Class 5 – Product Evaluation

"How well does your product work?" Does your product behave as you intended it to behave? How did you go about testing your product? Which aspects of your product are robust? Which are fragile? What are the known bugs?

----------------------------------------------------------------------------------------------------------------------------

Testing

**Overview**

Throughout the development of our application, we conducted various forms of testing to ensure that the user experience is as smooth as possible. In addition to client-level use case testing, we also conducted lower-level experiments, utilizing a toy example to refine the database logic in our system. Below are some primary examples of how we tested our system.

**Use Case Testing**

At the client level, we conducted use case tests by following various logical paths through the application.

*New User*

* Initially, we were testing Class 5 with our existing profiles, but needed a way to test the boundary condition of a new user and ensure that a profile can be created in the first place. We wrote the method remove\_user() in our database module to manually remove our user profiles, and simulate navigating the application as a new user.

*Leaving after committing / not committing changes*

* We navigated away from the “Profile” page after committing changes to our preferences or intentionally *not* committing these changes to determine whether this information was properly saved or not saved.

*Input cross-validation*

* We input all combinations of invalid course preferences (i.e. overlapping Liked/Disliked courses, overlapping Fifth/Disliked Suggestion, nonexistent dept and num, etc.) to test the functionality of our input validation.

*Refreshing Suggestions*

* When building our functionality to responsively refresh suggestions in the Dashboard, we reloaded the page, or navigated to and from the page between refreshing suggestions to test whether these updates persisted in our database.

**Low-level Experiment – Toy CourseGraph**

We conducted an in-depth experiment when debugging our Database methods to ensure that a user’s preferences were updated accurately. We initially tried updating our GourseGraph with a more precise technique – only touching edges for courses that have changed – but this created errors in our logic. Rather than debugging using a real user’s more complex data, we used simple lists of ints representing Liked/Disliked courses, and a dict of edge relationships to test our remove\_edges() and add\_edges() methods. Using this rudimentary graph structure which tracks only *relationships* without the unitary or edge weights, we were able to examine our logic step-by-step by printing out the Liked/Disliked lists.

Our code for this experiment can be found in **testing/toy\_graph.py**, housing the toy example, and **testing/test\_graph.py**, which we used to field test our reworked remove and add methods on the actual CourseGraph. We eventually adapted these methods to become the delete\_edges() and add\_edges() methods in the Database.

Robust features:

*Input validation*

* We cross-validate all input on the profile page so any invalid entries (e.g. the same course indicated as Liked and Disliked, a graduation year before 2020, an nonexistent department and number combination, etc.) will display specific warnings to the user. Hence, the user can see exactly what is wrong with their profile info and easily adjust it.

*Updating a user’s preferences*

* When a user updates their preferences, Class 5 successfully updates the *subset* of courses whose new inputs are valid without updating the *subset* of invalid courses. For example, if the “Courses You Like” section of the Profile page has an invalid entry and the user saves their profile, a warning will flash at the top of the page and changes will be committed based on the *new* valid “Courses You Dislike”, and the *previous* valid “Courses You Like.”

*Event handling when refreshing suggestions*

* When implementing the feature allowing users to refresh their suggestions, we initially came across a bug that prevented a new suggestion from repopulating in the list. We realized that when multiple suggestions were disliked in rapid succession, the second signal to “remove” a suggestion was overriding the first signal, and thus was preventing the first “refreshed” suggestion from ever entering the list. In order to prevent this problem, whenever a suggestion is removed, we temporarily disable the click action for “remove” buttons for all other suggestions by setting the attribute of ‘onclick’ to the empty string (‘’). The ‘onclick’ attribute is not set back to ‘removeSuggestion’ until *after* a refreshed suggestion has been retrieved and appended to the end of the Suggestion list, which guarantees that only one request will be sent to the database at any given time.

Fragile Features

*Invalid inputs*

* Even with Class 5’s autocorrect, users can still input an invalid department and number, class year, major or certificate, which could result in an inability to save their profile.

*Getting Suggestions*

* It typically takes about one second for a user to access suggestions when navigating to the “Dashboard” page. Between navigating from the previous page and getting suggestions, there may be a moment when it is ambiguous whether the database is being accessed properly. If the database crashes right before the page loads, then the Suggestions list may not populate with anything.

