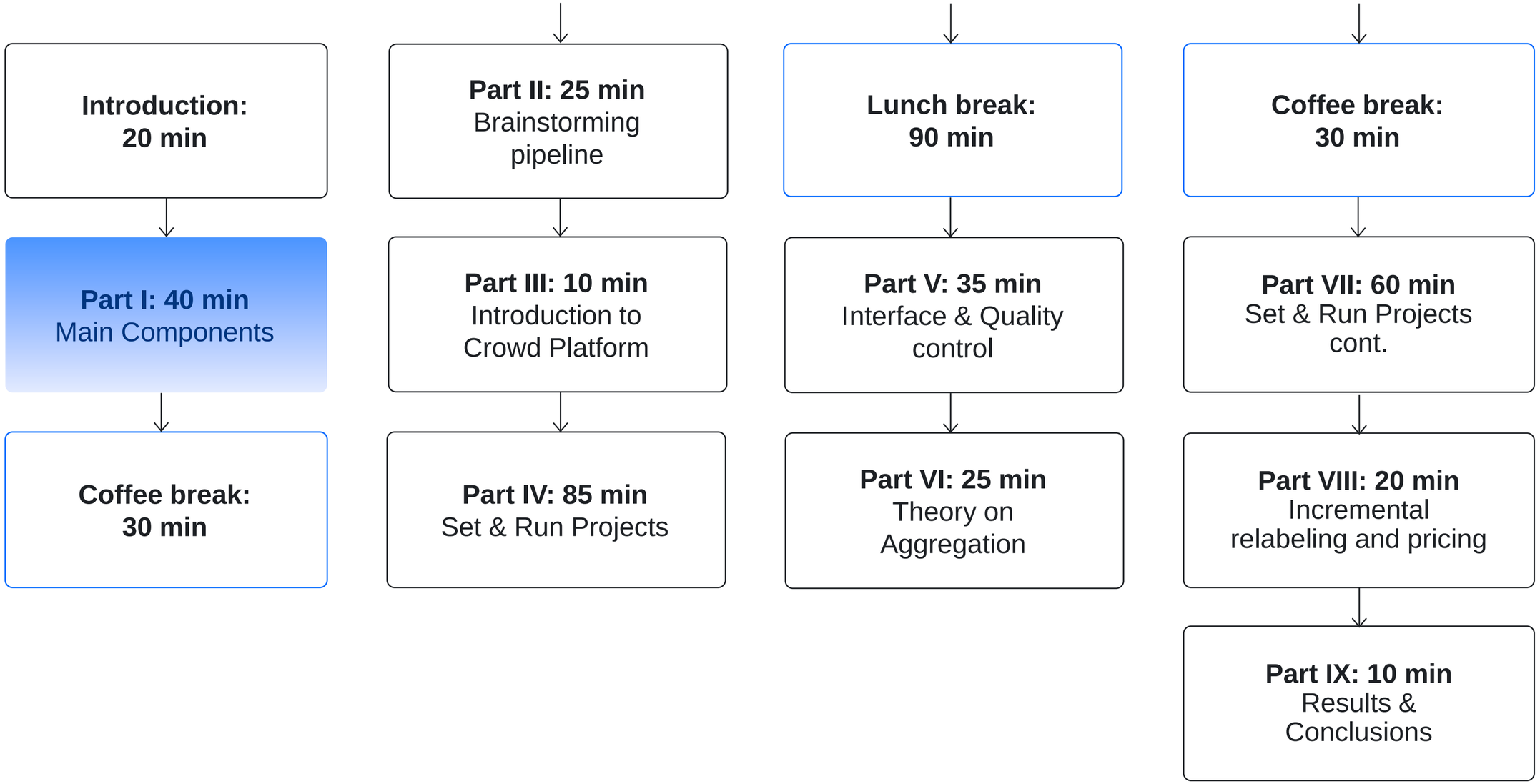
Part I

Main components of data collection via crowdsourcing

Olga Megorskaya,
CEO, Toloka

Tutorial schedule



Main components for effective crowdsourcing

Decomposition

Instruction

Task interface

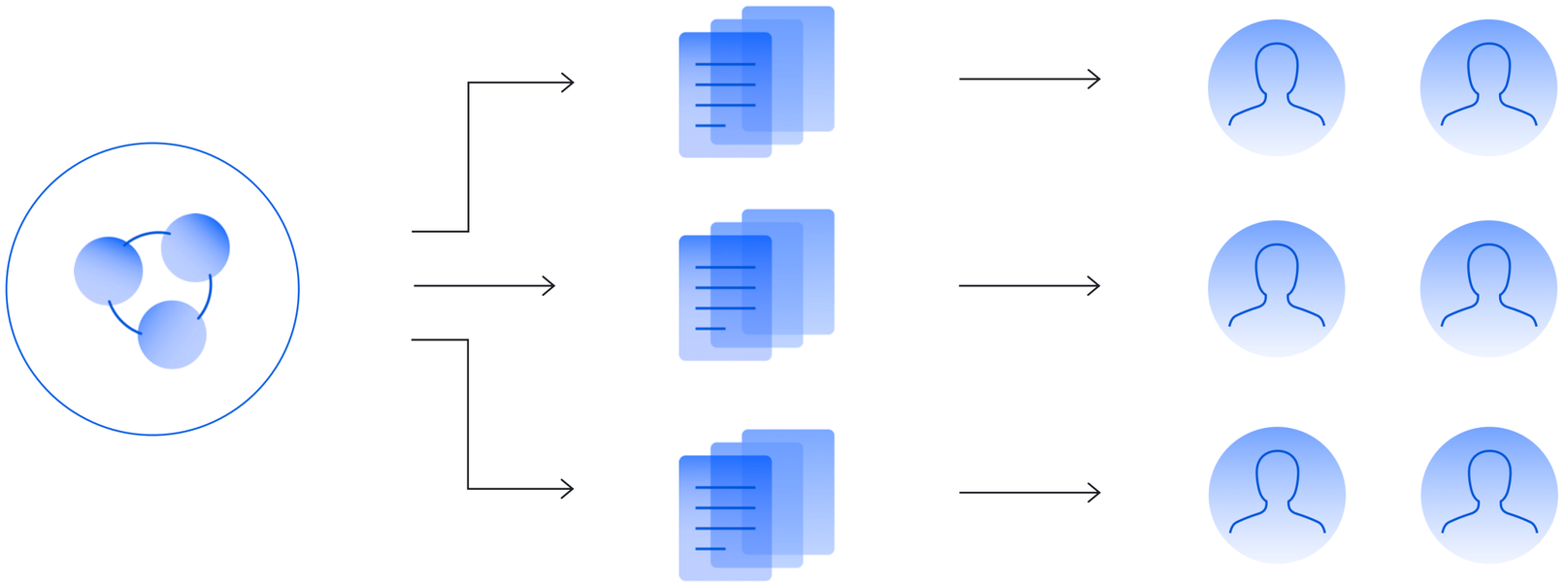
Quality control

Aggregation

Incremental
relabelling
& pricing

Decomposition

Decomposition



A big task

Projects with microtasks
of different type

Cloud
of performers

Decomposition: why?

- Performers are usually non-specialists in your specific task
- The simpler a single task is:
  - The more humans can perform your task
  - The easier its instruction
  - The better quality of performance
- A way to:
  - Distinguish tasks with different difficulty
  - Control and optimize pricing
  - Control quality by post verification

Decomposition: when?

- If
  - Your task requires an answer selected among more than 3-5 variants
  - Your task has a long instruction hard to read
- Then your task requires decomposition

Case of decomposition: a lot of questions

Bad practice: All questions in one task

What animal is on the photo?
- Cat
- Dog
- Rabbit
- Bear
- Whale
- Koala
- None of the above

Is its tail visible??
- Yes
- No

Is it running??
- Yes
- No

What color is it?
- White
- Black
- Brown
- Red
- Other

Where is it situated?
- On the grass
- On a tree
- On a road
- It is flying
- None of the above

Case of decomposition: a lot of questions

Good practice: Each question in a separate task

What animal is on the photo?
- Cat
- Dog
- Rabbit
- Bear
- Whale
- Koala
- None of the above

Is its tail visible??
- Yes
- No

Is it running??
- Yes
- No

What color is it?
- White
- Black
- Brown
- Red
- Other

Where is it situated?
- On the grass
- On a tree
- On a road
- It is flying
- None of the above

