colored box to visually separate them. However, there are no means of searching or filtering the requests. **(2.0)**

1.4. All of the interface elements that act as buttons are designed as such. The only difficult to discover action is refreshing by pulling down the screen (on screens with lists, such as Fig. 2.1). **(1.0)**

1.6. Making a request requires the user to go through 3 steps divided into 3 following screens. The users cannot keep track of which step are they on currently and how far they are from the end goal (Fig. 2.3). **(2.0)**

### 2. Match between system and real world

2.1. The interface does not exceed the appropriate count of icons. However, it is unclear to new users what the types of requests mean since they are only marked by symbols ( , , ) but provide no explanation. **(2.6)**

2.2. The icons used for indicating the beginning ( ) and the end of the destination ( ), by convention, bear a different meaning (current location and destination). **(2.3)**

### 3. User Control

3.1. Creating a request requires a lot of information to be input. To reduce information clustering, the process is divided into 3 parts, each with its own screen. It, however, does not provide the user with a safe

exit at all times (Fig. 1.5). **(2.6)**
3.6. There is no indication how the app handles payments, which is one of the most safety-crucial tasks. **(3.0)**

### 4. Consistency

4.2. The app does not exceed the recommended count but potential users have indicated that the current color palette is too dark. **(1.0)**

### 5. Error prevention

5.2. The current version of the prototype has no indication how the app handles the "Cancel" and "Edit" options.
5.3. The current version of the prototype has no indication how it handles different types of errors.

### 7. Flexibility and efficiency of use

7.1. The system provides access to the main menu with a bar at the top of the screen at all times.
7.3. Since the app does not provide beginners with any tips or tutorials, it may not be as easy to learn as desirable. **(2.3)**

### 8. Aesthetics

8.1. The interface implements too many icons and not enough text. This is mainly a concern with the unexplained requests' types. All of the other icons are quite standard (e.g. calendar, time). **(2.0)**

### 10. Help and documentation

10.1. More complex tasks' screens provide the user with an explanation-on-demand (Fig.2.3).
10.2. Help documentation is not consistent throughout the app. Although there is an easy-to-access help documentation during some tasks (bottom left corner, Fig. 1.5), it is located elsewhere on others (top right corner) or it is missing all together. Furthermore, there is no initial extensive explanation of the main features. **(2.5)**

### 12. Pleasurable and respectful interaction

12.1. When creating a new request, the amount of typing is not reduced by the system. Given the nature of the app, it would be useful to provide the user with a list of commonly used requests to choose from (e.g. delivery pick up). **(2.6)**

### 13. Privacy

13.1. Since the app implements payments, it is safe to assume that it should hold confidential bank account details. The current version of the prototype does not indicate how the system handles that. **(3.0)**

