CSC491 A4 - BlocX UX Research

Procedure

Provide background knowledge on what our application is aiming to achieve, along with its core features. Present Figma wireframe prototype to users and ask them to accomplish tasks aligned with the MVP's physical (visible) features.

Feature	**Task**
Users can create (multiple) contracts.	Initiate the creation of a new contract.
Contracts are highly customizable, with the ability for users to add custom financial behaviours based on a variety of digitally-verifiable conditions during contract creation.	Add and customize statements, making use of "conditions", "consequents", and "alternatives". If interviewee is stuck, suggest they create the example statement below.
Multiple users can participate in a contract.	Customize your contract to include other users.
Users can search for contracts they created or participated in.	Search for the contract `contract_id_1`.
Users can pay into contracts' accounts.	Deposit money into a contract's account.

Throughout the user research sessions, we want our users to speak out loud and voice their thoughts. We will take notes as the users speak. Our topmost priority is to gather feedback about the contract creation process - task 2 above.

Background Information for Interviewees

Financial contracts can be created. Each contract has its own account associated with it, in which users can pay into. Each contract contains well-defined logic that describes how the contract behaves. A contract is made up of *a sequence of logical statements*. Each statement follows the "if <condition>, then <consequence>, else <alternative>" structure. Conditions could be actions from users, or other factors, such as date and time. If conditions are satisfied, then some consequence is executed, else the alternative is executed.

Example of a statement

If
- date before Feb. 20

Then

1. jump to Statement 1

Else
1. pay account balance to original owners
2. END CONTRACT

Analysis and Summary

Overall, the interviews showed that our initial prototype of the UI was intuitive to use, *given that background information regarding what contracts are composed of was provided to the interviewee*. Without such guiding information, the UI was actually unclear and rough to use, with somewhat convoluted flows. Case in point, the second interviewee who received the background information was able to successfully complete all tasks with minimal guidance.

The only significant step where the second interviewee was confused was when she arrived at the statement editing page. There was confusion regarding when a statement is considered complete (e.g. whether there must be consequents/alternatives, and whether there could be multiple consequents/alternatives in a statement. The second interviewee also expressed that it would be helpful to explicitly write out “If”, “Then”, “Else” on this page to make it clear that each statement must follow such a structure. The instructor also brought up similar, but more intense concerns regarding the naming scheme and unclear structure.

The second interviewee confirmed that, given that she is able to come up with the correct logic, she would be confident that she is able to implement it given this interface. The instructor similarly stated that, at its core, the contract functionality appears to be correct.

The second interviewee also suggested that we make an in-app guide easily accessible. The guide need not be extensive, just an example of a contract with multiple statements is sufficient. She emphasized that having a sample contract to reference would help immensely.

Action Items

1. Redesign the (conditions / consequents / alternatives) UI for a clearer alternative.
2. Further abstract away the ‘hard’ structure dictated by smart contract logic; find ways to elicit the desired functionality without letting the user feel the underlying nitty-gritty.
3. Come up with more concrete functionalities to complement and help guide the design process.

***P.S.** While taking a break during Reading Week was necessary, we did not plan for it effectively. The delay in this assignment could easily have been prevented had we achieved better communication with each other and the teaching staff earlier on, rather than not being honest with ourselves regarding how much work would be done while the team was on break, and hoping an extension would be granted.

Appendix 1 - Interview with Instructor (Julian) Notes

Note: When running the interview with Julian, we failed to have a “background information for interviewees,” which likely could have prevented a lot of the pain points Julian encountered.

