Project Report – Group 64

Ioana Forfota, Emilia Ketterer, Tudor Măgirescu, Lazar Polovina
Dimitar Smenovski, Ziliang Zhang

ABSTRACT

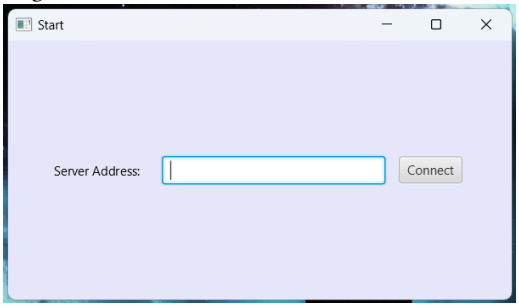
Over the short course of this project, our team has attempted to develop an application allowing users to better organize themselves. A heuristic usability evaluation is "a usability engineering method for finding the usability problems in a user interface design so that they can be attended to as part of an iterative design process" [1]. Conducting such an evaluation on our application is vital in the early phase of the project, as the results allow improvements to be made to the application, developing the prototype. Through obtaining feedback from experts, our team was able to find multiple areas of improvement, ensuring a better experience for the client when using our final graphical user interface (GUI).

1 INTRODUCTION

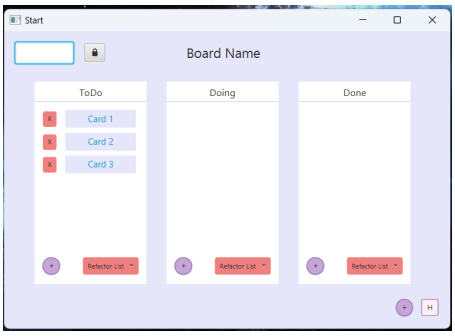
As stated, the aim of this project is to develop an application allowing a user to better organize themselves, specifically by keeping track of thoughts and to do's in the form of cards. During the planning process we created multiple mock ups of the application's GUI. While creating these we attempted to focus on minimal designs, clearly displaying the main features of the application and communicating their purpose through stylistic choices. We also wanted to clearly split the different scenes of the application by changing their default color, ensuring that there would be a separation between the main scenes and more specific features. To test whether we achieved all of these goals and find ways to further improve our application, our team completed this heuristic usability evaluation.

1.1 Prototype

When running the minimal application, the user first sees the following scene:

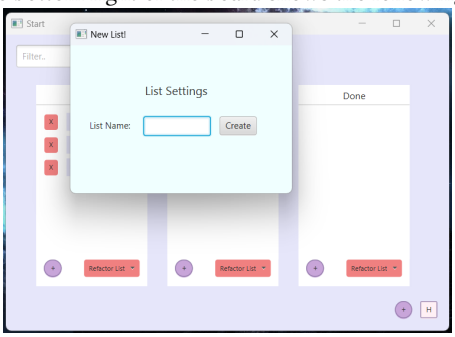


Here, they can input the server address they wish to connect to. If this server address does not exist, the user will see the message 'invalid' in the textbox. Once a valid address is input and "Connect" is clicked, the user sees the following scene:

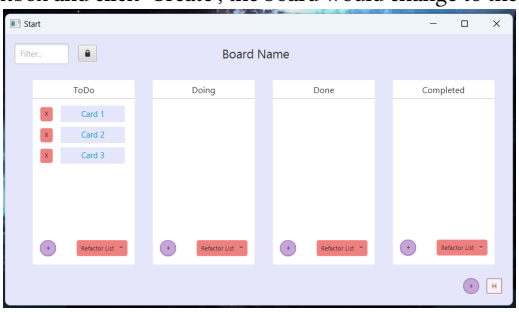


This main scene displays a board, which the user can interact with in multiple ways.

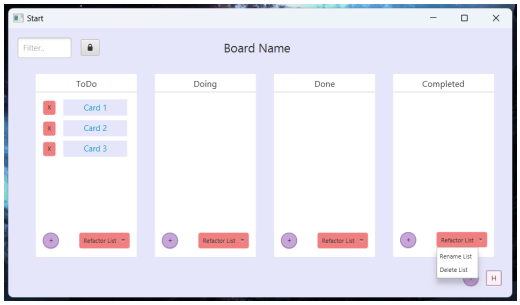
1.1.1 Creating a new list. Clicking on the circular purple button in the bottom right of the board shows the following scene:



