TeamName

Michael Theisen, Patrick Tibbals, Jasharn Thiara, Trevor Tomlin

https://teamname-tcss360.github.io/

teamname.tcss360@gmail.com

Table of Contents

Summary…….……..2-3

Requirements Definition / Functional Specification…….…...4-5

Story Board / Paper Prototypes……..6-13

Requirements Specification / Technical Specification…….….14

Summary

The group name is TeamName. Our shared vision for this project would be a self contained file finder window that will be tailored towards storage of solely text documents, pictures/drawings. This application will function as a digital file folder to store customer purchase documents to allow combined location for all associated files of all file types.

Personas:

1. Jeffrey Weiss is a software developer working on behalf of his development team. After receiving a contract request by Bob Keener, retired engineer and DIYer, he shows that he is keen on asking the correct questions to ensure that he and his team are able to develop an app that fits the specifications that Bob has in mind. Luckily Jeffrey is able to empathize with Bob's situation, leading to him asking some very good questions during the interview. Jefferey moves from question to question while allowing Bob to elaborate on a variety of scenarios that allow Jeffery to grasp a bigger picture on what Bob needs. In the process, Jefferey's questions move from what Bob wants in an app toward how Bob would like to see the app function and ties up many loose ends by asking Bob if there are any additional features he would like to see. Jeffery shows that he can ask good questions while still acknowledging the data he will confer to his team. In ending the interview, Jeffery shows that he has genuine enthusiasm toward dealing with this project and promises to continue to ensure that Bob will be able to continue watching the progress that the team makes.

2. Bob Keener is the potential product owner and happens to be a retired electrical engineer. After getting fed up with an excess of outdated manuals and broken URL links to old products, Bob launched an inquiry to a software development contract to Jefferey Weiss and his development team. Bob's frustration leads him to wanting and organizer for all his information that he has to keep track of regarding major appliances and home systems. As a retired engineer, Bob likes to keep his own notes on these appliances that have his own with measurements, drawings, and plans. His filing cabinet is substantially full and still holds old documents pertaining to appliances he no longer has. Bob's feelings and situation leading to a firm understanding between the two parties and allows Bob to convey his needs more accurately.

Scenarios:

1. Bob installs laminate flooring in the family room. He wants to keep the drawings he made, the manufacturer's information, and the warranty in one place, but he feels silly printing this information and storing it in a folder.

2. Bob purchased an oven range for the kitchen. The warranty, receipts, and manuals are scattered between paper and the phone app so he wants to have one place where he can keep all of this information organized and easy to find.

3. Bob feels frustrated when trying to search through an abundance of outdated information. He cant seem to find a way to remove all the unwanted clutter efficiently, there always seems to be leftover files.

4. Bob feels overwhelmed with remembering all of the expiration dates to his warranties. Often times he forgots were he placed his paperwork and misses out on a last minute servicing before experations.

5. Bob wants friends and family to be able to access certain files, but not have the ability to edit them. Like the instructions for using netflix on the guest TV allowing them access would make his life simpler and save paper/time.

6. As an engineer Bob will be storing many types of files into the program. With the countless applications used for modeling. It is imperative the program handle all possible file types.

7. Bob wants to be able to save a file into his Oven Range Folder within the program. To do this, he will want to use the drag and drop feature. After doing so and searching for the file using the Search bar, he finds that the file that was dropped into the Oven Range Folder was actually a PDF for the Flooring Warrenty. Bob wants to remove this file from the incorrect folder so he deletes it using the built in delete function, thus sending the file to the Deleted folder.

8. Bob realizes that he deleted the wrong file last week he goes into the Deleted folder and right clicks and clicks restore which then repopulates the file back to its original destination. If said destination no longer exists a folder will be auto generated and placed within the parent folder of the original file location.

Requirements Definition / Functional Specification

User Stories:

US01: As a user, I want to be able to store documents.
US02: As a user, I want to be able to use a keyword search.
US03: As a user, I want to be able to delete old info.
US04: As a user, I want to be able to edit documents.
US05: As a DIYer, I want to be able to keep track of measurements alongside documentation.
US06: As a user, I want to be able to access to all drives
US07: As a user, I want to have the ability to export data.
US08: As a user, I want to be able to upload my data to the cloud.
US09: As a user I want to sign in to a profile.
US10: As a user, I want to be able to control who has access to the info.
US11: As a user, I want to be able to share documents.
US12: As a user, I want to be able to sort the documents.
US13: As a user, I want to have navigation buttons to move through folders efficiently.
US14: As a user, I want to use the app offline.
US15: As a user, I want to have a service reminder.
US16: As a DIYer, I want to be able to edit the code.
US17: As a developer I want to have a user suggestion feature
US18: As a developer I want to be able to use a bug reporting feature.
US19: As a developer I want access to copies of the current code