- When tasked to create a new contract, Julian clicked the + button instinctively - seems to be quite intuitive.
- After explaining that contracts have conditions that must be met, Julian easily clicked on the “conditions” tab.
- Julian noted that having a back button on every page is desirable.
- When clicking to select users, Julian was a bit confused by the flow. There already was a button for users on an earlier menu, and yet new menus kept appearing for users-related stuff. Basically, it seemed that the nested menus were a bit too convoluted, too many layers.
- After explaining what an escrow account is and when asked to add one to the contract, Julian got completely stuck. He thought it wasn’t a condition because money goes into the escrow account, so it’s sort of an outcome, but also it doesn’t make sense for it to be a “consequent” because it’s not really being caused by anything. He thought maybe “alternatives” was to be used here. Julian failed to complete this task without help.
- After hinting to click on “conditions,” Julian clicked through the appropriate buttons all the way to user selection.
- Julian commented that the user selection should be a sort of dropdown / search menu instead of toggles.
- Julian also said he doesn’t see how escrow accounts are involved with the user deposits flow.
- When it came time to select consequents, I wrongly informed Julian that statements are an “abstraction” for any kind of functionality, to which Julian said “sure” - in theory, makes sense, but not clear in practice.
- Julian had no trouble searching for and finding a contract.
- When I asked Julian to configure payment, Julian intuitively wanted to enter the contract and set up payment on a “per-contract” basis. However he later easily navigated to the profile and completed the task successfully.

Overall comments from interview:

Julian noted on the “conditions/consequents/alternatives” menu that the choice of words is not the best - being confused on the meaning of alternatives, not sure if consequents is even a word. He also felt like those 3 categories were too systematic and “restricted”, “surfacing too many [backend] implementation details of smart contracts”, and using them didn’t feel very fluid or intuitive. However, he did also mention that we have “technically have gotten it, in the sense that, like yeah, you can create a process,” but he urges us to look at ways to simplify the process for end-users, removing the exposed underlying functionality. Lastly, he said overall the actual Figma prototype was fine, and we got the core stuff right, but we need more simplification.

Appendix 2 - Interview with Freelance Graphic Designer (Helen) Notes

- Helen was presented with the Figma prototype. Extensive background information regarding the features was provided.
- I instructed Helen to initiate the creation of a contract. She immediately tapped on the “plus” icon on the homepage, and without further instruction or prompts, attempted to edit the name of the new contract.
- I instructed Helen to try to customize the new contract and add some logical statements. She proceeded to add a new statement, and then tapped on the button to add a conditional to the statement, which brought her to the “Condition Categories” page.
- Observing some confusion, I asked Helen what she thinks this page was for. After thinking for about 5 seconds, and after reading each category, she was able to deduce that there are groups of conditions (e.g. time-related or user-related).
- She proceeded to add a time-related condition successfully to the current statement.
- After the prototype brought her back to the “Current Statement” view, she attempted to edit the newly added conditional (not supported by prototype).
- At this point, there was visible confusion regarding how to complete the statement. In particular, she expressed that she was confused by what “consequent”s and “alternatives” meant.
- She asked if she needed to “apply” the condition.
- I hinted that she needs to define here what happens when the condition she added is satisfied. I asked “what if you want to proceed to another statement when this condition is satisfied?”.
- Helen expressed that it would be helpful if it was made clear exactly what items need to be added in a statement. She proceeded to click on “Add Consequent” and this time around selected the correct category and successfully added a “Jump to Statement” action as a new consequent.
- After this, she realized that she needed to add an “alternative” to complete the statement. She then added two “alternative” actions, namely “Pay Users” and “End Contract”. She expressed that it is not obviously apparent that multiple “consequents”/”alternatives” can be added.
- I then prompted Helen to start over. This time, I instructed her to create a contract that involved multiple participants.
- She proceeded to create a new contract with a condition that required other users’ deposits.
- I then prompted her to search for a contract with the name “contract_id_1”. She successfully used the search bar.
- I asked her to try to deposit money into the contract. She proceeded to tap into the active contract, and attempted to tap the “Pay Contract” button on the next page.
- After completing all the tasks detailed above, I asked if she had any general recommendations regarding the user interface.
- She answered that the UI itself is intuitive, but only because she saw the example statement I provided her as background information prior to the interview. She

mentioned that it was also very difficult to come up with the correct logic for a contract. However, she concluded that given that a user already came up with the logic, then creating such logic in a new contract is not difficult. She suggested that we provide templates or a step-by-step video guide explaining how contracts could be created using the limited logic options.