Tools: PowerApps, Adobe XD, Excel, HTML, CSS

# The Problem

As a part of a larger service the pipette department provided, we needed to reconcile our inventory of 1200 pipettes with any of about 400 different researchers. The initial solution of email users resulted in a lot of back and forth and a wealth of inaccurate data.

# The Goal

Design and develop a solution that allowed researchers to claim pipettes as their own using our existing inventory.

# Background

During this time, I was working for Unity Lab Services, a division of Thermo Fisher Scientific that works onsite with customers to provide lab support. As pipette coordinator, my responsibilities included scheduling and preparing the client’s pipette instruments for their annual calibration while ensuring all records were kept to compliance.

# Initial Designs

## Design Considerations

Before any wireframes could be designed, I needed to plan out how it would work.

To make it as easy as possible for researchers to claim pipettes, I wanted to allow them the do so using their pipettes’ serial or asset (our identifier) number. With COVID-19, however, many researchers would only be onsite when they needed to be in the lab, and I couldn’t expect them to make this a priority while working on their experiments.

To alleviate this issue, I chose to allow researchers to give us the physical location of their pipettes along with an estimate of how many they had there, along with any other identifying information they could provide. With that information, I would then be able to reconcile that with our inventory on my own to ensure we matched the pipettes to the correct owner.

In addition to this, I had a few more requirements for this application:

1. Pull from our existing inventory records
2. Accessible by anyone within the company
3. No permission requests
4. Save submissions along with user’s name and email address

Due to these considerations, I decided the fastest and easiest method of implementation would be to use Microsoft PowerApps. The only issue, however, would be that I had never personally used this method before.

## Wireframes

As this was my first foray into PowerApps, I played it very safe on the earliest wireframes. While I had never designed for PowerApps, I had used a few and even ran some user testing on one before. I stuck to what I knew it could do with no frills, which resulted in the below mid-fidelity wireframes.

# Building the Application

I wanted to start building as soon as possible. I needed to make sure that what I had in mind would actually work and that I could create it within a reasonable timeframe.

Getting the first version of the application built was simple. PowerApps wasn’t difficult to learn, although I had to ignore all of my programming experience to understand how to assign values to anything.

I opted to build only two screens: the “By Number” lookup and the “By Other” lookup. The “By Number” would just switch a few controls between Asset and Serial number to simplify everything. The biggest challenge was getting the drawer to work – but it wouldn’t be anything compared to the challenges I would face later on.

I used a Microsoft Excel Online spreadsheet as the data source: one worksheet held our pipette inventory while another held all submissions. After a few days, everything seemed to be working well and I was pretty satisfied with my quick work.

# Improving the Design

With things working, I turned back to the UX and UI. I made a recent realization that if this were to be client-facing (which is not something we do often), it needed to be good. Not only did it need to work, but it needed to be easy to use and be visually appealing.

This meant it was time to return to Adobe XD. As I don’t claim to be a UI designer, I searched around for inspiration, and began work on this wireframe. I realized the system would be better built using flat navigation rather than the basic hierarchical navigation. To do so, I removed the “Choose Lookup Method” screen and opted to make the “Lookup by Asset Number” the first screen users see upon opening the app, which would also quickly telegraph that this was our preferred method of communication.

I performed some quick user tests and made some changes to the layout: I distinguished the most important information from other info, added a location summary to the drawer, and clarified what the plus button’s functionality. After adding a color scheme, I had finished what would be the final wireframes of the project.

# The Challenge

Bringing this design into PowerApps was much more challenging than anticipated. **All** of these challenges were caused because I wanted to make sure that my users would have the best possible experience with this app – it all took time, but was worth it.

The first issue I ran into involved drop-shadows: none of the pre-built controls supported them. Every control with a drop shadow had to be created as an HTML text object, which then had to be sized both within the HTML code and the PowerApps editor and styled with inline CSS.

The next issue was icons. PowerApps has a selection of icons that Microsoft has designed and programmed to work as you would expect: icons can change upon selection, hover, and disable. These icons are skinny, flat, and minimalist, and ultimately, not my style. I only had 2 icons – one of which is on one of the most important controls within the app – so I wanted them to be clear and noticeable. However, PowerApps only allows you to import images, it doesn’t allow you to make them work like icons, meaning I couldn’t make them show feedback when users interacted with them. On top of that, I couldn’t make the HTML text object change because it too didn’t have a hover control.

In order to show interactions with my buttons, I had to create 3 or 4 layers, depending on the button. In the back was the HTML object, which defined the style. In front of that was the text and/or the image of the icon. And lastly, in the front, was an actual button control that was invisible until you hovered over it – then it would increase its opacity, making both the image and html object seem like it was reacting.

The next issue was also involved with feedback: when I performed more testing, people didn’t notice that the counter at the drawer was incrementing up when users claimed a pipette. I needed some sort of animation. I decided to create an animation that would move the Asset number down to the drawer (growing to a larger font size along the way), then quickly fade away at the pipette count as the count increased.

PowerApps, however, uses declarative code, which means it gives results, not instructions. I couldn’t explicitly tell the control to go here and disappear, I had to tell the controls where it was at this moment. To get around this, I had to create a hidden timer and a dummy number object that would travel between the real number and the pipette count. I then told the dummy number to calculate its position between the start and end position (as well as its size) based upon the current value of the timer. For more information check out [this article from Michał Guzowski](https://michalguzowski.pl/animation-in-powerapps/), which is what I used to get started on this endeavor.

After building this, I added a similar setup for buttons. Each button’s drop-shadow would greatly reduce for a brief moment after being clicked, adding another layer of feedback to the app!

# The Final Changes

I was close to finishing the app but needed to make sure that everyone could use it easily. I decided to add a basic FAQ and some onboarding screens. Check out those below!

Of course, this wouldn’t have been completed without one final hurdle. As I prepared to roll this out to everyone, I learned of one simple issue; in order to share the app with everyone, I needed to also share the data source with everyone as well (all 24 thousand employees!). Easy fix, right? Just uncheck a box when sharing. But... where is that box? Spoiler Alert: there isn’t one – at least for Excel Online. In order to finish this up, I had to break up and migrate my data. The pipette inventory was moved to a static Excel workbook that I uploaded to the app, which automatically shares it with the users when they open the app. Submission data was migrated to a SharePoint list, which **does** allow you to uncheck the box that notifies all users when its shared. Crisis averted!

# The Results

This journey is all well and good, but how well did it work? This was launched in early January, and I only brought it up in meetings 3 times. Prior to this, we spent 2 weeks to accumulate 5% of reconciled data, much of which ended up being inaccurate. With this solution in place, in less than 2 months we jumped to over 95% of our inventory reconciled. We’ve been using this data to communicate with researchers ever since, and there have been 0 complaints about inaccurate data.