## 2.3. Evaluation of Shihang Wang's prototype

*Interface screenshots.*

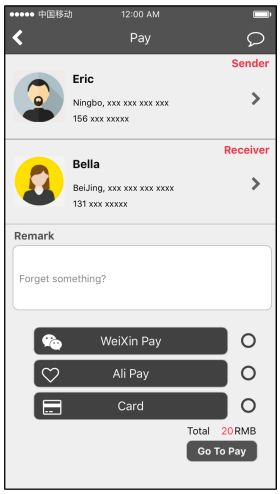


Fig. 3.1. Payment screen

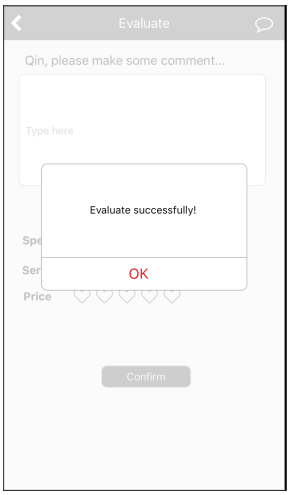


Fig. 3.2. Comment response

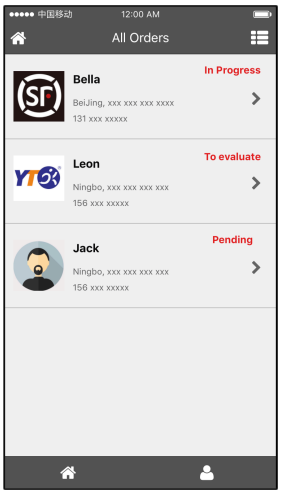


Fig. 3.3. Requests view

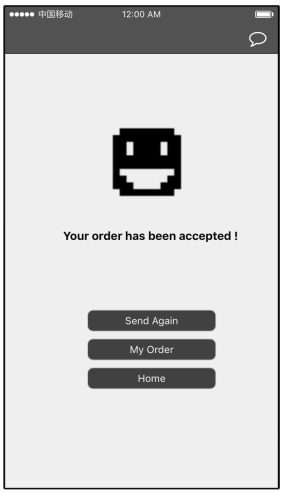


Fig.3.4. Order response

### Requirements checking

Based on the user requirements list of this app, it has fulfilled all mentioned requirements.

### 1. Visibility of system status

1.4. Upon clicking a button, its appearance does not change, which way may lead to the user's confusion. **(2.0)**

### 2. Match between system and real world

2.6. By conventions, red color is typically associated with danger and warnings. However, the interface uses red color to differentiate between senders and receivers (Fig. 3.1 and Fig. 3.3) which is neither dangerous nor risky. Upon glancing the screen and seeing the colors, the user will be alarmed for no reason. Moreover, when presenting options, usually the radio button is the clickable area. However, in the interface (Fig. 3.1), the text label on the left is the touchable content. Furthermore, the checkbox is usually placed at the left side, instead on the right. **(3.0)**

### 3. User control and freedom

3.1. Most of the time, the app provides the user with a return function which allows them to move forward and backwards relatively easy.

3.6. The app does not ask for confirmation before proceeding with high-risk tasks (such as payment). After the user presses the 'Go to Pay' button (Fig. 3.1), the app does not make sure that the user is making a deliberate action. **(4.0)**

3.7. After the user taps the 'Confirm' button, indicating he wants to submit the comment, the pop-up window does not provide any option for users to cancel the 'Cancel' operation. **(3.6)**

### 4. Consistency and standards

4.2. The app has two main colors, which offer a clear contrast. Aside from that, different statues are all colored in red.
4.4. The app is not consistent. Sometimes there is a bottom menu, sometimes there isn't. Sometimes there is a chat button on the top right, but sometimes it is "Submit" button. **(3.0)**

### 5. Error prevention

5.1. Menu choices are logical and mutually exclusive but their layout is not consistent.
5.2. The system does not ask for confirmation before proceeding with high-risk tasks (such as payment). **(3.3)**

### 7. Flexibility and efficiency of use

7.1. At most times, there is a menu at the bottom of the page which offers navigation to home page or user profile.
7.3. The app is relatively easy to learn depending on how tech-savvy the user would be.

### 8. Aesthetic and minimalist design

8.1. The app has a good balance between text and icons which makes it easier to navigate. However, as shown in Fig.3.3, the two 'house' buttons, one in the top-left corner and the other one in the left-bottom corner, are exactly two same buttons, with same functionality, but exist in one screen. This is redundant and unnecessary. **(2.6)**
8.2. The clear contrast of the design orders information correctly but there are no informative colors that could help. Furthermore, the statuses of the requests are all labeled with red font. On an intuitive level, this will alarm the user at first even though the red label says "Completed". **(3.0)**

### 11. Skills

11.1. After the user confirms the payment, the app should not leave the user in the transitive screen. Instead, the app should jump to the home page or 'My Order' screen based on predictions of user's activity. **(2.3)**

