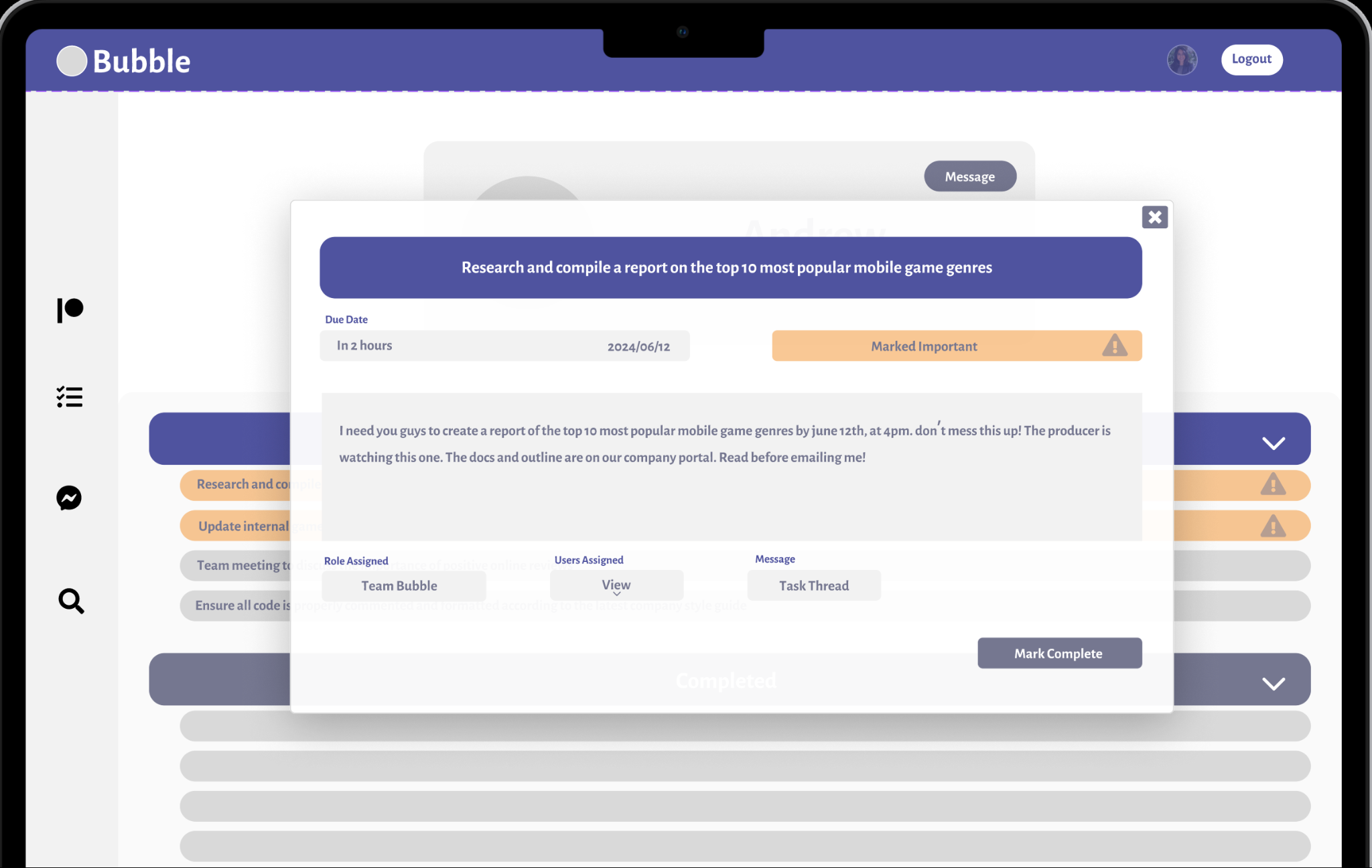
**Visibility of System Status (users should know the system status at all times and get feedback on interactions with it), and Help users recognize, diagnose, and recover from errors (make error messages understandable, and suggest ways to fix an error)**

* Yes, areas that need a system status will have clear error logging/output when needed. It will not contain data that is too specific and may breach system security.

