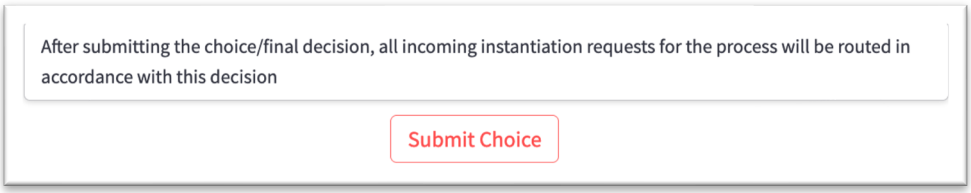
HITL-AB-BPM: Tutorial

This document aims to ensure **reproducibility**.
Although this document provides a relatively detailed outline, the application also includes a lot of help.
This in-app help can be in the form of small question marks to hover over:

Some buttons reveal additional details on hover:



And lastly, there are expandable info boxes with further information:

This tutorial starts with instructions on how to set up the project locally. Afterward, a brief overview is given to put the following instructions on how to use the app into context.

Get Everything Up and Running

1. Install Docker Desktop[1] and start it.
2. Clone the git repository from https://github.com/aaronkurz/hitl-ab-bpm.
   Alternatively, since **this tutorial document refers to the functionality of v0.1.0**, you can download the source code of that version in the releases section of the repository.
3. Go to the folder `hitl-ab-bpm/source` with your terminal.
4. Run `docker compose build` and wait for the images to build.
5. Run `docker compose up` to start the images.

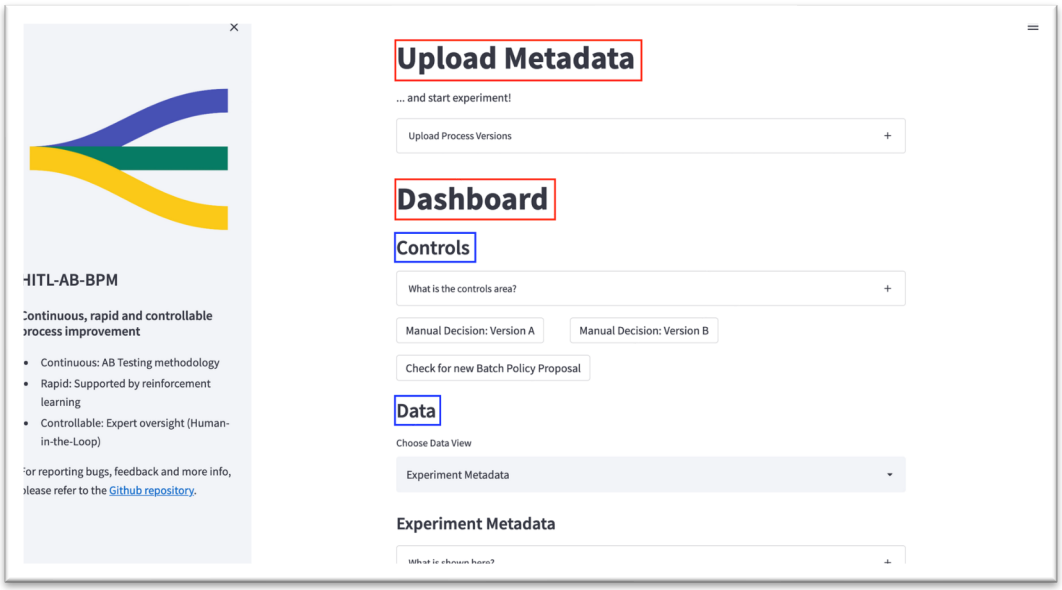
Now, one can reach the different system components under these ports:
- Backend: `localhost:5001`
- Frontend: `localhost:8501`
- Postgres: `localhost:5432`
- Camunda engine: `localhost:8080`

6. Since we want to demonstrate the usage of the *HITL-AB-BPM* tool, navigate to the frontend of said application (`localhost:8501`).