### 12. Pleasurable and respectful interaction

12.2. The app has the function to save frequently used pieces of information (default address).

## 2.4. Evaluation of Yichen Pan's prototype

*Interface screenshots.*

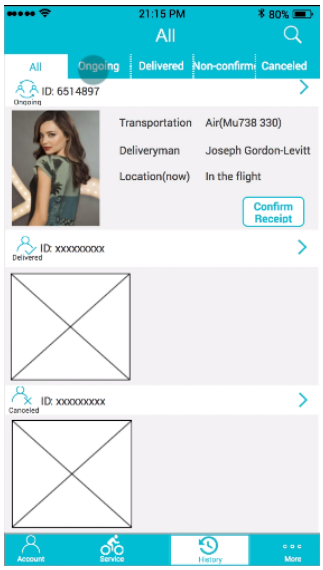


Fig.4.1. Deliveries lists

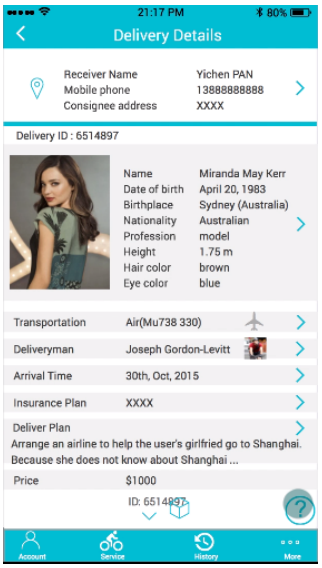


Fig. 4.2. Delivery details

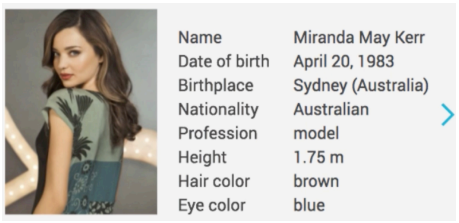


Fig. 4.3. Deliverable details

**Requirements confirmation**

Based on the user requirements list, the app has fulfilled all mentioned requirements.

**1. Visibility of system status**

1.1. The app allows the user to search by keyword or filter entries with a specific status.

1.2. The app presents entries in a list in an easy-to-scan way (Fig. 4.1). Each entry is marked by a status icon and all the important information could be seen.

**2. Match between system and real world**

2.1. The 'History' option usually indicates the past and already completed orders. In the system, however, the ongoing orders are also included in it. **(3.6)**

2.2. There is one button that could be a little ambiguous. By convention, a circle with a question mark in it would mean source of help (user manual, tips, explanation). In this case, it means accessing the AI of the app. **(2.5)**

**3. User control and freedom**

3.6. The system warns the user whenever there is an irreclaimable or high-risk action ahead (such as payment).

**4. Consistency and standards**

4.1. The interface does not exceed the recommended amount of icons.

4.2. The app has two color patterns according to the type of activity the user wants to make (deliver objects or ordering deliveries).

4.4. Even though the interface has two modes and color patterns, they both follow the same design scheme.

### 5. Error prevention

5.4. Some interfaces are too clustered (Fig.4.2). **(3.3)**
5.5. Some of the hit areas in the "Delivery Detail" screen may prove to be too tiny for some users (Fig 4.2). **(3.5)**

### 7. Flexibility and efficiency of use

7.2. It provides a search option whenever relevant, which upon clicking will show a visible search box. It is both efficient and saves layout space.
7.4. Since the app follows all the appropriate conventions, any average mobile user will have no problems getting familiar with the system.

### 8. Aesthetic and minimalist design

8.1. Some information is not necessarily needed to be display (Fig. 4.3). **(2.0)**
8.2. The interface has a clear, up-to-date look implementing icons and color-coding. It also uses background contrast to highlight and divide important chunks of information (In Fig. 4.1. we can see that the background of the text in the details is gray, while the general background is white). This aids scanning without reading the whole screen.
8.3 Some interfaces are too visually clustered.

### 12. Pleasurable and respectful interaction

12.1. The system implements a lot of features to reduce the typing from the user. First, there is the AI which supports voice input. Secondly, the system is able to read information from the photos uploaded by their users (such as the size of the deliverable).

