OOP Project Report – Group 79

Leonardo Marcuzzi, Aistė Macijauskaitė, Aldas Lenkšas
Jan Maris, Rebecca Andrei

1 INTRODUCTION

A heuristic evaluation is an evaluation technique that assesses the usability of an interface based on a set of established principles or heuristics. The primary objective of this assessment is to identify potential usability issues that could impact user experience and to provide recommendations for improving the design of the interface.

The purpose of TALIO is to provide an efficient and effective way for individuals or teams to manage their tasks and projects. Therefore, the usability of the prototype is of utmost importance as it directly impacts its usefulness and adoption. A heuristic evaluation will involve examining the interface design, information architecture, and the interaction patterns that users have with the prototype. It will also take into account the established usability heuristics such as visibility of system status, consistency and standards, and error prevention and recovery, among others. The evaluation will identify areas where the application meets or falls short of these usability principles.

By performing a heuristic evaluation of TALIO, potential usability issues can be identified and addressed before they can negatively impact the user experience. This will result in a more efficient and effective interface, leading to increased user satisfaction, and ultimately the success of the prototype in achieving its purpose.

2 METHODS

2.1 Experts

The experts we have chosen to aid us in completing the heuristic usability evaluation were six other students in the Computer Science and Engineering bachelor program at TU Delft. Taking into account the fact that they have also been working on their own version of TALIO for some time, it would be fair to conclude that they have full knowledge of the scope of the application, its libraries, and APIs. Furthermore, by ensuring that some of the experts were more practiced in the back-end side of the application and others in the UI and front-end, we received detailed feedback on every part of our product.

2.2 Procedure

We used a cognitive walkthrough model to test the usability of our project. The experts received a working prototype of our application and were asked to go through several scenarios, provided below, whilst writing down their thoughts. This evaluation would examine the steps a new user would take to complete these tasks and identify any potential issues that may hinder their progress. For instance, they assessed the layout and labeling of the interface, the clarity of instructions, and any potential confusion that may arise during the completion of these tasks.

This evaluation would also consider the user's mental models and expectations, i.e., how the user expects the website to behave based on their prior experience with similar systems. This would involve examining the consistency of the interface design and the navigation system with common user expectations. The cognitive walkthrough would also consider potential errors that a user may make during the task completion process and assess the prototype's ability to recover from such errors. The experts would identify any error messages that may be confusing or unhelpful to the user, and suggest ways to improve the recovery process.

Overall, a cognitive walkthrough of TALIO provided us with insight into how new users interact with the prototype and the potential barriers they face when completing tasks. This evaluation helped improve our project's usability and increase user satisfaction.

Below are the five main tasks that a user may use the application to complete:

(1) Sign up / Log in.
(2) Enter a board with a server URL.
(3) Add, remove, and edit lists within this board.
(4) Add, remove, move, and edit cards within a list.
(5) Work on the same board with multiple clients.

Evaluators assessed these five key tasks that users typically perform on the TALIO application. The tasks were then evaluated by using Jakob Nielsen's ten heuristics for user interface design, with a focus on assessing the processes involved from a user's perspective:

1. Visibility of system status: The system should always keep users informed about what is going on, through appropriate feedback within a reasonable time.

2. Match between system and the real world: The system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom: Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards: Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention: Even better than good error messages is a careful design that prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit

to the action.

6. Recognition rather than recall: Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use: Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design: Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation: Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.3 Measures (Data Collection)

Our primary goal throughout the evaluation was to measure how client-friendly the prototype was. Additionally, since we allowed our evaluators full freedom over the application, we were interested to see if they could identify design bugs that had previously escaped our notice. Thus, we were looking to our evaluators to report both issues with the application itself, as well as suggestions to aid in making the interface more user-friendly, and overall improving the GUI.

Our methods of collecting the data were simple, but effective - whilst using the application, the experts remarked upon everything that they felt could be even slightly improved. As such, the data from the evaluation was presented in the form of a text file consisting of notes which highlighted specific issues that the experts observed throughout their exploration of the app. For instance, when the evaluators attempted to enlarge the application to full screen, they noticed that it did not scale properly. As a result, they noted that "the application has a non-responsive design on full-screen". All of the data we collected follows the same format as this example, both for describing issues that have to do with usability and also for various UI design changes.