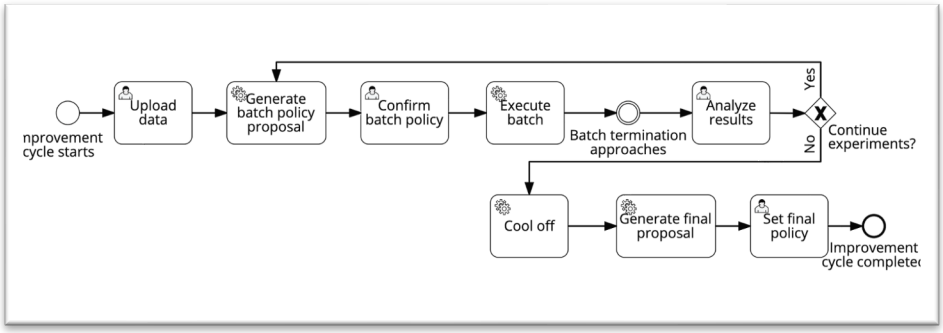
[1] Last tested with Docker Desktop 4.3.2, Docker Engine 20.10.11, Docker Compose 1.29.2

Get an Overview

Now, you should see something like in the screenshot below. The main sections of the application are *Upload Metadata* and *Dashboard*. The sub-sections of the dashboard are *Controls* and *Data*, and an additional subsection *Final Proposal* may appear at a particular application state (see section Get Started of this document).



An overview of the use of the application to test an improvement hypothesis is depicted below.



In the following, we will provide a detailed step-by-step guide on how one would go about testing two process versions. The guide will follow the general steps of the flow chart above to help understand where a specific step is situated in the overall *HITL-AB-BPM* methodology.