## 2.5. Evaluation of Meng Yuan's prototype

*Interface screenshots.*

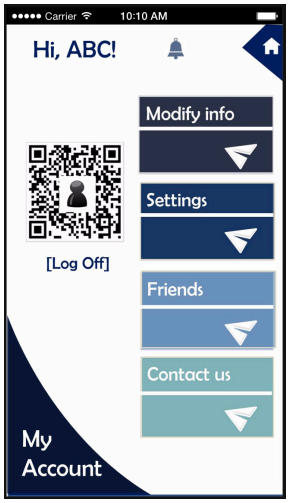


Fig.5.1. My account

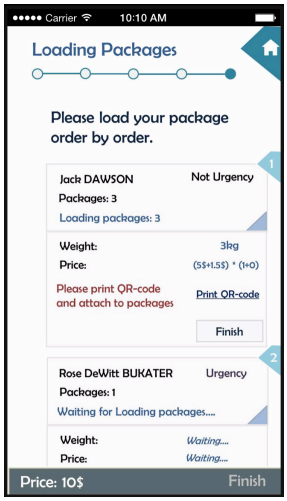


Fig. 5.2. Loading packages

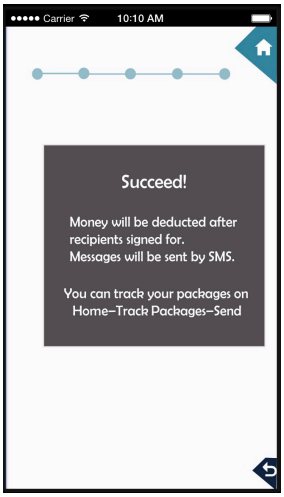


Fig. 5.3. Success response

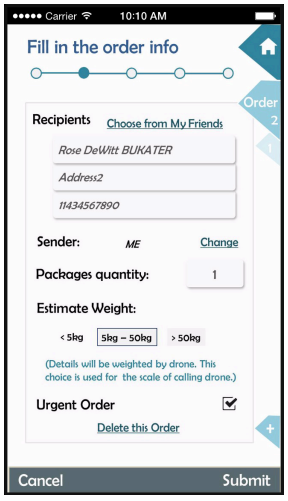


Fig. 5.4. Making an order

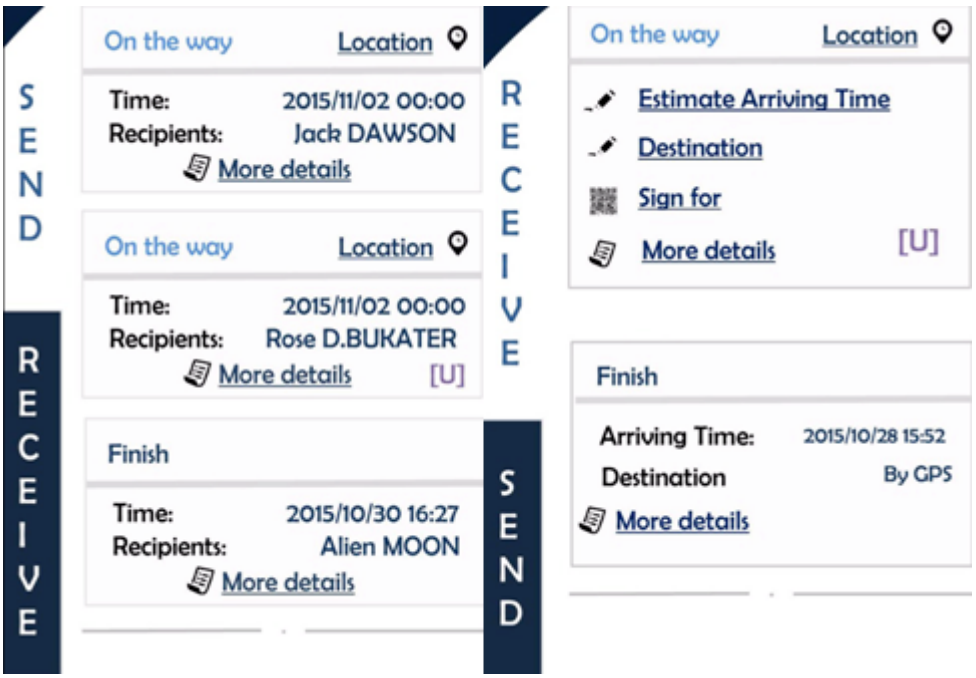


Fig.5.5. Sent and received orders

## Requirements confirmation

Based on the user requirements list, the app has fulfilled all mentioned requirements.

## 1. Visibility of system status

1.1. The app provides the user with a search tool (by name and by time).
1.2. The app has an adequate way of listing entries (orders). Each order is bordered and only the most important and relevant information is shown (Fig. 5.5).
1.3. All of the buttons follow the convention except one. The 'Log off' button (Fig. 5.1) is placed under the user's QR code and is plain text surrounded by square brackets. It looks like a clarification of the code above instead of a touchable area. **(2.3)**

## 2. Match between system and real world

2.6. When the user changes between ‘Send’ and ‘Receive’ (Fig. 5.5), the positions of these two buttons should remain unchanged, otherwise, the user may easily get confused. **(2.8)**

## 3. User control and freedom

3.4. Some screens do not have a 'Return' button. On the other hand, all of them have 'Home' button. **(2.0)**

## 4. Consistency and standards

4.4. In other screens, the ‘return back’ button in the right-bottom corner is used to return to the previous screen. However, in one screen (Fig. 5.3), this button returns the user to the home screen. **(2.3)**

## 5. Error prevention

5.2. Upon pressing 'Submit' an order (Fig. 5.4), all orders are submitted. However, there is a good chance that the user will only input the information of one order and tap on ‘Submit’ by habit. The app should inform the user if they have not checked the other orders. **(3.3)**

## 6. Recognition not recall

6.1. The app is consistent. If some task takes more than one screen/ step, there is a progress bar at top of the screen to keep the user informed.

