Developer feedback

The project used Pyth Entropy for selecting a random winner for a case where a prediction pool was created with the option of only single winner. The documentaton could be followed quite easily. The main complexity was to implement the functionality that would use the random numbers in the context necessary for our project. When comparing the implementation of Entropy vs Price feeds, the documentation and following the documentation went faster for Entropy. The examples were clear and could be implemented with small modificaitons. Liked the option that more than one random number can be generated with one call. The following points are worth mentioning:

- Proper callback handling, state transitions, and security considerations caused some initial confusion and could benefit from some more examples
- Understanding when and how events like RandomnessRequested and RandomnessReceived are emitted is not immediately clear
- Calculating and providing the correct entropy request fee raised a question how to determine the required fees and handle cases where the contract balance is insufficient. But this was not relevant for testnet deployment as for current project

The project used Pyth price oracles for determining the winning prediction. Although the documentation was mostly clear, it caused more confusion than for Entropy. Another point of confusion was the hackaton task of consuming the price from Hermes and then submitting that on chain which was not necessarily a required function from the perspective of our app. In the end it also took some time to implement properly in the frontend. When developing the backend, we had some confusion which interface to use to get the price data (we mistakenly wanted to use IPyth.Price at first and then corrected to PythStructs.Price)

Some issues or questions that arised during reading the documentation:

- The documentation could benefit from including detailed descriptions of all interfaces, structs, and functions, with practical examples which could ilnclude step-by-step tutorials about more advanced use cases of:
  - Fetching and updating price feeds
  - Handling stale data and errors
  - Integrating entropy services in various contexts
- Currently did not have time for unit testing but any additional examples for using mock data and interfaces could be beneficial.
- Errors like StalePrice cause some confusion and currently hard to understand, what exactly happens and what might be the solutions be for such cases.
- Could not exactly find data structure returned by Hermes. Had to run the code to see the response. Would have been nice to have docs for that.
- Getting that data into the correct format to post on chain provided also some challenge. Any additional examples of integrating fetching data and posting it would be beneficial.
- Data flow diagrams would be also a nice touch.

And actually the project had to be re-deployed to Base Sepolia in order to make full use of Pyth. We originally deployed on Eth Sepolia. We understand the possible reason but didn't think about it before starting developing and read Pyth docs slightly superficially and didn't notice that not everything was available on Eth Sepolia.