Case of decomposition: need to verify answers

The task: Highlight all koalas on the photo

Problem: **highlighting can be done in different ways**

Hence, it is difficult to:

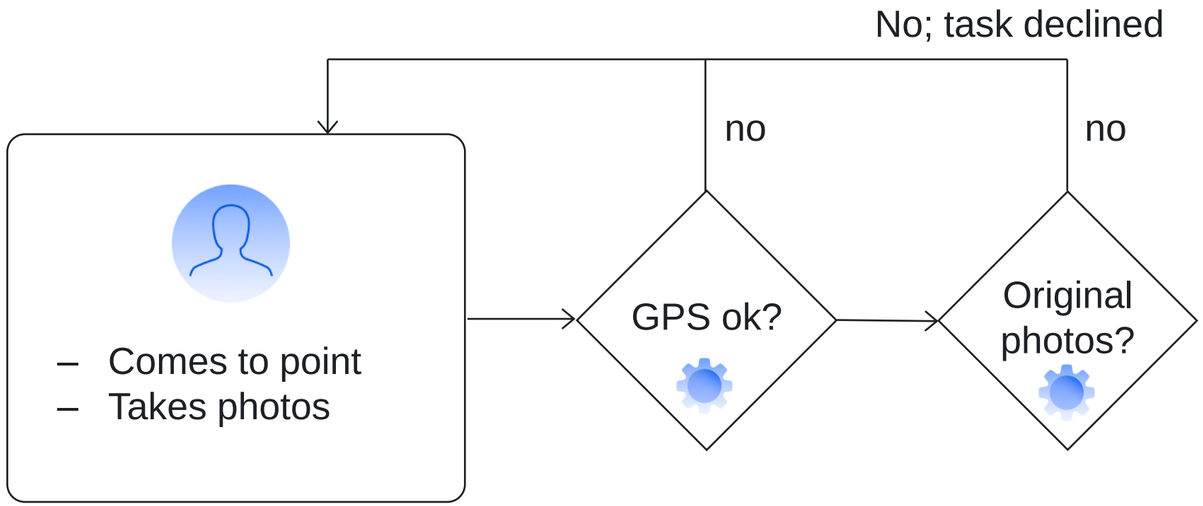
- Comparison with control answers