## 7. Flexibility and efficiency of use

7.1. The app has a consistent navigation bar all throughout the app (Home button at top right corner).

## 8. Aesthetic and minimalist design

8.1. The interface is both efficient and pleasant to the eye. It provides the necessary color contrast and information layout to make it as easiest to use as possible. It also has a serious look that makes it trustworthy which is important when the nature of the app is considered.

## 2.6. Evaluation of Hongzhi Zhang's prototype

*Interface screenshots.*

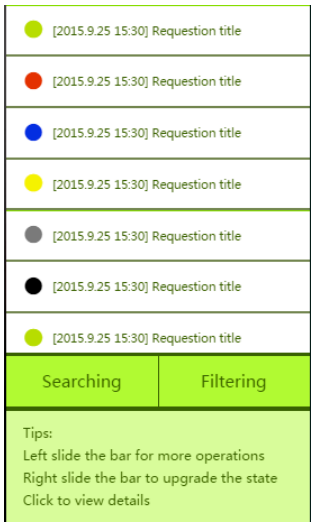


Fig.6.1. Requests List

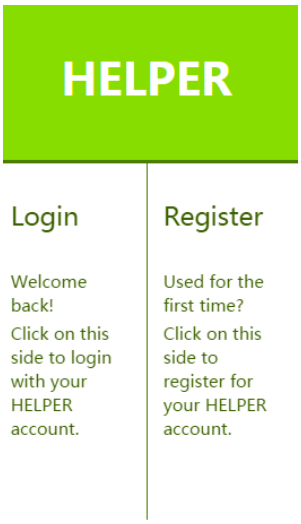


Fig.6.2. Login/ Register

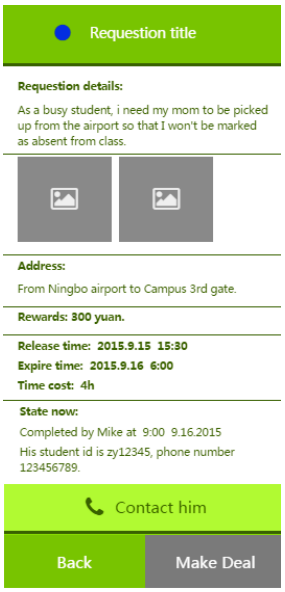


Fig.6.3. Request View


Fig.6.4. Request Making

### Requirements checking

Based on the user requirements list of this app, it does not fulfill some requirements shown below.

| Number | Requirement specification | Whether fulfilled | Explanation |
|---|---|---|---|
| 1.3 | The user profile is editable. | No | The app has completely overlooked this requirement. |
| 3.2 | Participants will be added to a willing list. | No | This app does not provide any list. |

### 1. Visibility of system status

1.1. The system allows the user to search the requests by keywords or filter them (Fig. 6.1).
1.2. The entries are not by default ordered in the most convenient way (completed task furthest down, on-going on top). There are no hints to remind the user of the meaning of the task's color (Fig.6.1). **(3.3)**
1.5. There are no headers to inform the users which screen are they are on. **(3.0)**

### 2. Match between system and real world

2.1. The app implements a rather complicated system for classifying the requests. It uses 6 colors and they are neither explained to the user nor reminded of their meaning. **(3.5)**

### 3. User control and freedom

3.1. Most of the time, the app does not provide the user with a "cancel" or "go back" option. **(3.6)**
3.4. The app does not have a main menu and it does not provide the user with a navigation bar which results in being stuck at one screen. **(3.5)**
3.5. The app does not have clearly indicated means of exiting. **(2.5)**
3.7. After the user request for a help, the app does not provide any means for the user to cancel this request. As a result, the user has no choice but to submit this request. **(4.0)**