For example:

* login -> “incorrect login credentials”
* signup -> “sign up information is not available” (saying “username taken” or the similar is TMI in system security).
* Task -> “title for task required” - and the similar.
* Messaging -> “User does not exist or can’t be found”.
* Roles -> “role already exists”
* etc

**Match between system and the real world (the system should resemble the experiences that users already had).**

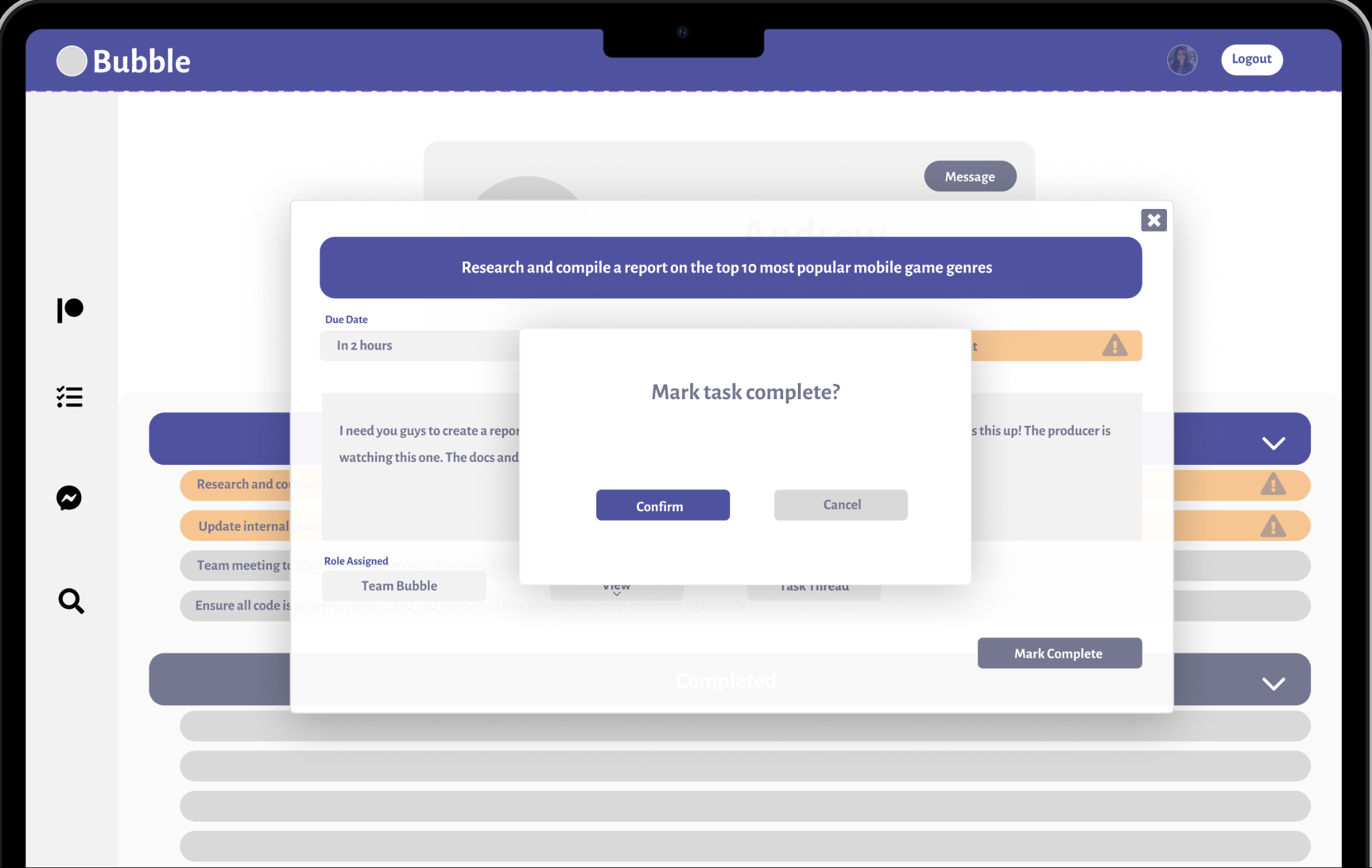
* I believe my designs follow this well. For example, I am not changing the core ideology behind icons. An “X” resembles closing a window, darker rounded corners elements represent buttons, orange highlights = important, and the like. Nothing has been changed to not represent what is common throughout real-life design. 

**User control and freedom (users should be able to reverse their action if done by mistake), and Error prevention (minimize the likelihood of making mistakes)**

* General users can prevent some of their actions. For example, marking a task complete requires additional layers of confirmation to prevent mistakes. The Team leader or Team moderator can undo a confirmation, but the general user can not. The layers of confirmation should deter the majority of mistakes.
* Messages can not be unsent, and should not be able to be unsent or edited by anyone. What is sent is sent, and this is entirely intentional.
* Moderators and team leaders can edit tasks and their team as needed.