- Aggregation of answers from different performers

A good solution
A task for another performer:
Is the highlighting of all cars made correctly?

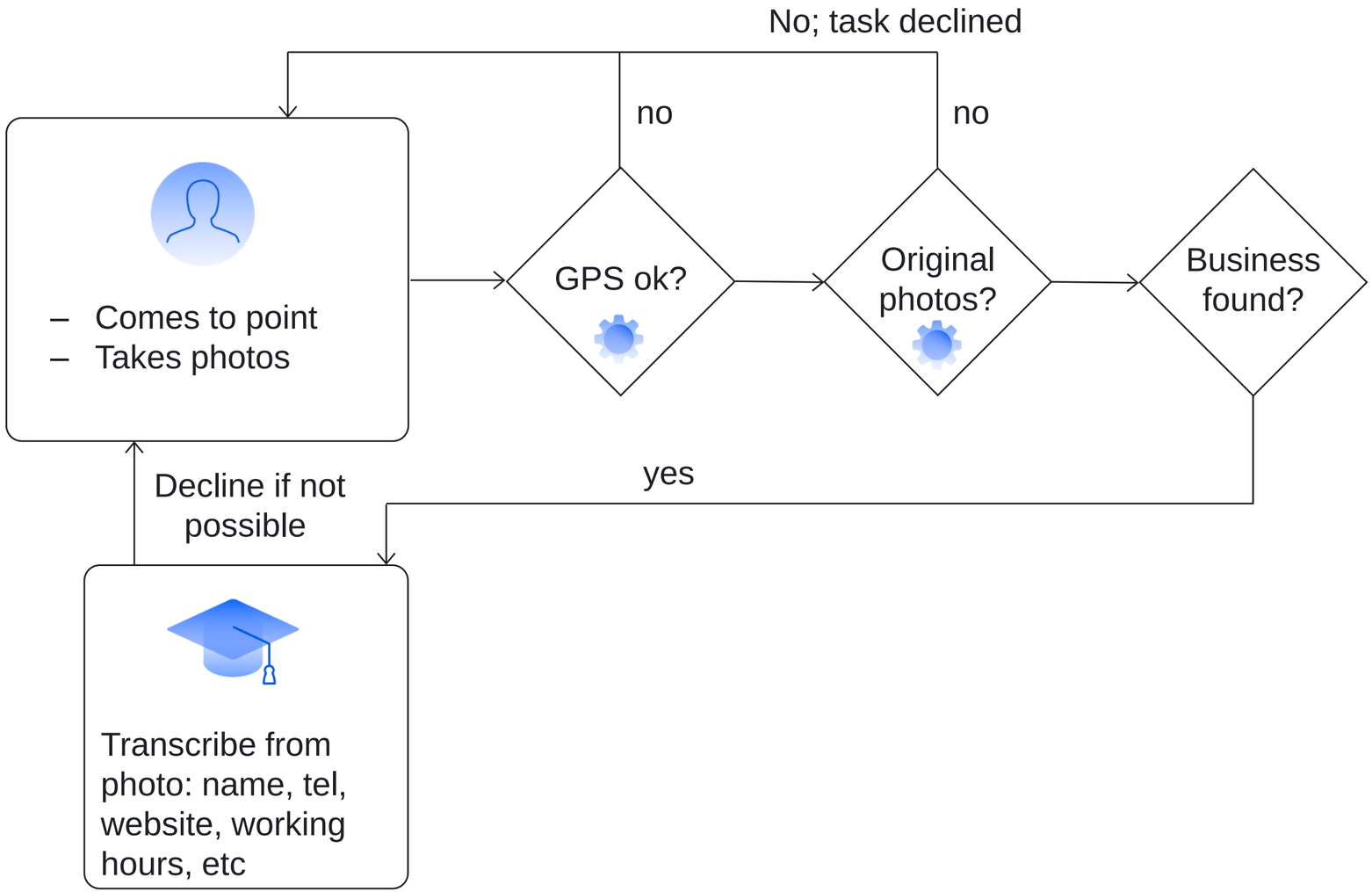
Real example: decomposition for a field survey