Get Started

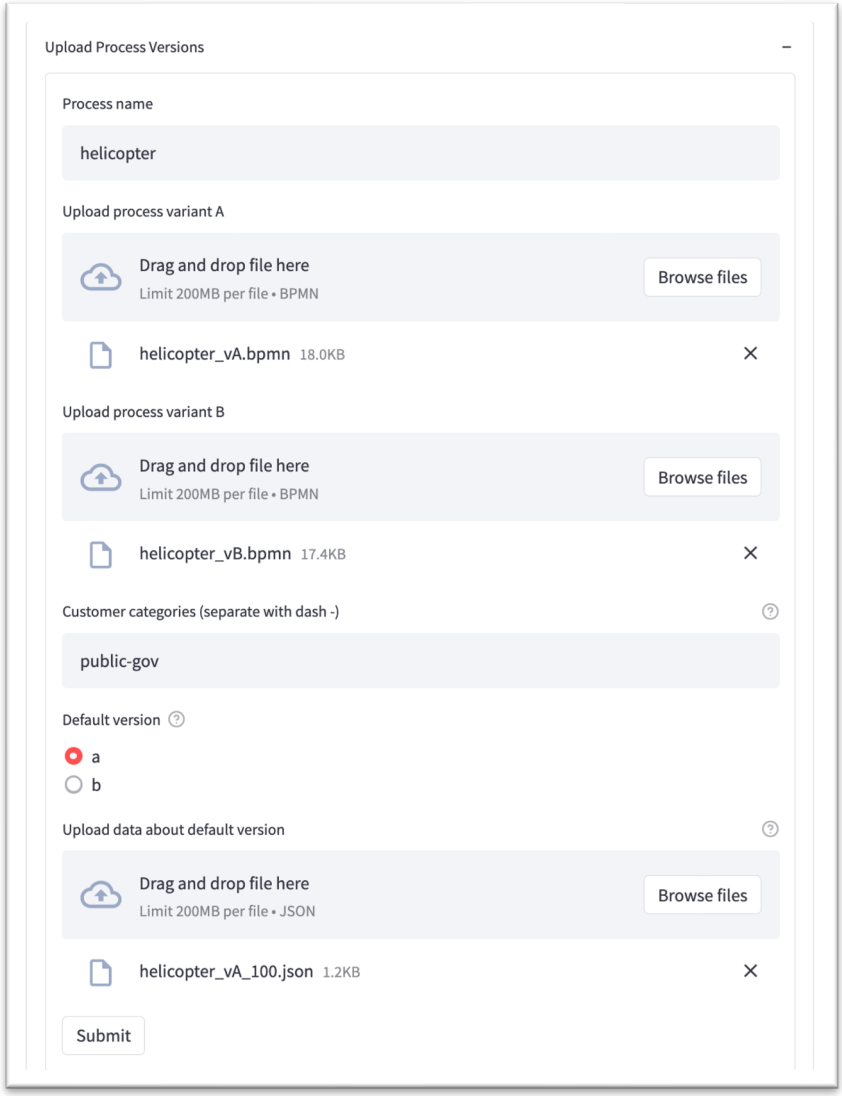
1. Upload data

First, the necessary data for the experiment has to be uploaded. Experiment refers to one AB test of two process variants. The required data are two executable BPMN files and a log of historical execution data. The BPMN files will be deployed on the Camunda engine, which

runs in a separate Docker container. The historical process data must be a JSON file of the following structure (values in seconds):

```
{
      "interarrivalTime": 20.4,
      "durations":  [ 21.198,
                13.041,
                100.124,
                97.04,
                132.099,
                45.144 ]
}
```

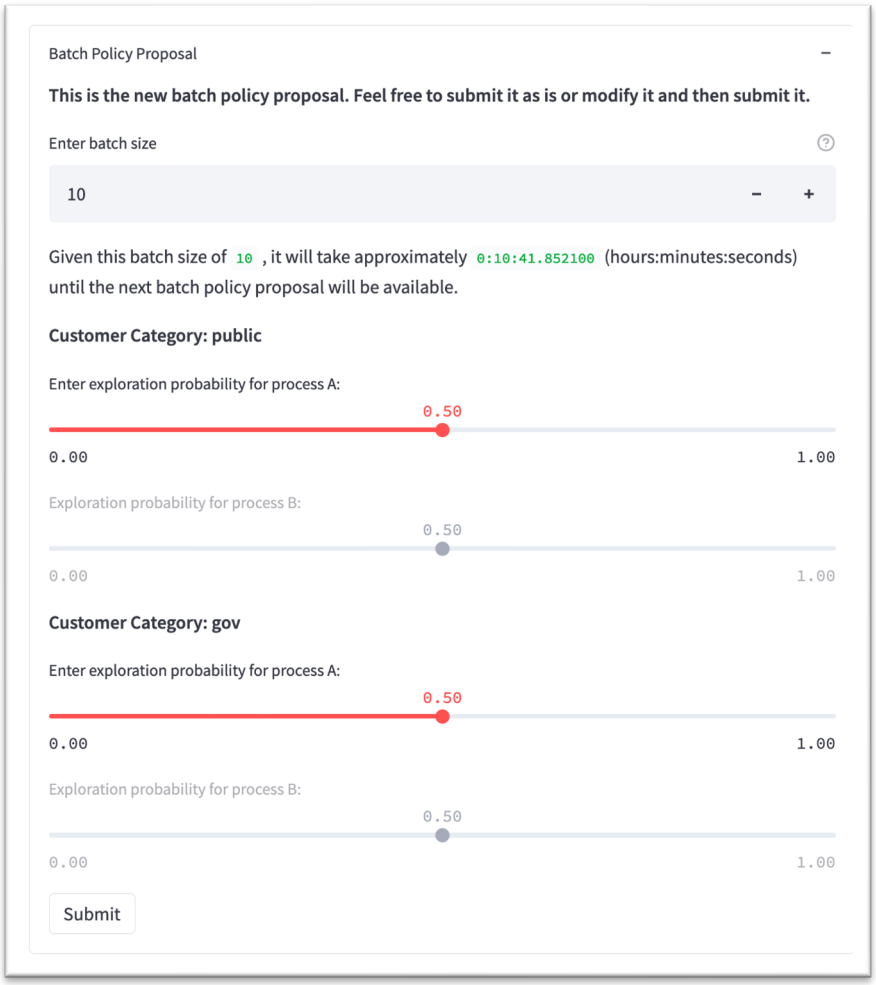
Furthermore, some metadata is necessary—namely, the process name, the relevant customer categories, and which is the default version. The default version is the version that new process instantiation requests will be routed to in between experimental batches. All this data can be entered into the upload form in the application section *Upload Metadata*:



You can find the example files in the repository `hitl-ap-bpm/api-tests/resources/bpmn/helicopter`.
The BPMN files we use have been extended with simulation parameters, so their execution is simulated automatically in the Camunda process execution engine. In case you want to test the tool with your own BPMN files, you either have to annotate them with suitable parameters too (more info on the repository of the Camunda BPM simulator) or execute them properly/normally in the Camunda engine.

1. Confirm batch policy

Afterward, the first batch policy is available. It can be seen by clicking the button *Check for new Batch Policy Proposal* in the section *Dashboard > Controls*. The first batch policy proposal will be a naive one, suggesting a 50/50 split between the versions for each category:

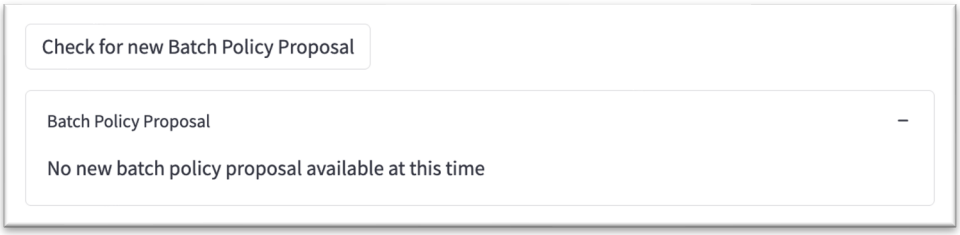


This is because there have been no experimental instances to learn from thus far, so the application can not yet make a data-backed decision on which process version is preferential. The human expert can, however, make adjustments to this batch policy before submitting it and starting the first experimental batch. Before submitting the first batch policy proposal and in between batches, all the incoming process instantiation requests will be routed to the version that has been set as the default version.

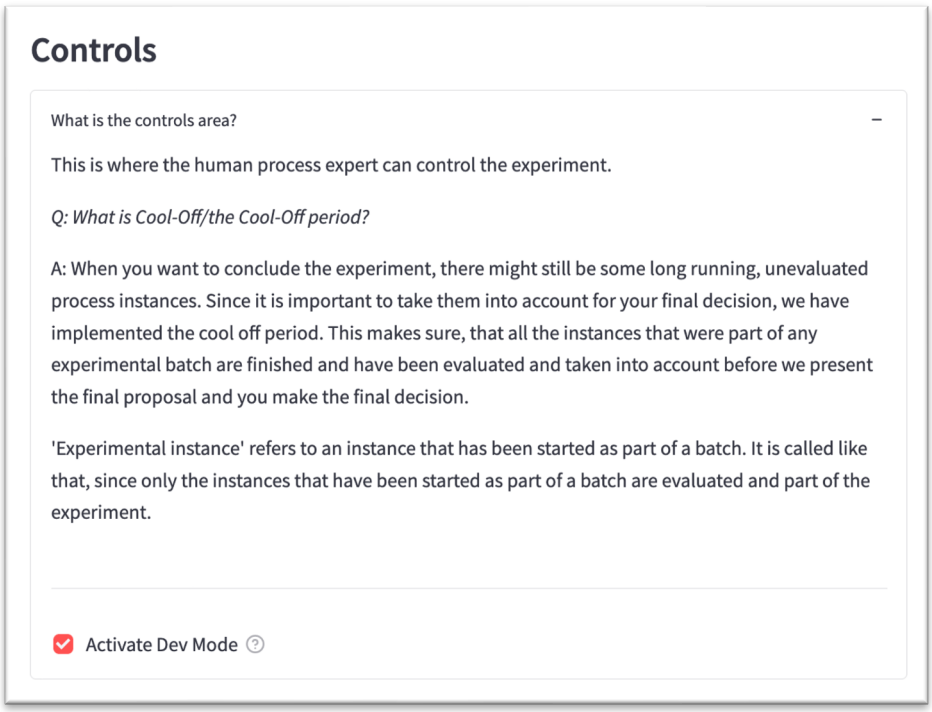
After reviewing the batch policy proposal, the human expert can either submit the batch policy as proposed or make modifications and then submit it. If the batch policy proposal is not the first, the expert may also decide to finish the experiment and enter the cool-off period (more below).
So, for now, either keep the naive policy or adjust it and hit submit.