*JavaScript Dependent Application*

* Class 5 is heavily dependent on JavaScript for the frontend and its responsive functionalities. If the user disables JavaScript in the browser, then the application will not function.

Known Bugs

* Our current graph system will not scale well. While the course graph is currently sparse, as more people input their data, the number of edges grows exponentially. A full graph would have > 4,000,000 edges, which need to be traversed every time a user updates their liked courses. A better graph database architecture is necessary for the future.
* There is no website history on the dashboard – we cannot go back and forth between previously or subsequently-viewed course information.
* When deleting a suggestion, the footer moves up and down again while the server loads in a new suggestion.
* We do not ensure users that their information is confidential (seen only by our algorithm).
* On the iPhone, whenever a user clicks on an input HTML object, which in our case includes the buttons in the Dashboard page and any inputs, the screen will zoom in slightly.
* When removing a suggestion in the dashboard page, the course information does not change. I.e., if I remove MAT 203 and have MAT 203’s course information pulled up, the course information will not change.
* If a user has more than 6 disliked suggestions and presses the “Save Changes” button, every disliked suggestion past the 6th disliked suggestion will be deleted from the user profile. The system does not crash, however, this is a *primary* bug to fix in later releases.
* There is currently a heavy bias towards COS and ORF courses for most popular suggestions. This is because we asked our friends to help populate the initial database, and since we are COS and ORF majors, most of our friends are too. Since non-STEM courses have fewer edges, non-STEM students see mainly the Most Popular courses to fill their suggestion lists, which are COS and ORF courses. This bug will be fixed as more people use our system.

----------------------------------------------------------------------------------------------------------------------------

Evaluation by Users

When evaluating Class 5 we sought the assistance of 3 acquaintances. We outlined a list of tasks that cover the majority of the application’s features, and most of its useful logical paths, and monitored\* our testers while they used our application. In order to ensure that we got as much feedback as possible, we asked carefully-framed questions at different points in the walkthrough to determine whether our testers thought our system was intuitive, and whether the application was satisfying their expectations. Attached at the end of this document is an Appendix containing the full User Evaluation Task List, and a summary of our findings.

\*One of our testers was a roommate, so we monitored him in person. We monitored the other two testers over Zoom by screen-sharing.

**Feedback**

Our feedback was curated from a larger sample of 15 users, whom we did not closely monitor while they used our system. Instead, we sent them a Google form and requested that they fill it out after exploring the application. Below are some suggestions for different components of our application, and ways in which these issues can be resolved.

Critiques / Suggestions:

***Home Page***

* “Makes sense, but might be more useful to take people directly to profile” (response to “Enter the Dashboard to get started”)
  + We initially flashed the warning “Please complete your profile before accessing the Dashboard” when a new user without a completed profile clicks this “Dashboard” link on the home page. We changed this warning to “Please complete your profile,” but in future versions of the application, we should change this Homepage message to “Log in to get started,” so that the user is less likely to be confused about what to do first.
* “Think more hyperlinks on objects would be nice. For instance don't have hyperlink on the word Dashboard - you could make a button or the entire box a hyperlink. Same goes for enter your preferences, get suggestions boxes - could make those hyperlinks bc I thought that was what they were able to do but then realized I couldn't click on it. Fun bios! :)”
  + This would simply require some reorganizing of the Homepage layout and adjusting the hyperlinks in our HTML.

***Profile Page***

* “Using only course codes for inputting favorite and disliked courses was a lot of work because I only remember the names, not the codes. If I hadn’t had a friendship obligation for filling out I probably would have given up halfway through.”
  + This was one of our most salient critiques, and one of the first updates we will make to the application moving forward. Including titles of courses will streamline the user experience and make it much easier to recall classes without knowing the department and number. We would display course titles within Class 5’s autocomplete dropdown, and in all text input fields.
* “Can you remind people to save when they make a change on the page?”
  + We resolved this issue in the current version of our application, which uses a Dirty Form Detector to determine whether the profile has been edited, and displays a browser-specific error if the user tries to leave the page without saving.
* “Might be useful to have a tigerpath tie in and have people like or dislike every course they took.”
  + Integrating with other TigerApps is definitely something we are considering when expanding this application over the next semester, but would require us to differentiate “all” of a user's courses from the ones they have Liked or Disliked.
* “In the Profile page, it was annoying that if you put an invalid input, the entire page would not save.”
  + We changed this feature in the current version of our application to specifically commit all valid information, and *only* refuse to commit inputs or sections with invalid information.
* “In the mobile version, the input fields zoom in a ton when you tap on it”
  + \*We have not yet found a solution to this issue, which will likely require some more wrangling of CSS.
* “I’d like to not have to keep going back and forth between the Profile and Dashboard page.”
  + We can discuss designs where everything is concentrated within one page with a sliding bar. This would be a drastic change, however, and would require much thought.