Business Rules:

BR01: The user must have access to combine online and hard copy information into one location
BR02: The user must have access to all files
BR03: The user must have access with or without internet
BR04: The user must be able to edit/move/delete files
BR05: The user must be able to share files with other users
BR06: The user must have the ability to search by keyword
BR07: The user will have a calendar to track repairs/maintenance needed for products
BR08: The user must be able to give desired permissions to other users
BR09: The user must be able to find all necessary functions easily

Non-Functional Requirements:

- Capacity/Scalability
- Product will not require any sensitive personal data to run
- Work on or offline
- Storage will be similar to the finder window though subfolders will likely be auto generated within each.

Paper Prototype

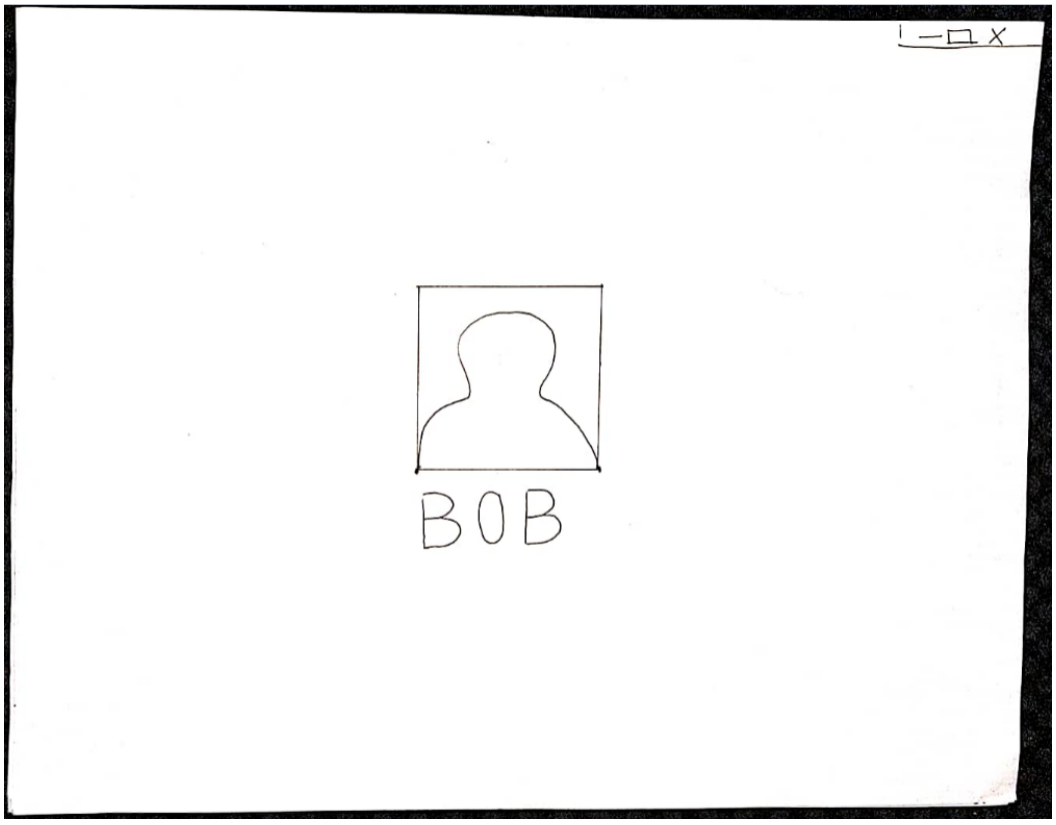
For the Paper Prototype, we decided to start small with some of the more simplistic processes that would be used such as navigating the Graphical User Interface. To do this, we implemented three primary functions to demonstrate the implementation as well as a Sign-In feature. We began with a simple method of adding a file to the primary window using a drag and drop feature. We then had the user search for the particular file and finally delete that same file. This implementation was done so with the forethought that the user would be someone who was familiar with a general Windows or Mac user interface that could manipulate input and output using the mouse and keyboard.