2. Batch is finishing

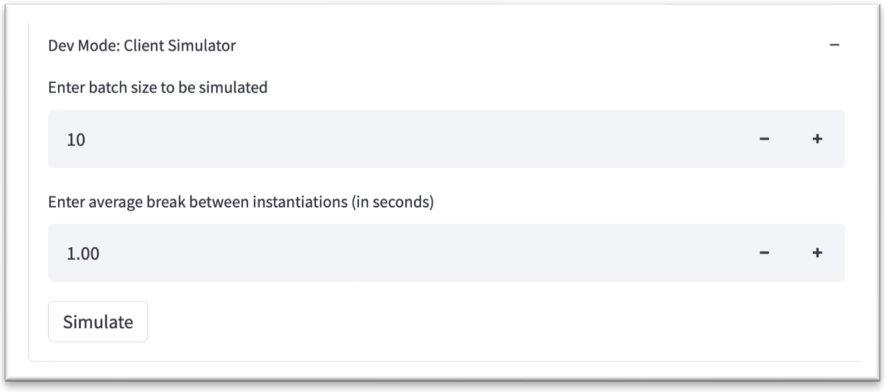
After submitting a batch policy, the next one will only be available once all the instances of the last batch have been started:



The batch of instances is finished by incoming process instantiation requests to the *HITL-AB-BPM* backend. Since these don't exist for development purposes, we have added a client simulator. One can access it by activating the development mode in the expandable help box in the *Controls* section:



Then open up the client simulator box and simulate as many instances as you have set as your batch size.



Afterward, you can look at the new batch policy proposal and start another experimental batch or end the experiment and enter the cool-off period. All batch policies except the first, naive one, also have an *End Experiment/Start Cool-Off* button.

3. Analyze results

You can also, at all times, have a look at the data regarding the already finished experimental instances in the *Data* of the *Dashboard* area section. The information there is displayed with varying granularity and can even be downloaded as a CSV for further analysis.

Experiment Metadata

What is shown here? +

Refresh

Process name: helicopter-license

Start of experiment: Tue, 15 Mar 2022 13:05:54 GMT

State: Done, Experiment ended normally

Customer categories: gov-public

Default/old version: a

Winning versions:


```
[
  0 : {
    "customer_category" : "gov"
    "winning_version" : "b"
  }
  1 : {
    "customer_category" : "public"
    "winning_version" : "a"
  }
]
```