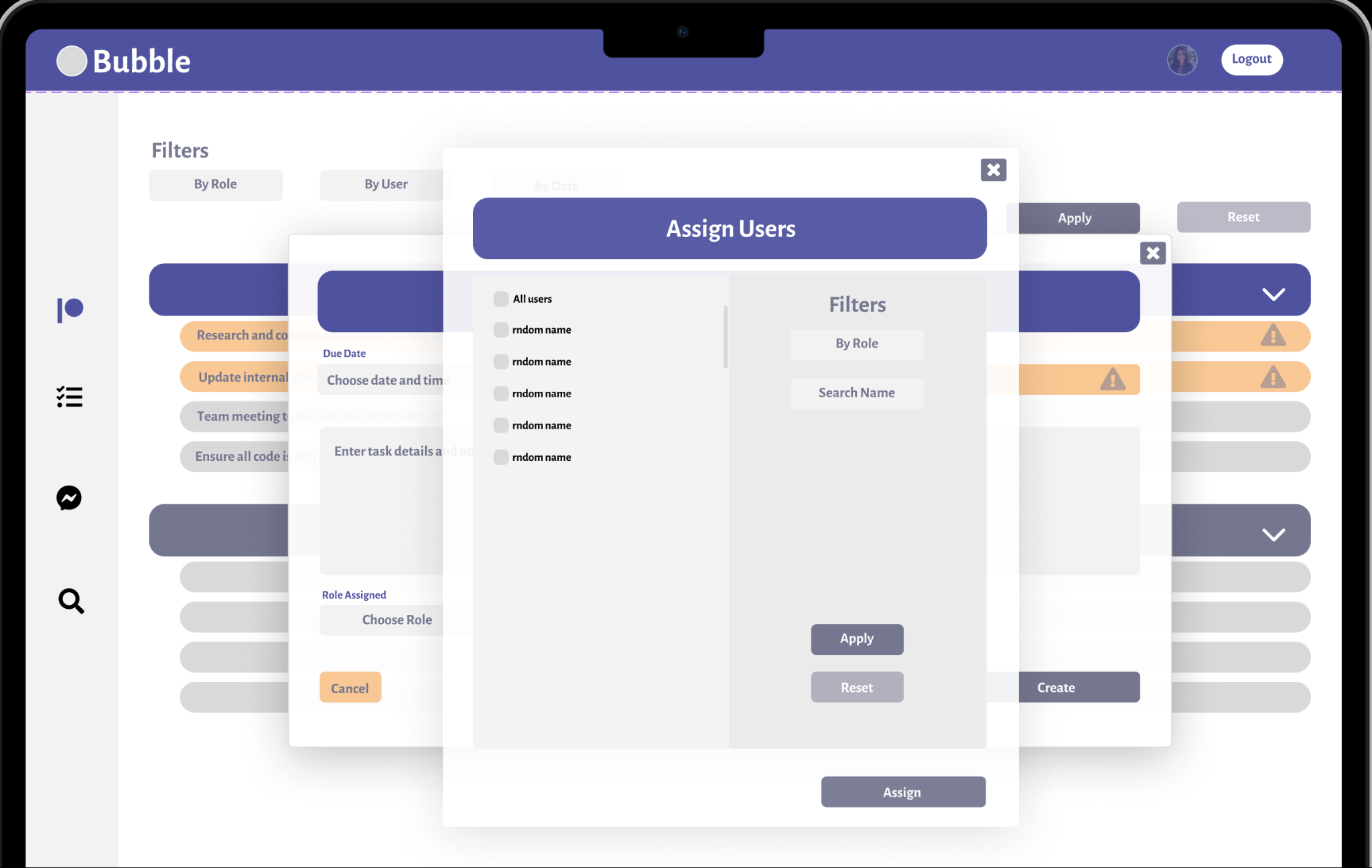
**Consistency and standard (similar system elements should look similar)**

* I believe my project follows this design philosophy. Elements, themes, and sizing are represented to flow cohesively. An example with the screenshot below and in the Figma prototype link above.



**Recognition rather than recall (users should be able to interact with the system without prior information or context), Flexibility and efficiency of use (both new and experienced users should be able to efficiently use the system), and An aesthetic and minimalist design (declutter as much as possible, less is more)**

* When needed, the information needed is always supplied. For example, assigning users to a task. The user list is always supplied, and can be further filtered. No “search and guessing”.



User Experience with the UI is intended to flow cohesively with inner and outer menus as subject to importance and layer-organization. For example, the assign-users window may contain a large number of users in a vertical list (scrollbar will be available). This means the window should be overlaid in a vertical-oriented design (longer on the Y axis) than the landscape (X axis) design of the previous window. Windows have a slight transparency to them to reflect the layering of the menus. If I went with the landscape design, then I would need to make the second layer of the window smaller than the parent container, which is not ideal for this scenario. It would also require additional scrolling (if not searched/filtered) by the user.

1. **Help and documentation (if a user has a hard time interacting with your app, make sure there’s help that’s easily accessible).**

* I don’t believe documentation is needed for this sort of product. I think user trial-and-error would answer any questions with the first mistake they make. Error prompts and generated information, or confirmation layers, should suffice to guide the user with this type of product. (For example, Amazon, YouTube, Facebook, eBay, etc… does not require documentation on how to use their service, since it is straightforward to most people through UI/UX).