Functionalities:

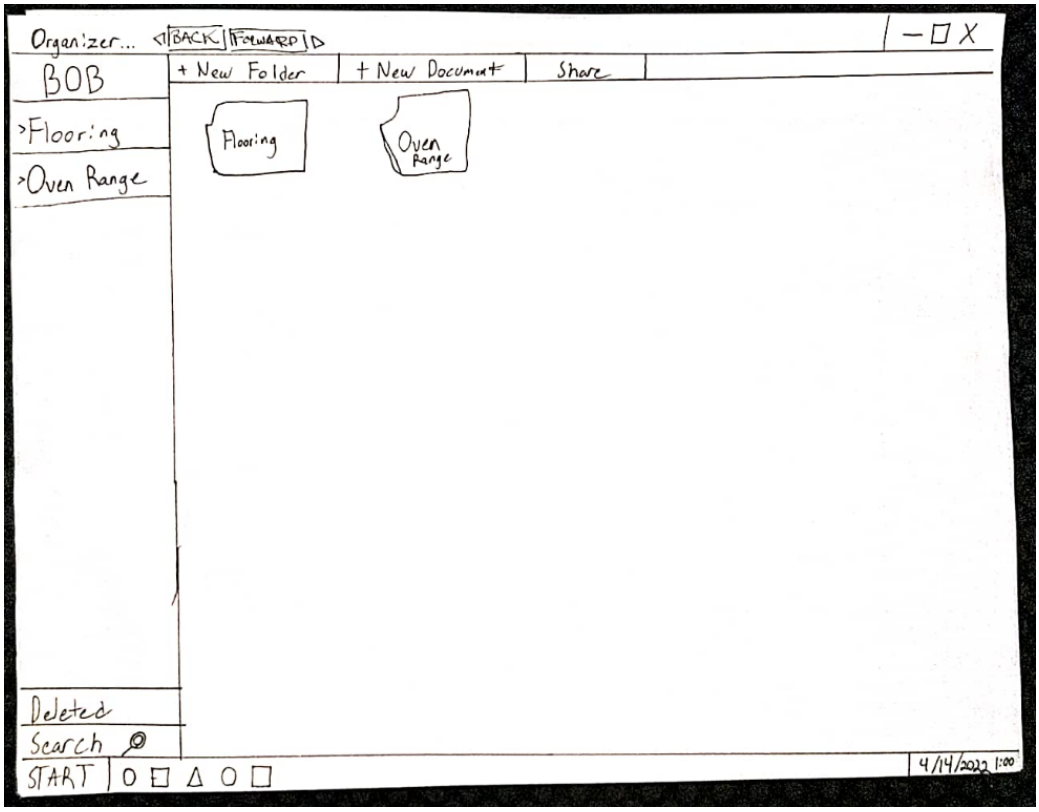
Sign-In

- Click on profile icon.

In order for a user to sign-in to the program, They would need to click on the image of the person that also had their name written underneath. This is done to show that there can be more profiles added later to assist with file sharing compartmentalization.



After clicking on Bob's profile picture, the user would then have this screen come up as the main screen for the Organizer program. From this screen, they will be able to click on any folder, add new folders, open the deleted section, search for files or folders, share files, and exit the program.