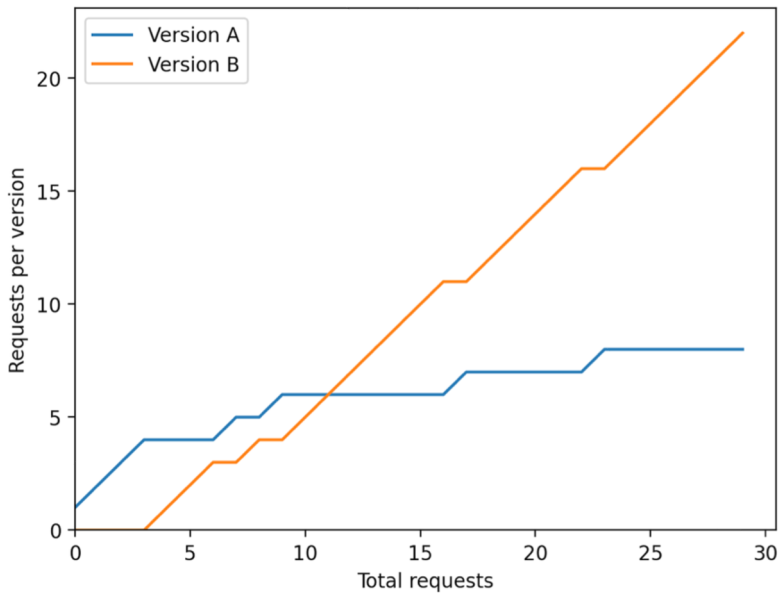

End of experiment: Tue, 15 Mar 2022 13:47:11 GMT

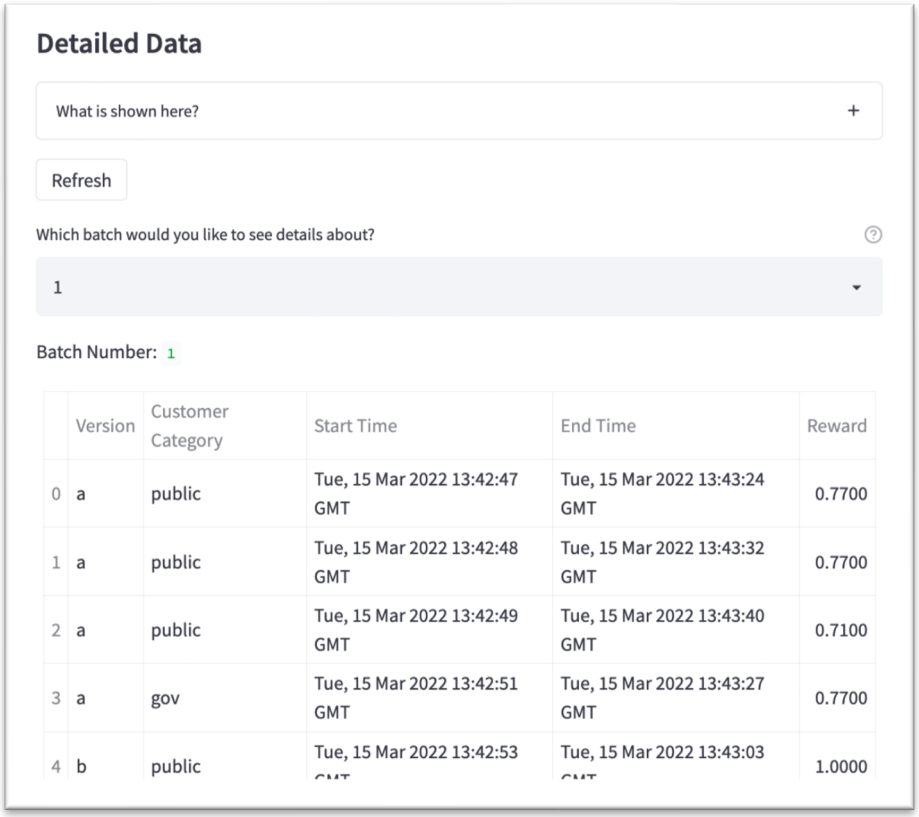
Aggregate Data

What is shown here? +

Refresh

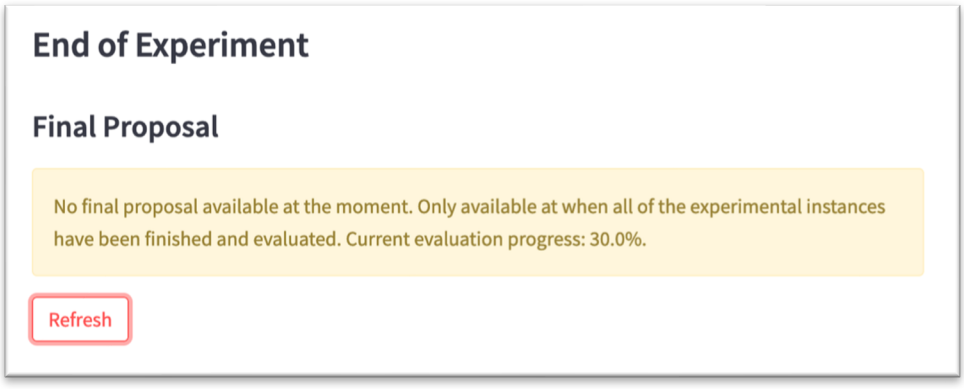
Version	Number Started	Number Finished	Mean Duration (sec)	Mean Reward
a	8	8	51.52	0.64
b	22	22	22.09	0.9