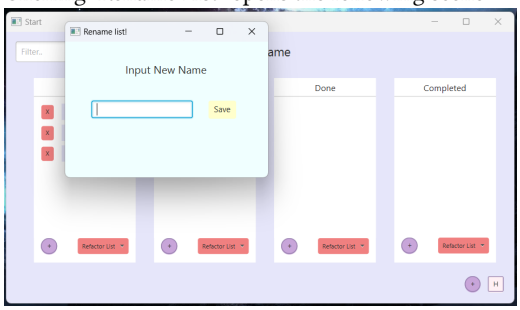
Here, the user can specify the name of the new list. Once the 'Create' button is clicked, this list is created and appended to the other lists. For example, if the user were to type 'Completed' into the textbox and click 'Create', the board would change to the following:



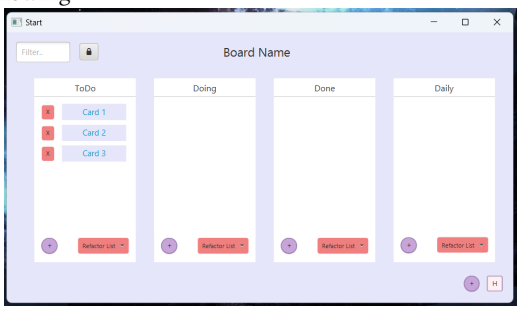
1.1.2 Refactoring a list. In addition to adding new lists, existing lists can be refactored using the 'Refactor List' button in the bottom right corner of each list. Clicking this button would present the user with the following options:



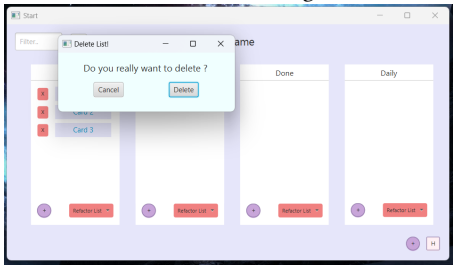
Clicking 'Rename List' opens the following scene:



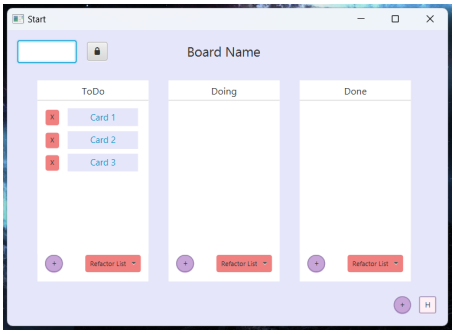
In this scene, the user can input the new name for the list. For example, if the user wanted to rename the list to 'Daily', they could type this and click 'Save'. This would change the board to the following:



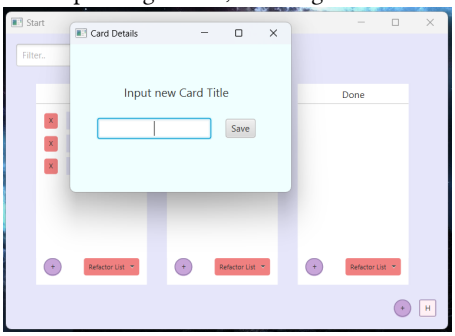
If the user now wished to delete the 'Daily' list, they could click on 'Refactor List' again and then click 'Delete List'. This would cause the user to see the following confirmation scene:



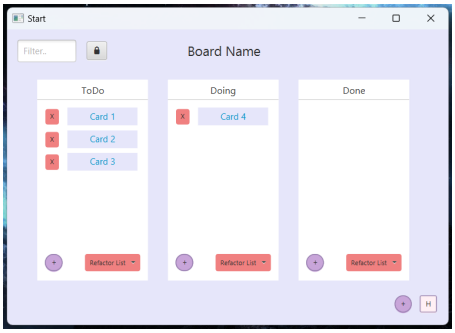
Clicking cancel would mean no changes occur. However, if the user clicks 'Delete', the board reverts to the following state:



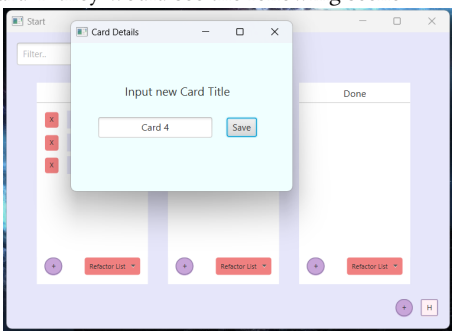
1.1.3 Adding a card. In any of the existing lists, a new card can be added. For this, the user must click on the purple button in the bottom left of the list they wish to add a card to. For example, if the user wishes to add a new card to the 'Doing' list, they could click the corresponding button, resulting in the following scene:



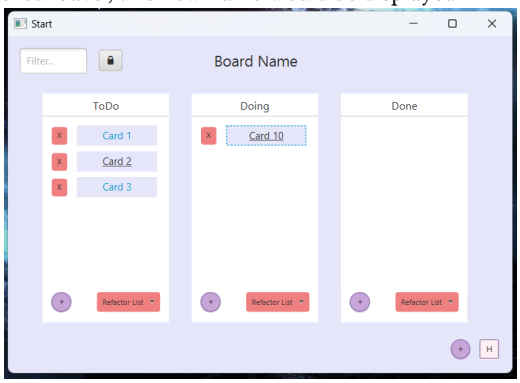
Here, the user can input the name for a new card. If they type 'Card 4' and click save, the user would see the following board:



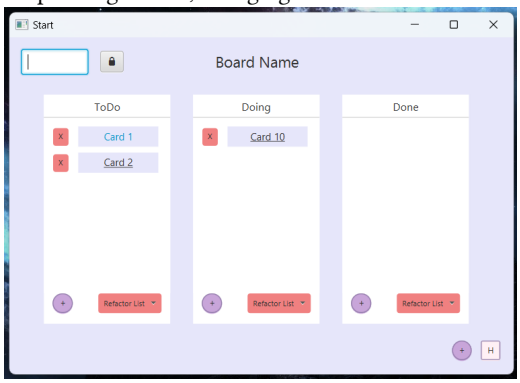
1.1.4 Renaming a card. If the user wanted to rename a card, they could simply click on it. For example, if the user wanted to rename 'Card 4' they would see the following scene:



If they changed the text inside the text field to 'Card 10' and clicked 'Save', this new name would be displayed:



1.1.5 Deleting a card. If the user wanted to delete a card, they could do so by clicking the red 'X' button next to the card. For example, if the user wanted to delete 'Card 3', they could click the corresponding button, changing the board to the following:



2 METHODS

To complete this heuristic usability evaluation, we chose another team of six people as our evaluators. Since they are also a team creating a similar application with the same goals, we hope that they will be well informed on what features the application should include, allowing them to judge their implementation. In case anything would still be unclear, we had one of our teammates act as an assistant and observer, ensuring the other team was following the procedure and helping them if they had questions. While this observer helped the experts and gave hints, we made sure the comments remained objective as to not influence their feedback.

For the evaluation, we sent these experts our prototype, as included above, over email and asked them to go through it individually at least once. While doing this, we asked them to note down any problems they noticed about the design specifically related to the following heuristics we provided them in the same email.

(1) Visibility of system status
(2) Match between system and real world
(3) User control and freedom
(4) Consistency and standards
(5) Error prevention
(6) Recognition rather than recall
(7) Flexibility and efficiency of use
(8) Aesthetic and minimalist design
(9) Help users recognize, diagnose and recover from errors
(10) Help and documentation

They were asked to note down these problems in a given Google Drive document using the following formatting:

(1) Description of the problem
(2) Possible cause of the problem
(3) Likelihood of the problem to occur
(4) Possible solution to the problem

This formatting should allow us to easily follow our experts' thoughts, see where there is overlap and analyze the results we obtained on how usable our interface is.

3 RESULTS

We received quite a long list of problems from our team of experts once they reviewed our prototype. To begin processing these results, we split the problems into two categories, a list addressing issues we had already planned on improving in our design when going beyond the minimal application, and a list of problems we had not previously considered. We decided not to include the first list of problems in our analysis of the feedback since we had already considered them. To summarize these problems shortly:

(1) "There is no disconnect button, preventing a user from leaving a server." This button will be added in sprint two.
(2) "There is no explanation for the buttons on the board (like "H", "lock", "filter")." We will add documentation under the "H" button - which we will rename to "help" for clarity - specifying the actions linked to each button.
(3) "Fast paced workflow is not possible because there are no keyboard shortcuts." We plan on adding these to the application once more pressing features have been implemented.