### 4. Consistency and standards

4.2. The interface has one main color which is both fresh and welcoming. On the other hand, the colors used to classify requests are too many and too confusing. **(3.0)**
4.4. The style is consistent throughout the app but the layout is not. Sometimes the buttons providing the same functions are located at different places. **(2.2)**

### 5. Error prevention

5.3. The explanation of errors is rather ambiguous. It does not specify what is the error, but only says "Wrong information". **(3.0)**

### 7. Flexibility and efficiency of use

7.2. Whenever relevant, the system provides the user with a search box. There is no indication on how smart it is and whether or not saves search history. **(1.5)**
7.3. The system provides a tutorial which would be useful for new users.
7.5. The tips are provided all the time without taking into account the expertise of the user. The system should implement a function to turn them off. **(3.6)**

### 8. Aesthetic and minimalist design

8.1. The app has no icons at all. As a consequence, at times the page is clustered with text which is often unnecessary. Fig 6.3 and 6.4 show the screens for creating and viewing a request. It is all text based when icons for address and time could be used to reduce the text and possibly fit the operation in one screen since now, the user has to scroll down to see the end of it. Also, two conventions are used simultaneously to indicate mandatory fields when filling up a form (star sign and explanation in brackets, Fig 6.4.). Only one of them should be used. **(2.2)**
8.2. The app has an adequate contrast between its visual elements.

### 9. Help user recognize, diagnose and recover from errors

There is no indication whether or not the app provides appropriate error messages according to the error itself. It also does not provide a suggestion about the actions the user could take to recover. **(2.0)**

### 10. Help and documentation

10.3. The app provides perhaps too much help. It often shows tips to the user and while this is quite useful for beginners, it could get annoying after all the knowledge is gained. Also, there is explanation for options that don't necessarily need explanation (login and register, Fig 6.2). **(3.0)**

## 3. Summary of common problems

The table below summarizes the most frequently encountered usability issues among all of the prototypes and provides general solutions.

| Heuristics | Occurrences (out of six) | Comment | Solution |
|---|---|---|---|
| **1.4** | 4 | In most of apps, buttons are difficult to be distinguished from inactive content. | Use color contrast, shading or border to make them look touchable. |
| **2.1** | 4 | A significant number of metaphor designs are hard to be intuitively understood. | Provide some brief test description and conform to conventions. |
| **3.1** | 3 | Most of apps do not allow the user to navigate freely back and forth. | Provide return buttons, consistent navigation bars and cancel options. |
| **3.6** | 5 | Most of apps do not provide confirmation prompts before attempting high-risk tasks. | Implement adequate warnings and require confirmation from the user. |
| **4.2** | 4 | Colors are used inappropriately. | Use pleasurable color schemes. |
| **8.1** | 5 | The screens are often clustered and display unnecessary information. | Apply metaphor design to make screens more clean and efficient and divide information into appropriate chunks. |
| **10.3** | 6 | Most of apps lack necessary help documentation. | Provide tips, tutorials, user manuals or on-demand explanations. |

## 4. Group conclusion

As Nielsen (1992) points out, although one person could find many problems while evaluating, it is very unlikely to discover all of them. This is why group evaluation is useful. While some of the issues would be noticed by all of the experts, some will be noticed by very few of them. Therefore, having an appropriate number of evaluators is both efficient and productive.

Collaborating in a group was an interesting experience. We learned how to combine great amounts of individual work into one summarized structured product, how to communicate criticism and how to effectively manage our time.

## References

Budiu, R. and Nielsen, J. (2008). Usability of Mobile Websites: 85 Design Guidelines for Improving Access to Web-Based Content and Services Through Mobile Devices. Nielsen Norman Group.

Budiu, R. and Nielsen, J. (2011). Usability of iPad Apps and Websites. Nielsen Norman Group, 2nd edition.

Nielsen, J. and Monhlich, R. (1990) Heuristic evaluation of user interfaces. In Proceedings of CHI '90. ACM, New York.

Nielsen, J. (1992). Finding usability problems through heuristic evaluation. In Proceedings of CHI'92, 373-800.

Nielsen, J. (1994). Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. http://www.nngroup.com/articles/guerrilla-hci/.

Nielsen, J. (1995). Severity Ratings for Usability Problems. http://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/

Pierotti, D. (2005). Heuristic evaluation—a system checklist. Tech. Rep., Xerox Corporation, Society for Technical Communication.