***Dashboard Page/Suggestions***

* “I thought thumbs up / down could be clicked. Would prefer course reviews/evaluations to be on the same page for the course rather than linked to Princeton courses.
  + In future versions of the Class 5, we’ll look to integrate more of this external information more natively within the application.
* “Avoid suggesting lower level courses (like COS 126) if higher levels are already taken”
  + This would require some more advanced filtering – perhaps taking a user’s class year giving certain level courses (1XX, 2XX, etc.) more or less weight in our algorithm based on this class year.
* Users wanted to be able to go back through their history when navigating through course information.
  + Instead of using callbacks to separate URLs, we could store some information in the current URL and make the URL change dynamically. Then we could add this URL to the user history so they can navigate back and forth.

***Other Feedback***

* Allowing functionality for incoming freshmen without Princeton accounts.
  + One user had a terrific suggestion that we should allow incoming students to be able to use the platform (i.e. without a Duo Account).
  + We would have to make sure that our security scheme is well thought out, because CAS Authentication is our primary security measure.
  + However, we wrote the code with this functionality in mind (mainly for family members to also see our platform :), so it is definitely not out of reach.
* More information without going to FAQ page
  + This can be resolved by integrating more of this information into the application itself, either by directing the user to a tutorial before they fill out their Profile, or integrating information “( i )” bubbles throughout the Profile and Dashboard pages.

**Positive Responses**

* “Pretty”
* “Interesting bios”
* “HIS384 is what I’m taking in the spring!”
* “I got some pretty solid suggestions lol”
* “Would definitely use this for every semester”
* “This looks so good!”

----------------------------------------------------------------------------------------------------------------------------

Evaluation by Experts – Heuristic Evaluation

**1) Visibility of system status**

Our application ensures that users are aware of how their actions interact with the application through checks on input on the “Profile” page, and appropriate reminders to save changes.

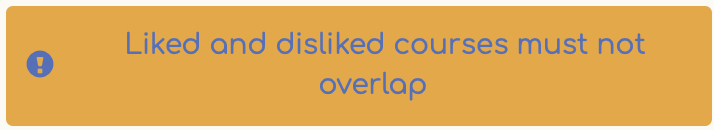
*Input validation and cross-validation:*

(In any of the following cases, changes do not save)

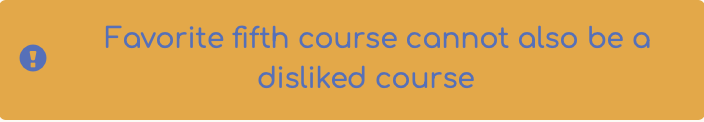
* If a user enters an invalid graduation year, major, certificate, or course department & number on the profile page and tries to save the profile, the app displays a warning of the form “Please enter a valid \_\_\_\_,” such as



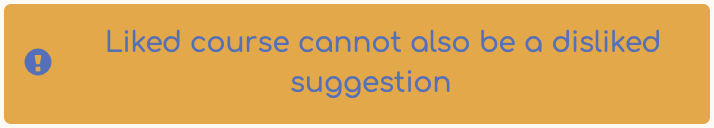
* If a user enters the same department and number *DEP XXX* under “Courses You Like” and “Courses You Dislike” and tries to save the profile, the app displays the following warning:



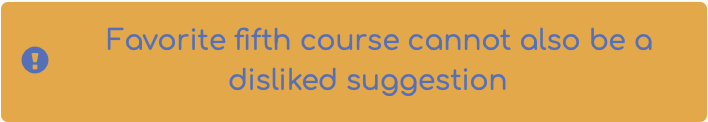
* If a user enters the same department and number *DEP XXX* under “Favorite ‘Fifth’ Course” and “Courses You Dislike” and tries to save the profile, the app displays the following warning:



* If a user enters the department and number *DEP XXX* of a Disliked Suggestion under “Courses You Like” and tries to save the profile, the app displays the following warning:



* If a user enters the department and number *DEP XXX* of a Disliked Suggestion under “Favorite ‘Fifth’ Course” and tries to save the profile, the app displays the following warning:



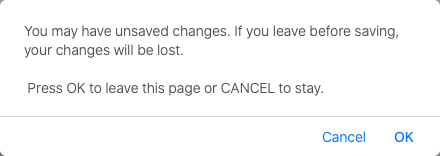
* If a user enters the same department and number *DEP XXX* as more than one Liked or more than one Disliked course and tries to save the profile, the app displays the appropriate warning:





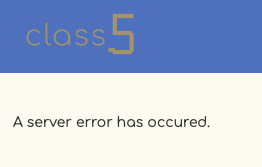
*Browser-specific messages:*

* If a user makes any changes to their profile and navigates away from the profile page, we flash the following warning message, reminding them that if they leave the page, their changes will not be saved.



*“Suggestions You Dislike”:*

* If a user has no Disliked Suggestions, our app displays the message “You have no disliked suggestions.”
* Because clearing suggestions is a much faster action than changing an input, if a user clears their Disliked Suggestions on the “Profile page,” the app reminds them that they must still save their changes.

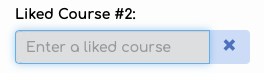


*Website Crash:*

* If Class 5 crashes, due to a broken connection to the database or an inability to access data, the user is presented with a server error message.

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

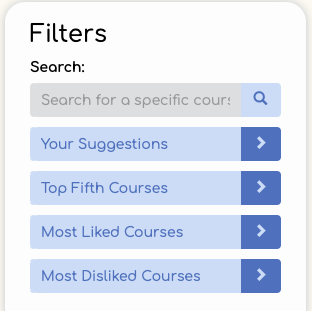
On the “Home” page, users immediately see the Class 5 logo and subtitle, which communicates the essence of the application as a class recommendation system for students. By the time a user logs into the application, they know from the homepage that “courses” refer to academic classes.

On the “Profile” page, our section headings are colloquial and easily understandable by users, who are all assumed to be Princeton undergraduates. For example, “Courses You Like” further clarifies to the user what a “Liked” course is. When completing their profile, a user is likely to start with their personal information and then move onto their course preferences, so the placement of the input sections allows a user to move naturally through the page from left to right. 

Moreover, when a new user first logs into the application, we use placeholders in the input fields to further guide the user.



We used orange as an accent color for the “Save Changes” button, as well as input validation warnings in order to draw the user’s attention.

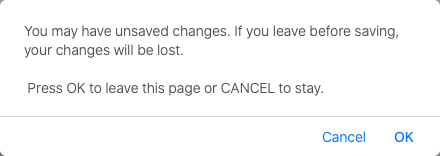
The “Dashboard” page is only accessible after a user completes their profile. Similarly to the “Profile” page, the “Dashboard” has an ergonomic layout. “Filters” are on the left, which control the list of courses in the middle section, and either searching a course or clicking a course in a ranked list displays its information in the rightmost “Course Info” section. When the browser window reduces in size, “Filters” and the list of courses are both above “Course Info,” maintaining this natural left-to-right or top-to-bottom ordering. 

Under “Filters,” we use a placeholder to prompt the user to “Search for a specific course.” While “Filter” is perhaps not the most intuitive heading for users, the button labels help to clarify what a user should expect to see when they click a given filter.

Under “Course Info,” information is organized naturally, and Class 5 user statistics are displayed graphically to more clearly present a courses like-to-dislike ratio to users.

**3) User control and freedom**

Although Class 5 does not directly support undo and redo actions, our profile page has functionality to warn users before leaving the page without saving. We adapted a Dirty Form Detector from Github (<https://github.com/codedance/jquery.AreYouSure/>, Accessed 10/15/20) to detect when a user has edited any of the inputs in their profile or has deleted or added any inputs, and thus warn them if they try to navigate away from the page without saving. The user can either click “Cancel” to stay on the page with no consequences, or hit “OK” to leave the page, *knowing* that any changes made after last saving will be lost.



**4) Consistency and standards**

All input fields in Class 5 have the same color and similar sizes, so the user can easily identify them throughout the application.

On the “Profile” page, all add and remove buttons (“+” and “x”) are consistently colored and labeled across input bubbles. We use the labels “Liked Course #,” “Disliked Course #,” “Favorite,” and “Disliked Suggestion #” to always differentiate between types of courses (especially Disliked Courses from Disliked Suggestions).