After filtering out these issues and grouping the remaining ones, we were left with 9 problems relating to our minimal application that we had not previously considered. Summarizing these problems with their description, related heuristic and related element in the GUI:

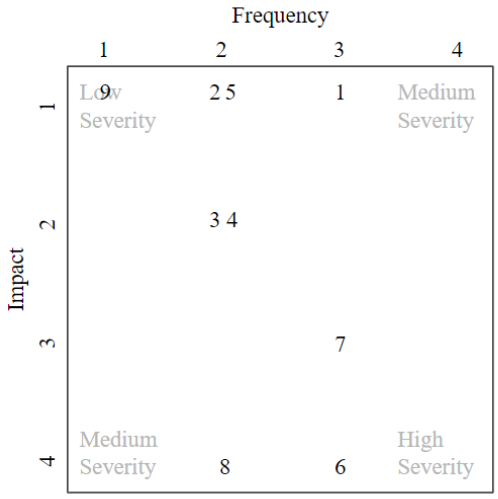
(1) Visibility of System Status: There is no clear indication that a connection was successfully made when connect is clicked in the first scene.
(2) Visibility of system status: There is no clear indication that a list was stored and saved besides the list appearing on the GUI.
(3) Match between system and the real world: The "+" button for adding a list is not intuitive enough to signify "adding new list".
(4) Match between system and the real world: The naming of the "Refactor List" button may be ambiguous for some users.
(5) User control and freedom: There is no undo feature on the GUI for deleted cards/lists.
(6) Consistency and standards: When users want to delete a card by clicking the corresponding "X" button, no confirmation scene appears like for lists.
(7) Error prevention: When a user creates a list by clicking the respective "+" button, or a new card by clicking the

corresponding other "+" button, they can leave the new name empty without any error/warning.

(8) Recognition rather than recall: There is no text/button that allows the user to see which server they are connected to.

(9) Help users recognize, diagnose, and recover from errors: There is no error message that appears when an error occurs such as duplicate names of cards and lists.

We then organized these by priority, based on our and the experts opinion on the likelihood a problem would occur, and based on what problems we deemed most severe in impact for a client (represented in the graph below).

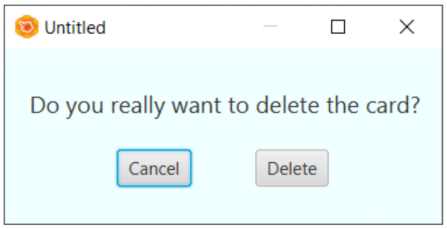


4 CONCLUSIONS AND IMPROVEMENTS

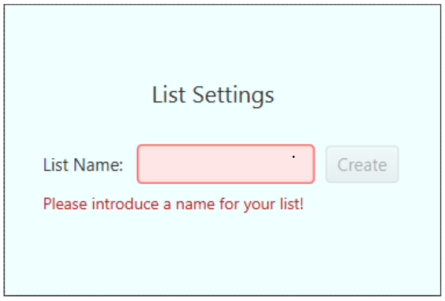
4.1 IMPROVEMENTS

Based on this feedback, there are some changes that we would like to make to the application, explained in approximate order of priority from the graph. These should improve the general experience for the user.

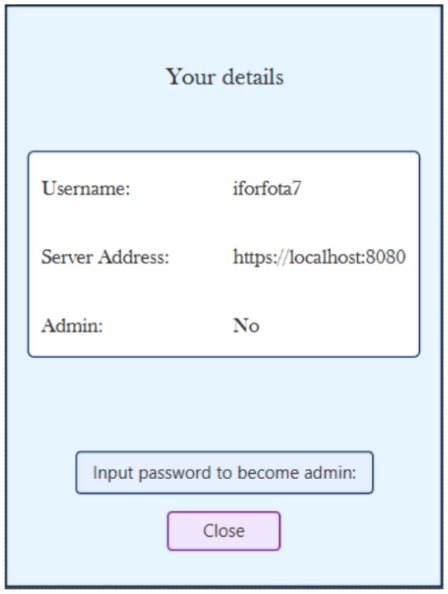
The first problem we wish to address is problem 6 regarding not having a warning when deleting a card. This is a crucial oversight on our behalf, and should be implemented to ensure the user does not delete a card without a simple way to recover from this error. This becomes especially important when more information can be added to a card, making it difficult to simply recreate it. Therefore, even though it might slow down the workflow of the application, we deem this a very important problem that we will address. For this, we will add the following scene to our application:



The second problem we wish to focus on is 7, the ability of a user to create a list/card without a name. As pointed out by the experts, it makes little sense for a user to be able to create such a list or card, as it would serve no purpose. A simple way to solve this problem would be to prevent the user from being able to click the "save" button when creating or renaming a new card or list if the textbox for the name is empty. In this case, the user could be informed by a warning message that appears below the text box. Through this simple change displayed below a lot of future errors and confusion could be avoided.