Yong, G. J. (2006). A Usability Checklist for the Usability Evaluation of Mobile Phone User Interface. International Journal of Human–computer Interaction, 20(3), 207–231.

# Appendix A: Evaluating procedure and methodology

## 1. Evaluating procedure

### 1.1. Briefing and first hands-on session in class

We gathered as a group for the first time and we nominated Eason as the leader of our group. After achieving consensus on the project's goal and the basic strategy of heuristic evaluation, we decided the procedure of our group work.

During the first session, we worked in pairs which allowed us fast communication. That allowed us to investigate each other's prototypes from a high-level perspective. Each member explained the target users and the requirements of their own prototype. This is fairly crucial as the requirements and potential users of six prototypes vary from each other significantly. Without stating these clearly, the usability heuristic evaluation will lack user focus and be less effective.

New pairs were being set up until all prototypes were evaluated by all the members. Given the short length of the hands-on session, this procedure allowed us to be both fast and efficient.

This was the first pass through the prototypes and it was meant only to give us general idea of their essence and scope. We also decided on our heuristics list based on the 10 usability heuristics allocated by Nielsen (2014: useit.com).

At the end, our leader helped us collect all the mid-fidelity prototypes, individual and group reports, packed them and provided access to them to everyone.

### 1.2. Independent evaluation period

During this period, we evaluated all other five prototypes independently, using the decided heuristics as guidance. This second pass through the prototypes requires us to focus on specific interface elements in the context of the whole application and record any usability issues and the heuristic that has been violated. These evaluation notes are valuable in the debriefing session. The whole individual evaluation session lasted for 5 days.

### 1.3. Debriefing Session

We set up a group meeting to compare findings, prioritize problems and propose solutions. Firstly, we decided the structure of our group report and then, each of us became the host, introduced and demonstrated our own prototype to the rest of us. The rest of us asked questions, reported usability issues based on our notes made in the independent evaluation period and discussed at the same time. By interacting with the creator and reporting issues at the same time, each of us could get further understanding of each heuristic guideline and broaden thinking in what to evaluate so as to be prepared for the individual evaluation part. At last, we collected reported issues together and decided each of us to write a detailed evaluation of two prototypes so that each prototype has two evaluation reports.

### 1.4. Digging session

Each of us was required to thoroughly and systematically investigate the two given prototypes. We wrote all usability issues based on the debriefing session and investigated more potential issues after getting deeper understanding. This is the third pass through the prototypes and each of us was as specific as possible while evaluating. These individual evaluation documents can be found in Appendix D.

### 1.5. Merging and severity evaluation session

We merged and integrated our evaluation reports together and removed duplicated issues. According to Nielson (1995), it is hard to come up with good severity assessments from the evaluators during the heuristic evaluation session when they are more focused on finding new usability issues, so we made the severity evaluation at the final stage by sending a questionnaire to ourselves, evaluators, after the heuristics evaluation. The individual severity ratings can be found in Appendix D. The principle we adhered to when calculating severity can be found in Appendix C.

According to Nielson (1995), the quality of the mean severity rating increases rapidly, and using the mean of a set of ratings from three evaluators is satisfactory for many practical purposes, so we calculate the mean values for each issue to represent the importance of each individual problem and prioritize them. General recommendations and solutions are presented in the part 4. Finally, we concluded and reflected on our whole group heuristic evaluation process.

## 2. Methodology

### 2.1. Heuristic evaluation

According to Nielsen(1990), heuristic evaluation (HE) is an inspection method based on evaluation over real system or prototype, conducted by experts. In HE, experts check the accomplishment of a given heuristic checklist.

### 2.2. Generating a list of heuristic guidelines centered in mobile applications

First of all, we got a clear description of the scope of problems to categorize and classify features of mobile interaction. Then, we reorganized the 10 usability heuristics allocated by Nielsen (2014: useit.com) into a new compilation, as these heuristics are high-level design principles general to adapt to all kinds of design. By creating mobile-specific sub-heuristics for each ten heuristic, we managed to create a heuristic checklist specially designed for mobile applications based on findings of Nielsen(2011) ,Pierotti (2005) and Yong(2006) (Appendix B).