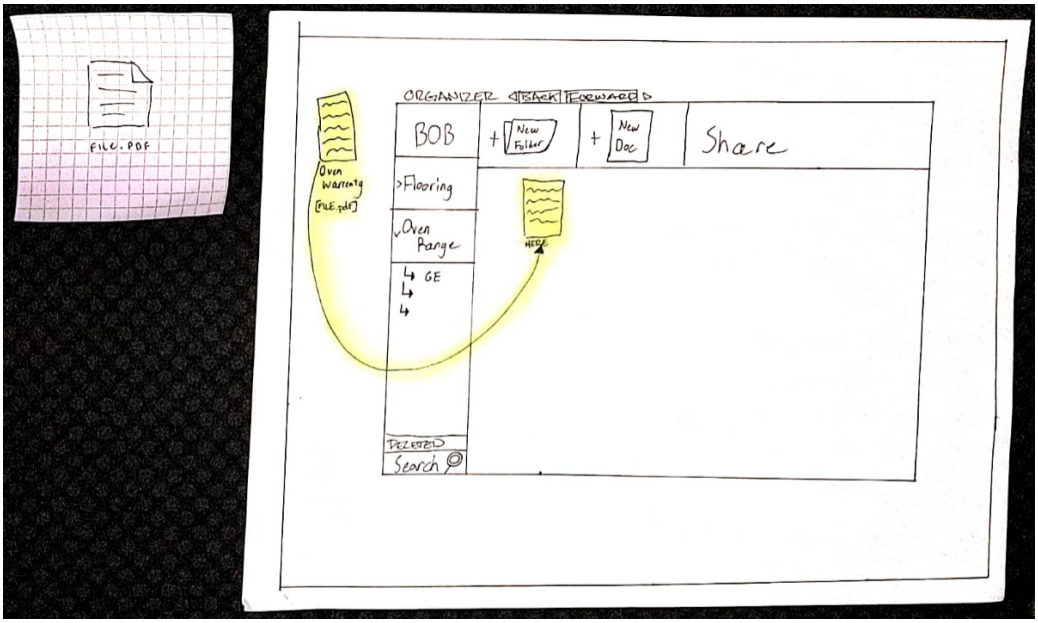


After seeing this screen, the user would then click on the Oven Range folder to take them to the next process that includes a highlighted visual of their next task.

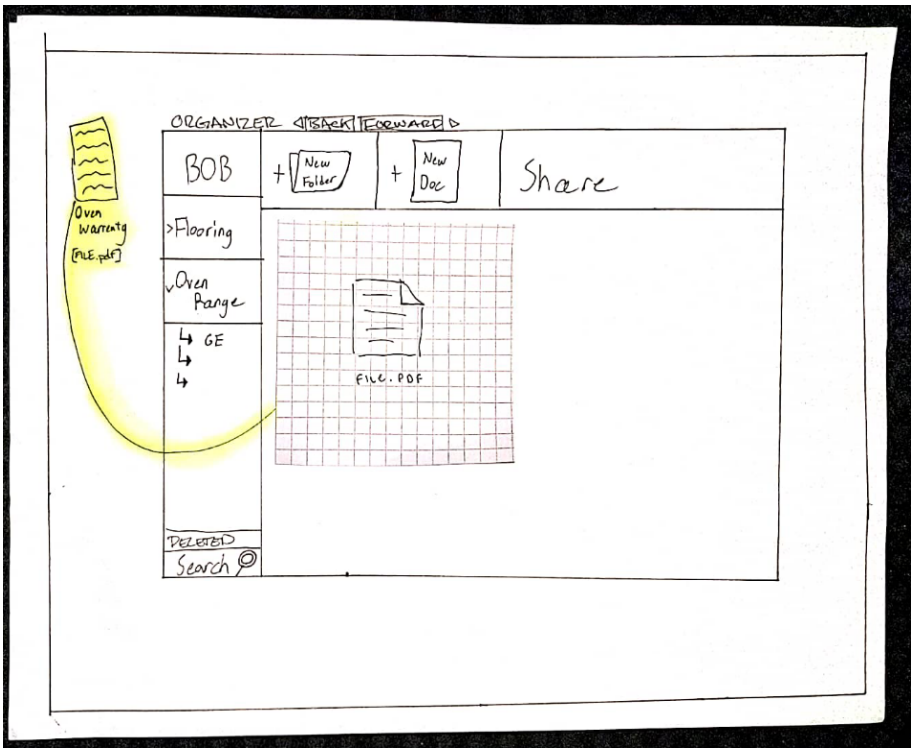
Add

- Drag and drop.

In order for the user to complete this task, they would need to click on a file that existed outside the window that was on the Desktop and drag that file into the primary file location without letting go of the mouse click until the file was in the correct location. After letting go of the mouse click, the file would then be moved to the new location. For the purposes of the paper prototype, we used a sticky note on the table as the file that was to be added into the Organizer. Since the folder for Oven Range was clicked on using the user's finger as a mouse clicker, the Oven Range folder would be presented with a highlighted visual of the intended task.