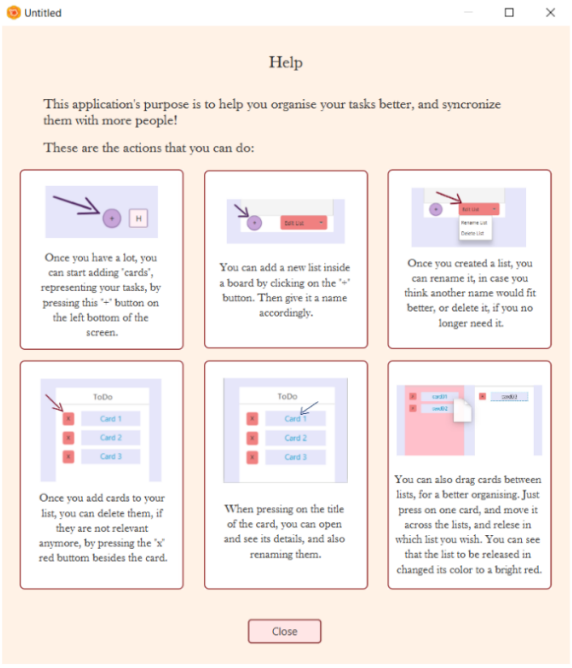


Currently, the user cannot see what server they are connected to (problem 8). This has to be remembered by the user while they are running the application. Having the server address displayed somewhere on the board would spare the user the difficulty of remembering it, making it more pleasant to use the application. To maintain a minimalist design, it could be beneficial to add a small button with an information icon in the corner of the board, which once clicked opens a new scene with the server information. Additionally, this scene could also display the username, whether the user is an admin, and present the option to become an admin.



Regarding the third problem, being the unclear naming of the add list button, we would like to consider multiple options. As stated, the current button is not clear in its labeling, leading to confusion for the user. However, we would like to maintain a minimal design and therefore not change the button to have a name such

as “add list”, since this would take away from the current aesthetic. Another solution would be to explain the use of the button in the “help” scene, which we already plan on implementing and can be accessed directly through a button named “help” (currently “H” in the prototype). The help scene might look as follows.



Fourth on the list of problems is that the word “refactor” used for lists might be ambiguous and too technical for a general user. This is a very good consideration we missed, and we will therefore change the button to “edit list” instead, making it less technical and clearer.

As the experts pointed out, there is no clear indication that tells the user that they have been successfully connected to a server except for the board overview scene appearing (problem 1). We thought that the board overview becoming accessible would be enough of an indicator to the users that they have been successfully connected, and we will likely not include a more direct message since working on other features appears more pressing to us.

The fifth problem listed states that there is no undo feature for deleting a card or list. This is true, however we did not deem this necessary since there is a warning upon deletion that ensures users cannot accidentally delete cards or lists. Therefore, we will likely not make any changes to the application based on this problem.

The second problem listed by the experts was that the user has no indication whether a list has been successfully stored. While we acknowledge that there is currently no specific feature to inform the user that a list has been stored, we believe that this is quite intuitive for the user since the list appears on the scene once it has been created. We will therefore not make any changes to address this issue in the near future.

The ninth problem, regarding the detection and prevention of potential errors such as duplicate names of a card/list, is one we had not considered. This was mainly because, on the database side, these duplicates would have no effect since each list and card has its own generated unique id. The experts are right that this could lead to confusion for the user, since they might accidentally add duplicates and then no longer be able to tell them apart. However, we think this is a minor inconvenience, and not even necessarily an error on the users behalf, so we will not implement any changes regarding this issue.

4.2 CONCLUSION

By getting feedback from experts and obtaining a list of problems with our GUI, we have managed to analyze our application in more detail. While going through the list of problems, our team was able to find solutions to challenges we hadn’t even previously considered, ensuring we could present our users with the best possible interface. We have come to the conclusion that our application needs a new scene confirming the deletion of a card, a system to check that no lists/cards are created without a name, a more general information scene with the server address, a more detailed help scene, and a more clearly labeled “edit” button for lists. By implementing these changes (mocks shown in section 4.1), we can ensure that our GUI will more closely adhere to the listed heuristics, and consequently provide the user with a better experience when using our application.

REFERENCES

[1] Jakob Nielsen. 1994. How to Conduct a Heuristic Evaluation. In *NN/g Nielsen Norman Group.*