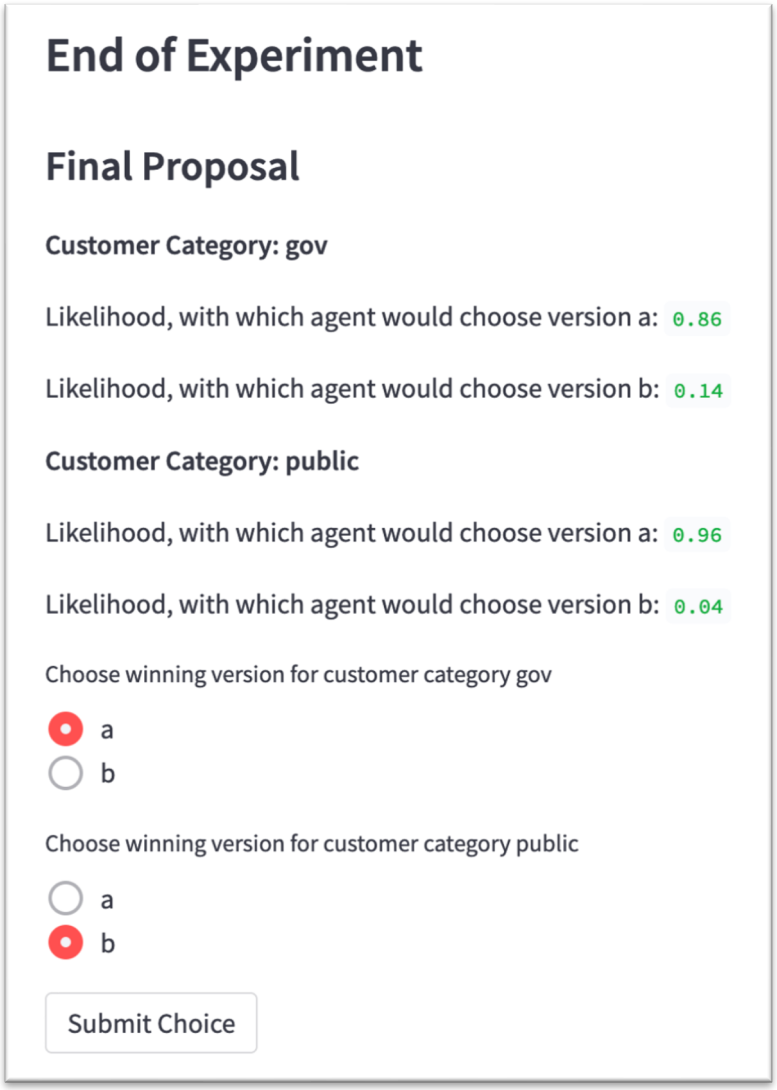
4. Cool-Off Period

After finishing the experiment, there may still be open instances. In that case, you would get a hint at the bottom of the application that it will still take some time to finish the cool-off period.



5. Set final policy

After all experimental instances have been finished, the human expert can set a final policy. The instance router will follow this policy for all future incoming process instantiation requests for that process.
To help make a decision, the RL agent will make a final policy proposal. Furthermore, the user can always refer to the *Data* section for more information.



6. Manual decision

The user also has the option to make a manual decision for version A or B at any point. This is meant as an "emergency exit" in case an experiment has gone awry. One can find this option in the *Controls* section.