On the “Dashboard” page, we use a different button label (“>”) to indicate switching filters within the list on the right, and use a magnifying glass icon to distinguish the search feature. Under “Course Info,” we use the thumbs up, thumbs down icons to logically display the liked-to-disliked ratio, and we use “5th course <3’s” label to distinguish the number of fifth course favorites.

In the ranked list section, labels only read “Course #,” but the type of course (Suggestion, Liked, Disliked, Fifth) is indicated by the specific filter heading above. The button with the “x” label is used to remove a suggested course (like the “Profile” page), but unlike in the Profile, this button removes *and* replaces another suggestion. In the future, we will be looking to either change the icon on this button to match its more specific function, or to make it more visually clear that a suggestion is being replenished.

**5) Error prevention**

In our main.py function housing the Class 5’s Flask framework, we used careful error-handling with try / except statements to avoid any database-related errors. Every time we access the database, we start a new PostgreSQL Session() and call the appropriate database function *within* a try statement, ensuring that any exceptions are caught and handled with an appropriate error message. We then utilize a finally statement to guarantee that the session object is closed after each database interaction. We also handle empty JSON responses from Princeton’s OIT API Data source. If a JSON response is empty, then it should not be interpreted, and the client will just receive an empty string in the place of where that response would usually show.

**6) Recognition rather than recall**

Both the “Profile” and “Dashboard” pages have simple, practical layouts. Although the information is divided into different sections, they are all simultaneously visible within the window (assuming the user is on a desktop), and each takes up a size relative to the information it contains. The users options on their Profile are easily visible, indicated by the “+”, “x” and “Clear all disliked suggestions” buttons, and the Dashboard allows users to switch back and forth between switching filters, refreshing suggestions, and perusing course information.

Using Class 5’s navigation bar, the user can also freely go between pages, traversing to any other page of the application with a single click.

On smaller displays, the “Save Changes” button in the Profile may fall outside of the window if the user is at the top of the page, so we might want to move its location in the future.

**7) Flexibility and efficiency of use**

*Flexibility:*

On the “Dashboard” page, we give the user a variety of options to explore their suggestions or top courses for other filters. The user can switch filters with a single click to cater their search to a particular focus, or simply search for any course themselves. In the “Course Info” section, under “Users also liked,” the user is given a list of the top 5 most mutually liked courses, and can click on any of these courses to see its information. By exploring similar courses in this way, users can traverse our Course Graph structure dynamically and look for suggestions on the fly.

*Efficiency:*

On the “Profile” page, adding or removing a Certificate, Liked course or Disliked course can be done with a single click of the add (“+”) or remove (“x”) button. We also included a “Clear all disliked suggestions” button which allows the user to reset all of their Disliked Suggestions one click and thus eliminates the need for repeatedly removing Disliked Suggestions.

On the “Dashboard” page, users can responsively dislike/refresh a suggestion with a single click. This action is quick, easy, and allows a user to seamlessly comb through suggestions without worrying about committing changes.

On both pages, Class 5’s autocomplete helps to guide the user’s input, mitigating issues with validating information and spending up course searches.

**8) Aesthetic and minimalist design**

Class 5’s homepage gives a taste of the app’s color palette without overwhelming the user upon first viewing. The user can scroll down to find clickable screencaps of the “Profile” and “Dashboard” pages, with headings that describe each page’s primary function: entering your preferences and getting curated suggestions.

The app itself has a simple design, consisting of only the user Profile and the Dashboard. Both the “Profile” and “Dashboard” pages have a minimalist aesthetic that gives the user full view of relevant input fields without overwhelming them. In the Profile, bubbles – denoting rectangular sections with rounded corners – are used to segment different types of information (e.g. Personal Info and Courses You Like). In the Dashboard, bubbles are similarly used to enclose working “modules” within the page such as the “Filters” menu and “Course Info” pane. We chose Comfortaa as our main font to match this rounded aesthetic.

Both pages adapt to mobile displays and resize responsively, so that the bubbles maintain a natural ergonomic order (left-to-right and top-to-bottom). The user can interact with all of the main elements in the application – aside from the labels. The application also does not provide too much information at any given state (e.g., the “Course Info” pane only displays one course’s information at a time, and the ranked list of courses only displays results for one fitler at a time).