# Appendix B: Mobile-specific Heuristics guidelines.

**1. Visibility of system status**
1.1. If there is a list of elements, the user should be able to search them by keywords or filter them according relevant criteria in order to narrow down the number of elements that must be inspected.
1.2. The elements in the list should be sorted in an order that matches the needs of the task.
1.3. Keep users informed, appropriate feedback within reasonable time.
1.4. The buttons should be distinguish to look touchable.
1.5. Current status of an icon should be clearly indicated.
1.6. Keep users informed of actions or interpretations, changes of state or condition using clear, concise, and unambiguous language familiar to users.

**2. Match between system and real world**
2. 1. Appropriate use of metaphors: they should be easily understood.
2.2. The design of icons match cultural conventions.
2.3. Related and interdependent fields appear on the same screen.
2.4. Make simple, common tasks simple to do.
2.5. Make information appear in a natural and logical order.
2.6. The use of colors, layout and style conforms to the real world.

**3. User control and freedom**
3.1. Provide a way for users to leave unwanted state without extended interaction / effort.
3.2. Allow users control the system by their own decisions.
3.3. Provide the option to customize features.
3.4. Allow users to move forward and backward easily.
3.5. Clearly marked exits.
3.6. Prompt users to confirm commands that have significant or destructive consequences.
3.7. Allow users to undo or cancel the operation.

**4. Consistency and standards**
4.1. No more than twelve to twenty icon types.
4.2. No more than four different frequently used colors (additional colors for occasional use only).
4.3. Avoid a heavy use of uppercase letters.
4.4. Provide a consistent icon design scheme and stylistic treatment.

**5. Error prevention**
5.1. Menu choices are logical, distinctive, and mutually exclusive.
5.2. Inform users if they are going to make a possibly serious error.
5.3. The representation of errors must be clear to users.
5.4. Avoid crowding targets: placing targets too close can make users can easily hit the wrong one.

5.5. Avoid too small targets: research has shown that the best target size for widgets is 1 cm × 1 cm for touch devices.
5.6. Appropriately use of padding.

**6. Recognition not recall**

6.1. The user interface produces results that are in accord with previous commands and states (Cognition support Predictability).

6.2. The user interface conforms to some standard design patterns to be familiar to users.

**7. Flexibility and efficiency of use**

7.1. Provide navigational aids so that users could easily go back or navigate among interfaces.

7.2. Have a visible search box.

7.3. Easy to learn.

7.4. Be flexible as to adapt to various environments and users.

7.5. Be efficient to use so that once the user has learned the system, a high level of productivity is possible.

**8. Aesthetic and minimalist design**

8.1. Clean design that minimizes unnecessary information.

8.2. Clear contrast between visual elements, balanced layout and informative colors.

8.3. Avoid visual clutter.

**9. Help user recognize, diagnose and recover from errors**

9.1. Error messages should be expressed in plain language.

9.2. Error messages should be easy to understand.

9.3. Error messages should precisely indicate the problem, and constructively suggest a solution.

**10. Help and documentation**

10.1. Focus on one single feature at a time.

10.2. Present only those instructions that are necessary for the user to get started.

10.3 Provide necessary help and tutorial.

**11. Skills**

11.1 Correctly anticipate and prompt for the user's probable next activity.

**12. Pleasurable and respectful interaction**

12.1. Because users dislike typing, compute information for the users.

12.2. Use defaults that make sense to the user.

12.3. Use image thumbnails that are big enough for the user to tap.

12.4. The app should response based on the prediction of the user.

**13. Privacy**

13.1. When displaying the private information, the size should be small enough.

# Appendix C: Severity Ratings Strategy

Severity ratings are useful for prioritizing usability issues found in heuristic evaluation so that resources are distributed to the most serious problems.

There are three criteria when rating the severity:

- How often the problem occurs?
- How serious the problem is if it occurs?
- Is the problem a one-time problem or a continuous problem?

The following table shows our designed rating scale, which is used to rate the severity of usability problems:

| Usability issue severity scale | |
|---|---|
| **Rank** | **Description** |
| **0** | Purely a design or layout issue (Cosmetic issue) |
| **1** | A design issue that burdens the workflow |
| **2** | Minor usability problem: low priority of fixing |
| **3** | Major usability issue: high priority of fixing |
| **4** | Usability catastrophe, carrying safety or security risks |

# Appendix D: Individual evaluation and severity ratings

This appendix, accessed by the attachment with the report, contains all individual work of each group member during the group work. They are divided into six directories with the name of each of us.