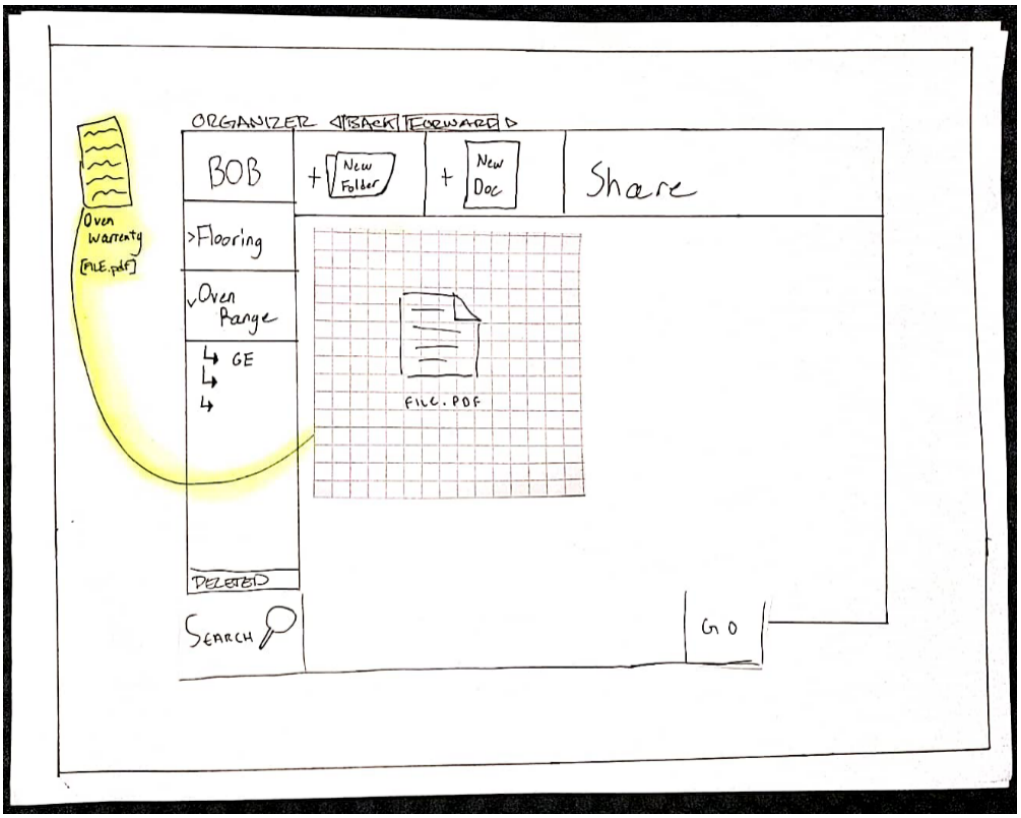
As can be seen from the highlighted visual, the intent was to have the user drag and drop the File.pdf from the outside desktop into the program's Oven Range folder. In hindsight, it might have been better to simply ask the user to complete this task without having the process described blatantly. This may be incorporated in the second iteration of the paper prototype. After the user manages to do this task successfully, the Oven Range folder will look like this:



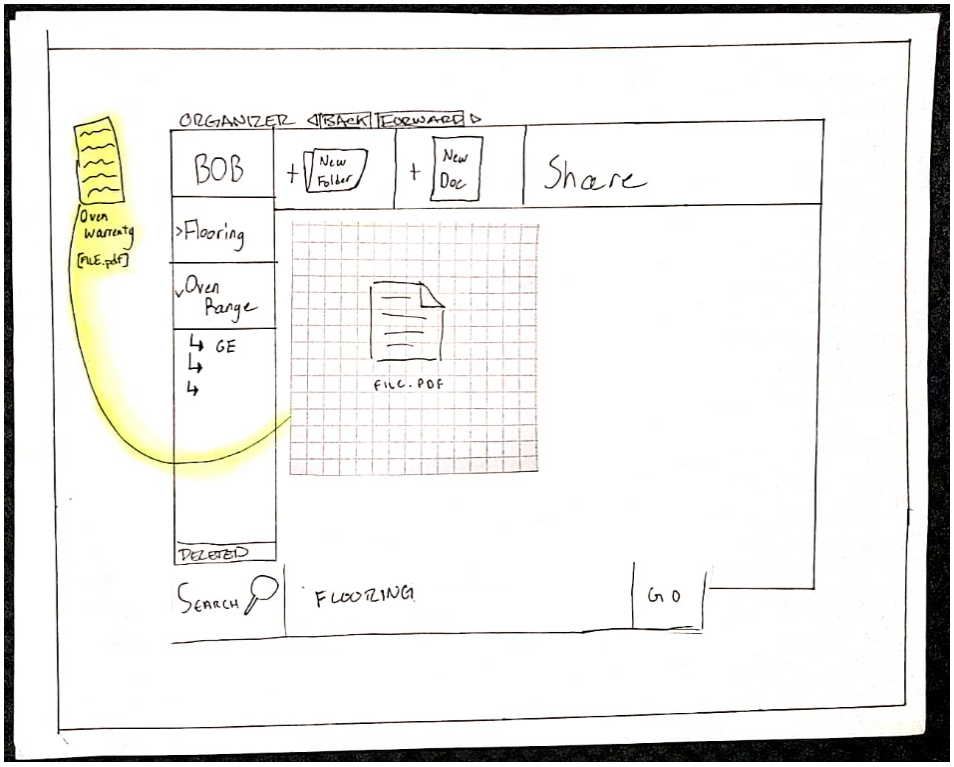
Search

- Via keyword

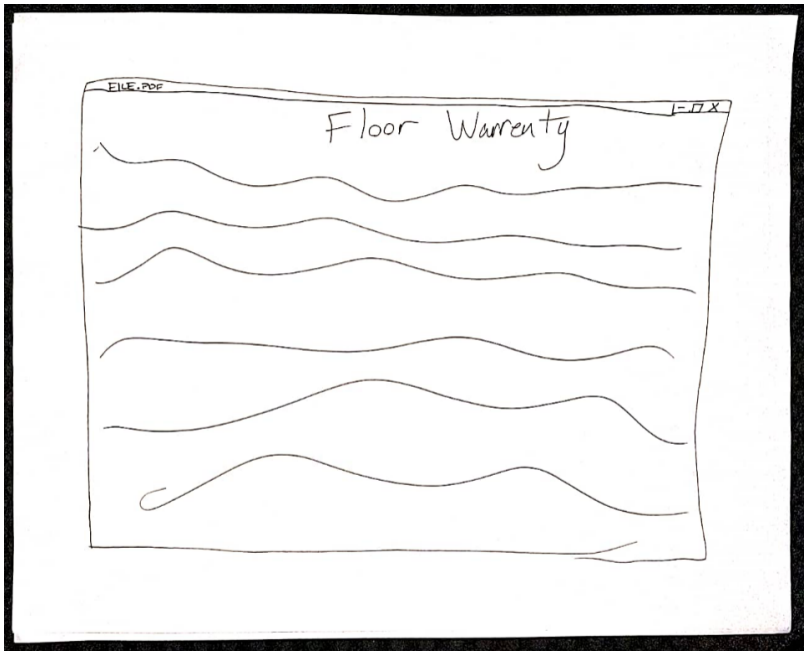
For the user to be able to find the file that they just added to the Organizer program, we implemented a Search feature at the bottom of the window that when clicked on would extend a textbox for typing in keywords that would allow the user to find particular files from keywords that existed within the file.



The user in our situation would type in the word “Flooring” within the textbox and then click the GO button at the end of the text box as in the image shown below:



After clicking GO, the user would find that they in fact did not change any location from within the program. This would indicate that the Search feature actually did it's job and that the only file in the system that contained the keyword "flooring" is actually contained within the Oven Range folder. We would then have the user double click on the File.pdf that was contained within the Oven Range folder and the PDF would then be displayed:

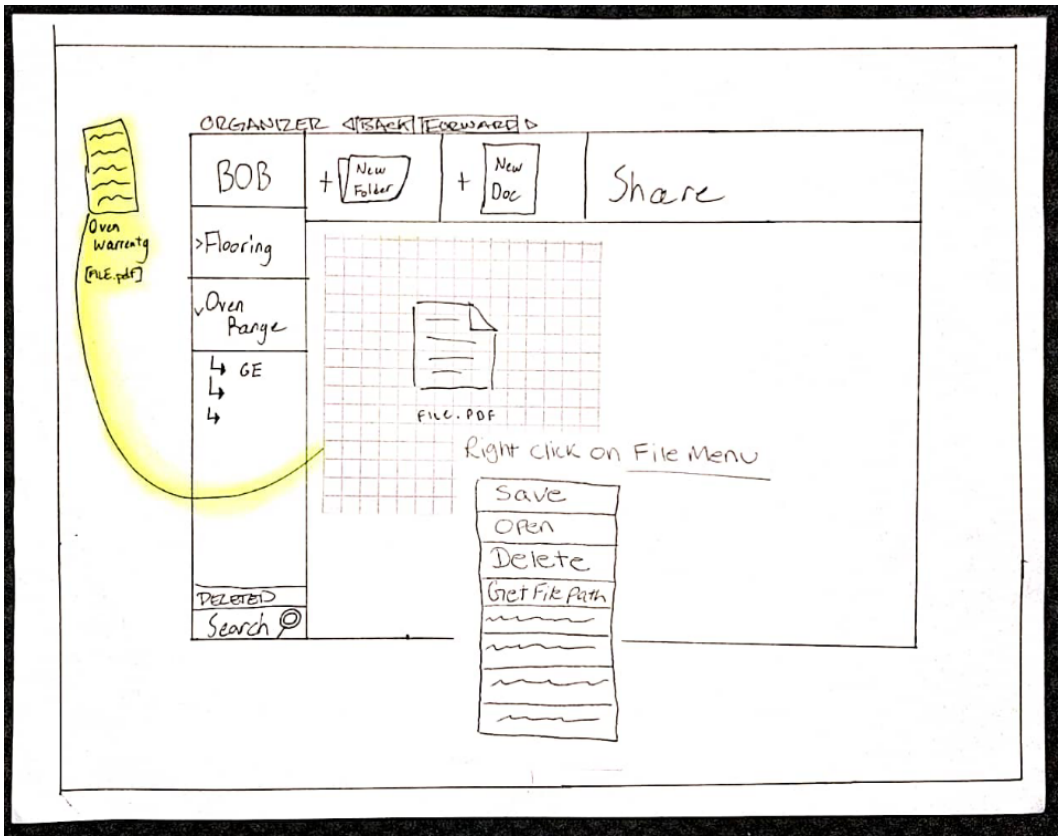


It seems that in the user scenario, the user dragged and dropped the file File.pdf into the Oven Range folder under the assumption that the file may have contained the Oven Warranty instead of the Floor Warrenty. This tends type of unfortunate happenstance tend to happen when one chooses to name their files “File.pdf”. We then have the user exit out of the PDF by clicking on the X button on the top right of the window.

Delete/Remove

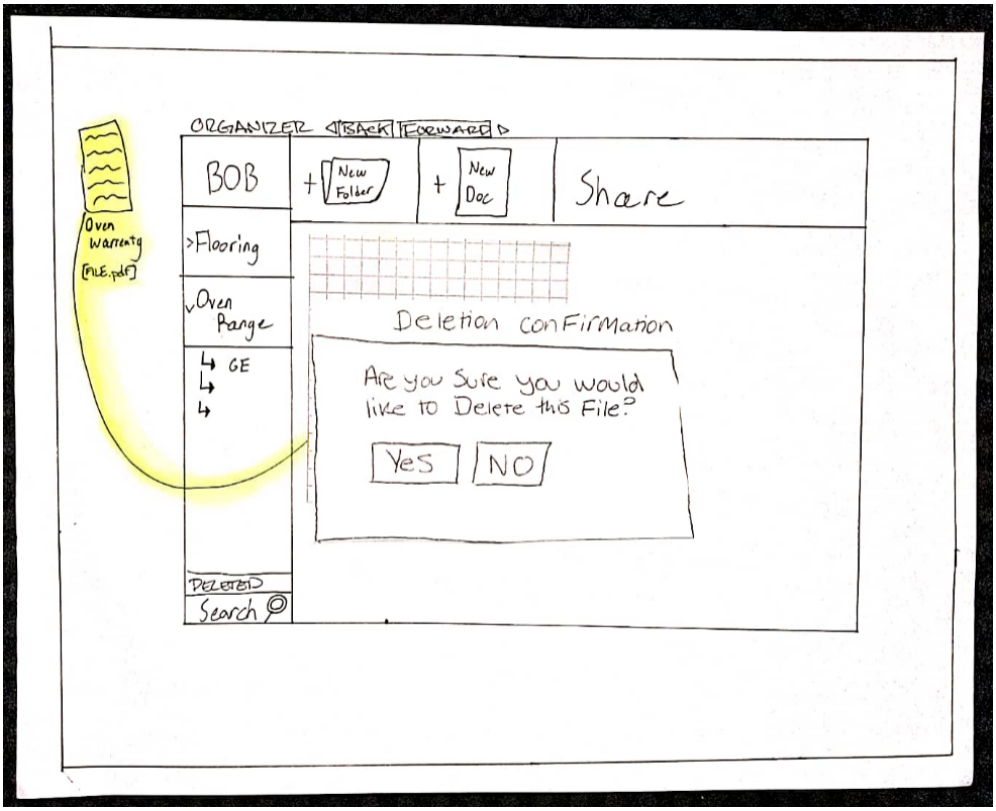
- Right click, Delete.

Now that the user has moved around files and enacted the primary keyword search method, it now comes to the final implementation of functionality, the ability to delete unwanted files. It is important to state that if the user wanted to, they would still be able to drag and drop the File.pdf from the Oven Range folder into the Flooring folder if they wanted to. However to show the process of deleting the file, we would have the user right click on the file File.pdf. This would show a drop down menu with a few different processes for the user to choose from.