**9) Help users recognize, diagnose, and recover from errors**

In the “Profile” Page, input validation warning messages are clear, specific and expressed in plain language (see “Input validation and cross-validation” above). The user can then easily determine which section of the profile page contained the invalid input, and quickly fix it when subsequently saving their profile.

**10) Help and documentation**

From any page in the application the user can click “INFO” and then “FAQ” in the navigation bar to peruse a list of Frequently Asked Questions and their answers. This Q&A includes how to fill out your profile, a high-level overview of how our suggestion algorithm works, the utility of Filters, refreshing suggestions, and a few more pieces of information that clarify how the Class 5 works and will help users get the most out of the application.

----------------------------------------------------------------------------------------------------------------------------

Appendix

User Evaluation Task List and Notes

* **Log in to Class 5**
  + Comments how aesthetically pleasing the website is
  + People wanted more clickable objects
* **Fill out your Profile and save your changes**
  + People said the alert to “fill the profile before moving to dashboard” was a little confusing considering they hadn’t tried to go to the dashboard yet
  + People got annoyed when they made a mistake and the system made them start all over again
  + Some confusion over what the department and number of the courses they like and disliked are - suggestions for using course names instead, or a mixture of both
  + For smaller screens where the save changes button was unviewable, the user could not see the button and left the page without saving - they then lost all of that information. Of course, they were not happy
  + Main concern was the save changes button: people expressed they wouldn’t come back to use the site if it reset their inputs again
* **Navigate to the Dashboard**
* **Suggestion evaluation**
  + Report how many of these classes you intend to take, or have taken and liked in the past
    - All 3 had at least 1
    - 2 had at least 2
    - 1 had 4
  + This shows that our suggestion system is somewhat working. Our goal was to have at least 1 out of 10.
* **Search for a course**
  + Users were quickly able to figure out how to search for a specific course
* **Within the “Course Info” panel, navigate to external links to peruse supplemental course information**
  + Didn’t like this aspect
  + Wanted more information on the page itself
    - Wanted basically everything from princeton courses, but on our website
* **Use the “Users also liked” feature to dynamically explore the graph**
  + All testers enjoyed this aspect
  + It was here where they realized that there was no history in the dashboard page (i.e. that a user cannot navigate back and forth between course information in their history)
  + They wanted this to be a feature
* **Go to Filters, press “Top Fifth Courses”**
  + “Fifth” course evaluation
  + **Report whether these are an accurate representation of fifth courses at Princeton**
    - All 3 responded with a yes.
    - Positive feedback regarding the usefulness of this feature
    - Good for us as this list has only been created with 45 users in the system! It will get more accurate later on.
* **Go to Most Liked Courses / More Disliked Courses**
  + Liked/Disliked course evaluation
  + **Report whether this is useful information to know**
    - All 3 responded with a yes.
* **Go back to suggestions**
  + Dislike 10 more suggestions
  + Subsequent suggestion evaluation
  + **Report whether these suggestions are still good**
    - After disliking 10 suggestions, only 1 user had a suggestion that they intend to take or have liked
    - The users expressed confusion about if deleting a disliked suggestion meant disliking it. We remembered that during our presentation, Professor Dondero also posed this question. In a future release, we will update the FAQ or provide more robust tutorials.
      * Speaking of tutorials - one of the users stated that this would be a wonderful implementation and gave some suggestions on how we could do it (an info button on each of the bubbles).
* **Navigate back to the Profile page**
  + Update at least one of your Liked/Disliked courses
* **Return to the Dashboard**
  + 2 testers got one more suggestion that they intend to take or have liked
* **Refresh Suggestions Evaluation**
  + Report weather it is an interesting feature to remove / refresh suggestions
  + 100% reported yes
* **\*\*\*At this point, we did not have an FAQ page**
  + Our users through certain aspects of the app were confusing and some of our terminology (e.g. what precisely constitutes a “5th class”)
* **We created an FAQ page**
  + Asked our users whether the FAQ page helped them → 100% said yes
    - However, they still wanted more information in the FAQ page and again stated that maybe info buttons or a tutorial would be even more helpful.
* **After completing the task list for the desktop application, asked users to use the mobile version, and asked them to complete the same task lists**
  + Realized some bugs in the input fields
  + Consensus that the mobile experience was “comfortable”
  + One user said that the font size in the navigation bar was too small, so we made it slightly bigger later on.