3 RESULTS

To further examine each of the problems, we estimated its severity in terms of usability principles. We looked at two factors to determine the severity - frequency and impact. The impact is considered high when it hinders users from utilizing the website and, therefore, is difficult to overcome. The severity of frequency is high if the problem occurs frequently within the application, causing inconvenience to the user. In a four-quadrant priority matrix, the problem could fall into one of the four categories:

- High impact and high frequency
- High impact and low frequency
- Low impact and high frequency
- Low impact and low frequency

By using a four-quadrant priority matrix, we can prioritize the problems based on their severity and determine which ones require immediate attention. This approach helps to focus our efforts on the most critical issues and make the most efficient use of the resources.

Here is the list of problems, each problem is linked with one or more heuristics that it violates, and also has a prioritization level assigned to it.

- When opening multiple clients, making changes on one board do not appear automatically on the other.
  - 1: Visibility of system status
  - High impact and high frequency
- Non-responsive design on full screen.
  - 4: Consistency and standards
  - High impact and high frequency
- Non-consistent UI (e.g. adding a card list and a card looks very different).
  - 4: Consistency and standards
  - High impact and high frequency
- User cannot drag lists.
  - 3: User control and freedom
  - High impact and high frequency
- Delete icon is almost not visible.
  - 1: Visibility of system status
  - Low impact and high frequency
- When entering an invalid URL address and then entering a valid one, the error message stays.
  - 1: Visibility of system status
  - High impact and low frequency
- User cannot add a card with an empty name (but it doesn't say that anywhere).
  - 5: Error prevention
  - High impact and low frequency
- The buttons do not feel like buttons(there is no visual indication).
  - 1: Visibility of system status, 8: Aesthetic and minimalist design
  - Low impact and high frequency
- Application doesn't ask the user for confirmation before deletion.
  - 3: User control and freedom
  - Low impact and high frequency
- Card list name letters are too big.
  - 8: Aesthetic and minimalist design
  - Low impact and high frequency
- If a very long name is entered for a card list, it doesn't show the full name. It's confusing.
  - 2: Match between system and the real world

  - Low impact and low frequency
- When a user wants to move a card by dragging a list to the end, it would only go to the end of the frame of the window.
  - 2: Match between system and the real world
  - Low impact and low frequency

4 CONCLUSION AND IMPROVEMENTS

The main conclusion we have drawn from this particular evaluation is that our product is far from complete, which is to be expected at this point in time. Past this, however, the results indicated that we were lacking primarily in providing a user-friendly experience and having an intuitive and aesthetically pleasing UI.

Based on the specific feedback we received and the data retrieved from the evaluation, numerous improvements will have to be made to the current interface. Firstly, the UI will be augmented so as to be more intuitive, by implementing measures such as providing a visual aid for the user when interacting with the buttons within the application. They should be very visible and their purpose should be obvious at first sight, playing into the user's expectations of what they are supposed to look like according to their previous experience in interacting with buttons of a similar nature.

Additionally, one of our top priorities is ensuring that our design choices are consistent throughout the entire application, so there is never any doubt in the user's mind as to how to navigate the prototype. Moreover, we are also interested in developing our application so as to be able to handle anything the user does. For instance, we will implement a responsive design throughout our interface so that the content of our application will automatically fit on a user's screen, regardless of whether they are in full-screen or windowed mode, and regardless of their monitor dimensions. We also plan on improving user freedom by expanding the set of actions that they can take within the prototype, such as allowing them to drag lists and fixing an issue where cards are not able to scroll with the application, preventing them from being dragged to a list off-screen.

Finally, we will continue working on improving and expanding the back-end: adding more features, and implementing Web Sockets (which were not functioning at the time the heuristic usability evaluation was conducted). Overall, our improved version will be aiming to fix all the issues that were discovered during the evaluation, and thus provide the user with a much more intuitive application with an aesthetically pleasing design.