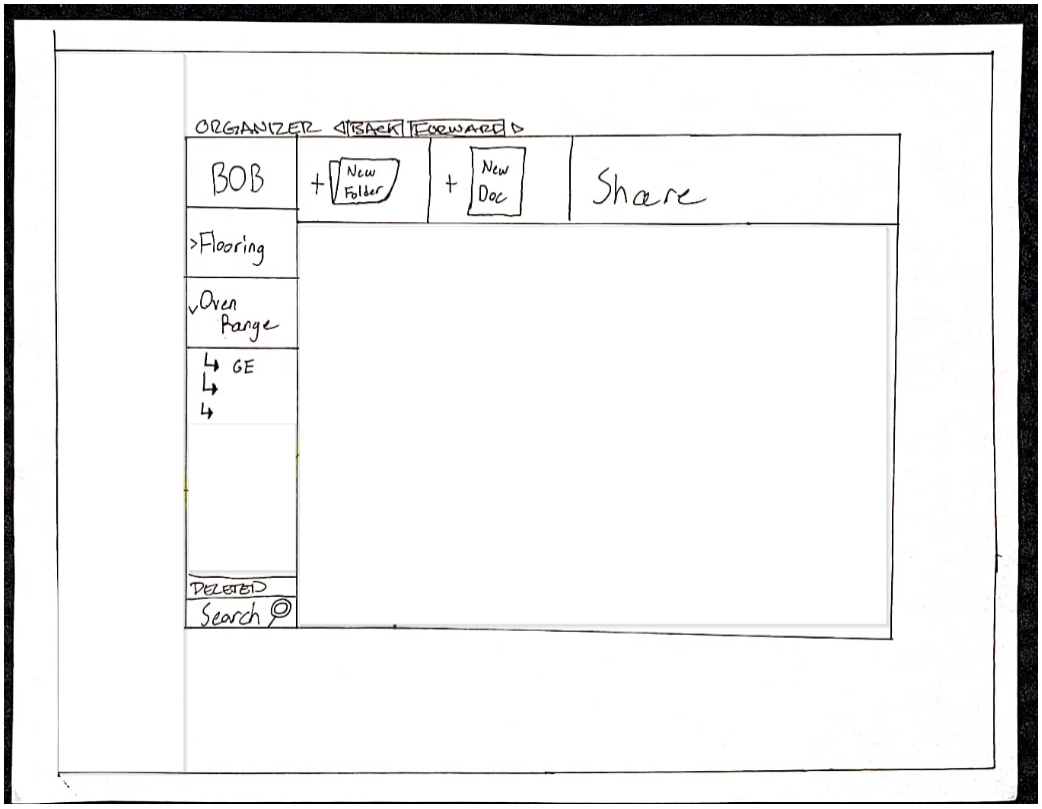
This menu has a few different feature to choose from including “Save” to potentially save the file somewhere else or if it is still open in the background, or to “Save As” a new name. There is also the the ability to “Open” the file and a secondary aspect of the Search function is still present, which allows the user to copy the “Get File Path” if they wish to look up the file later using either their operating system or from within the Search bar.

Instead, for the purposes of the user test, we will have the user choose “Delete”.



A popup box titled “Deletion Confirmation” is displayed with the question, “Are you sure you would like to Delete this File?” with two clickable button that display “Yes” and “No”. This is done to ensure that the user does indeed wish to delete this file by sending it to the Deleted folder present at the bottom left of the screen.

Under the assumption the process was successful, the final screen of the program during the user prototype phase should look like this:



This is the final image of the paper prototype.

Technical Specification

At this point of the implementation, the first iteration of our Paper Prototypes were successful, however, we found that we were scripting the user into particular scenarios that we wished for them to move through. Obviously at this stage, it was what was needed. We cannot have the user attempting to crash the program if we only have a few pieces of paper and a thousand ways to make the program fail such as deleting the user profile or deleting the Deleted Folder. We make no assertions about future implementation technologies, but we do know that the next iteration of our prototype will include more features as well as “failsafes” to stop the user from making the program crash on purpose. We will try to remove any abilities to edit things that should be uneditable by simplifying the next iteration of the paper prototype. We will also include additional partitions of the paper prototype that will allow an indication that something has crashed or that an inappropriate function or action has taken place.

Our future goals will be to implement more of the features contained within our User Stories, but getting a fully functioning paper prototype that has a successfully working implementation of our first three priorities takes precedence before we finish this sprint. We will continue to refine the current iteration of our paper prototype prior to the next user testing phase. Once that is finished and we have a working prototype that exhibits the required functionality, we will move on to the the next implementation in our backlog that are from the next few User Stories in our list.