- Comes to point
- Takes photos

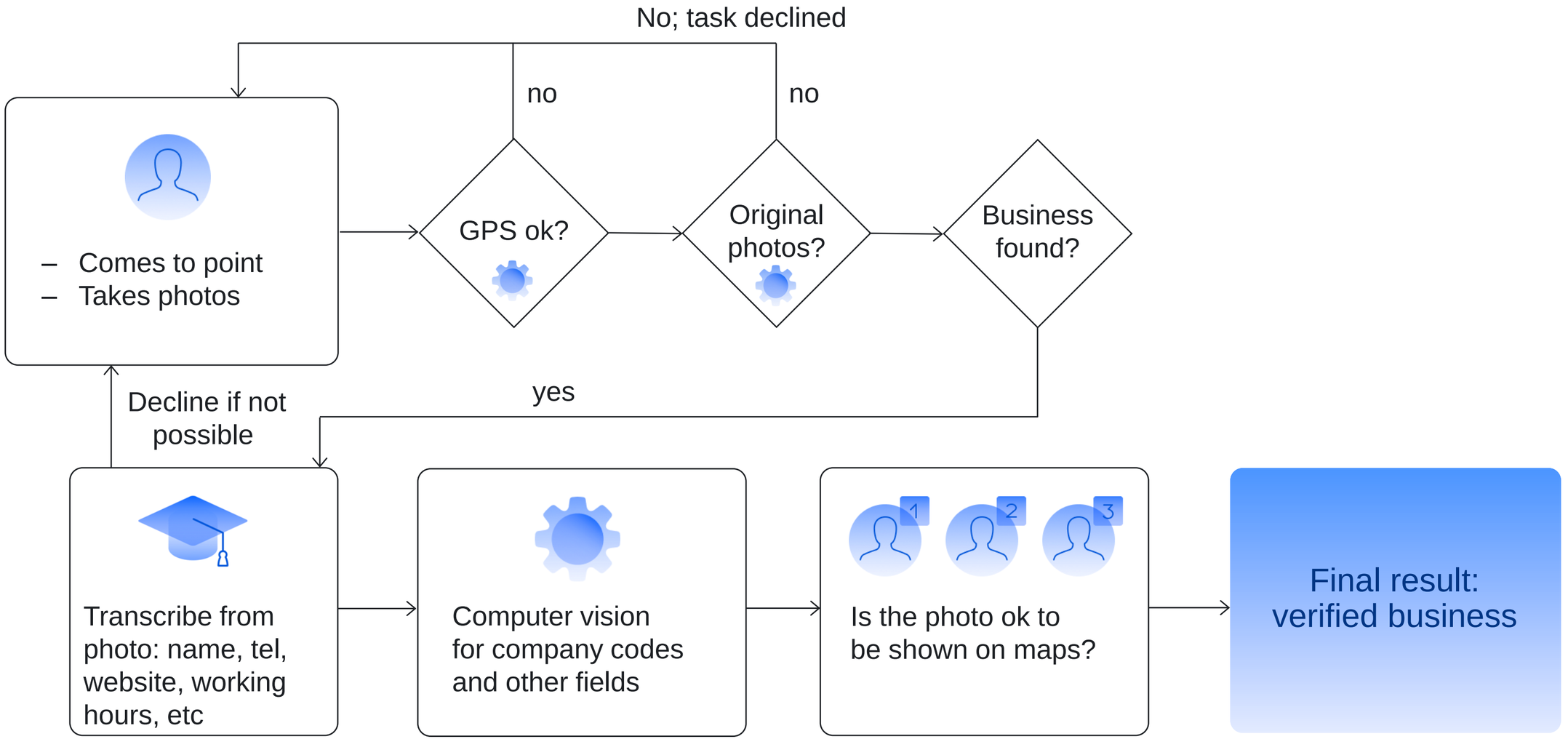
Final result:
verified business

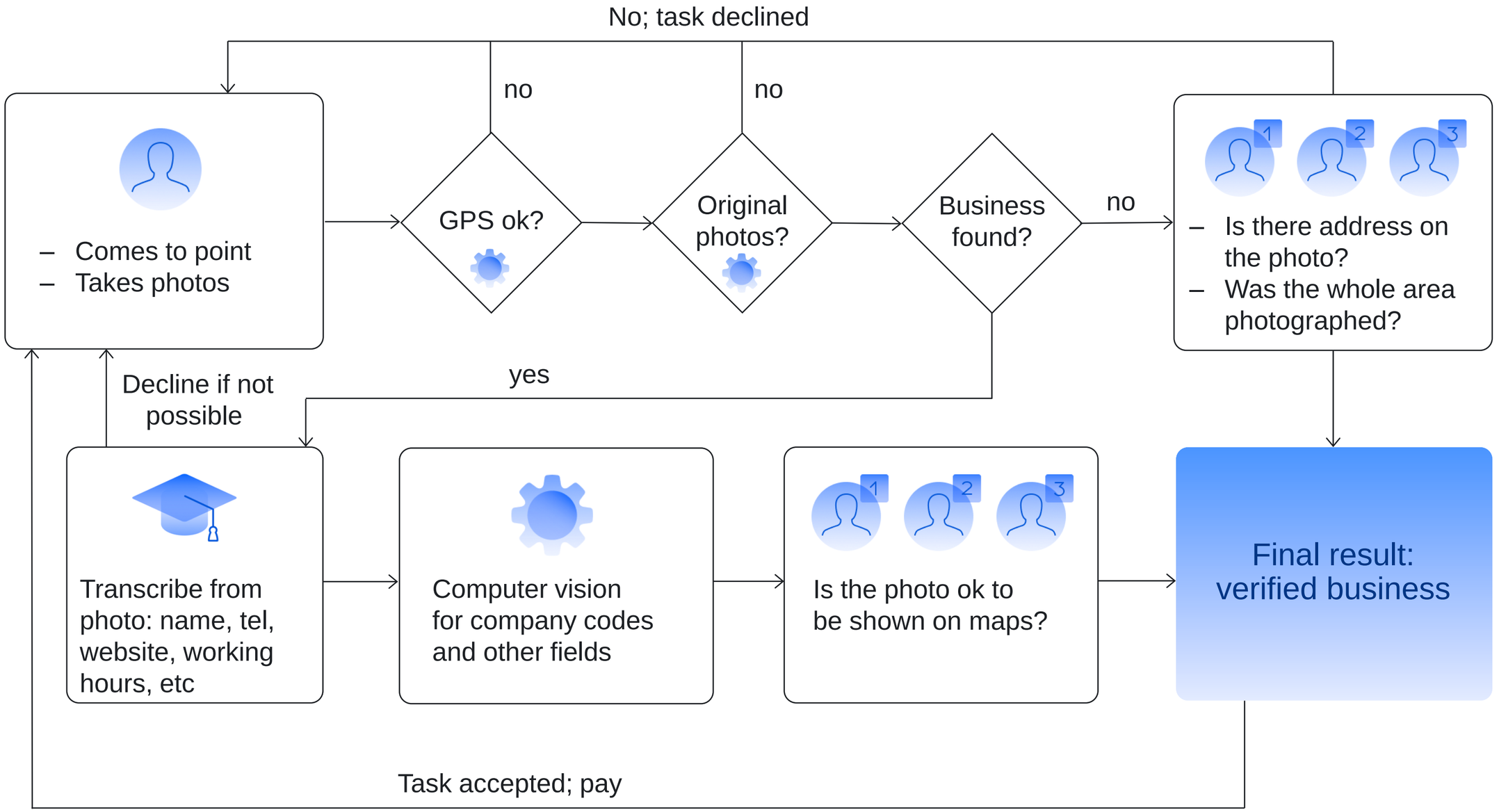


Final result:
verified business



Final result:
verified business


No; task declined
no
no
– Comes to point
– Takes photos
GPS ok?
Original photos?
Business found?
Decline if not possible
yes
Transcribe from photo: name, tel, website, working hours, etc
Computer vision for company codes and other fields
1
2
3
Is the photo ok to be shown on maps?
Final result: verified business


No; task declined
no
no
– Comes to point
– Takes photos
GPS ok?
Original photos?
Business found?
no
– Is there address on the photo?
– Was the whole area photographed?
Decline if not possible
yes
Transcribe from photo: name, tel, website, working hours, etc
Computer vision for company codes and other fields
Is the photo ok to be shown on maps?
Final result: verified business
Task accepted; pay

Instruction

Instruction: a typical structure

- Goal of the task to be done
- Interface description
- Algorithm of required actions
- Examples of good and bad answers
- Algorithm and examples for rare cases
- Reference materials

Most pitfalls are here

Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No

OK: the answer and the task seem clear

Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No

What is the correct answer?

Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No

How to fix In the instruction: clarify what you mean under “a white cat”

Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No

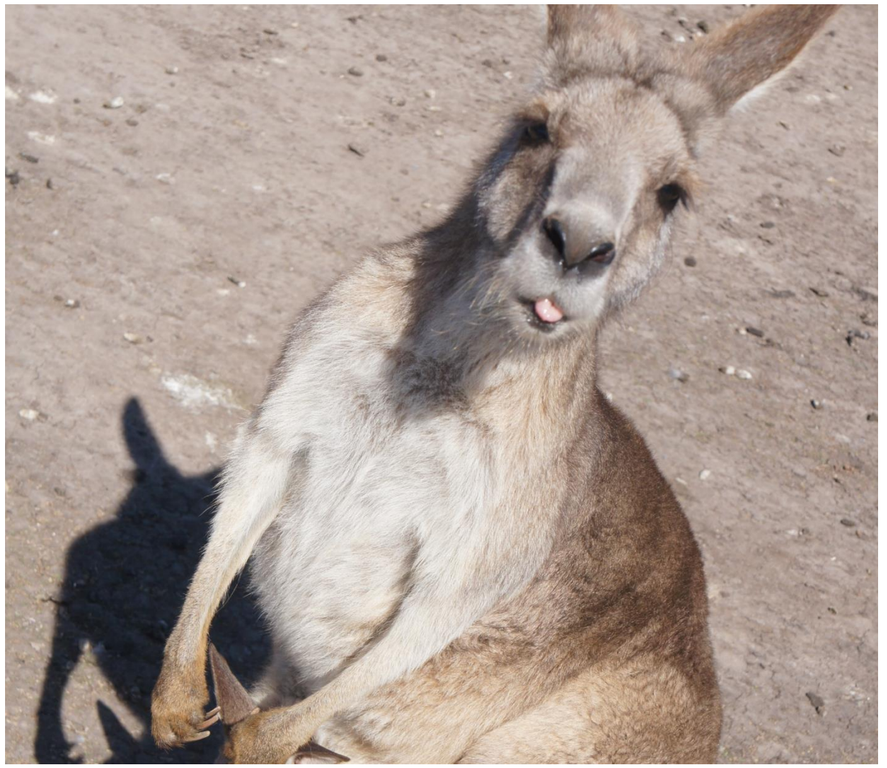
Rare case: many cats

Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No



Rare case: not a cat

Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No

404: Cannot download the image

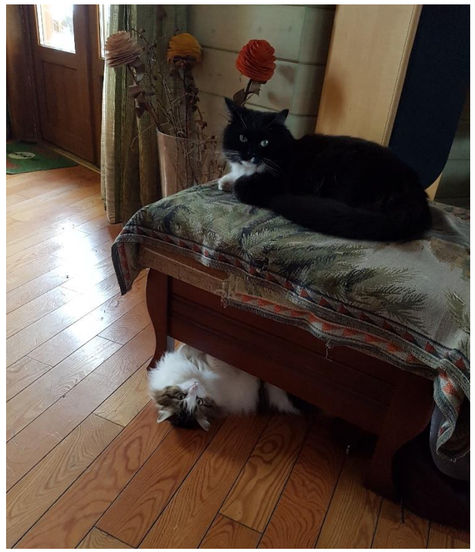
Rare case: image has not been shown

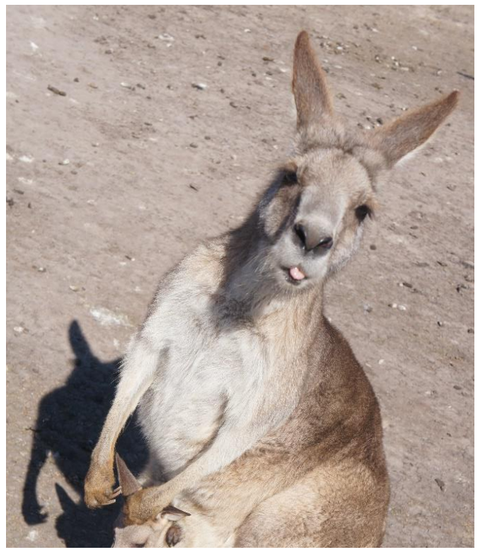
Instruction ambiguity for a rare case: example

Is this cat white?

Yes

No





404: Cannot download the image

It is difficult to predict situations of any kind, but you can:

- In the instruction: clarify what should be done in a non-standard situation
- In the interface: add a text field to allow a performer to report the case

Task interface

Task interface: summary on best practices

For faster performance

- Hot key combinations for checkboxes/radio buttons/buttons
- Reduce navigation to third-party sites
- Effective composition of a task template
- Optimal position of tasks on a page

For better quality and less errors

- Dynamic interface (show/hide input controls depending on user actions)
- Adaptive interface (good view for any device and screen resolution)
- Always test your interface (template testing)
- Dynamic validation of input data (e.g. a text is less than 3 words)

Quality control

Quality control

“Before” task performance

- Selection of performers
- Well-designed instruction

“Within” task performance

- Golden set (aka honey pots)
- Well-designed interface
- Motivation (e.g. performance-based pricing)
- Tricks to remove bots and cheaters (e.g. quick answers)

“After” task performance

- Post verification (acceptance)
- Consensus between performers and result aggregation

Selection of performers

- Filter by static properties (e.g. education, languages, citizenship, etc.)
- Filter by computed properties (e.g. browser, region by phone/IP, etc.)
- Filter by skills
  - To select proper specialization
  - To control quality level on your tasks
  - To get performers with best quality on past projects
- Educate to perform your tasks
  - Use training tasks to show how to perform tasks
  - Use exam tasks to evaluate education level

Golden set (aka honey pots)

Tasks with known correct answer shown to performers to evaluate their quality

- Distribution of answers in golden set = distribution in whole set of tasks
- But should contain rare answer variants with higher frequency
- Refresh your set of honey pots regularly to avoid bots and cheating
- Automatic golden set generation via performers:
  - Tasks with answers of high confidence (e.g. aggregation of answers from a large number of performers)

Best practices

Motivation

- Bonuses for a good quality within a period
- Gamification (e.g. achievements, leader boards, etc)
- Price depending on quality

Will be discussed in Part VIII

Tricks to remove bots and cheaters

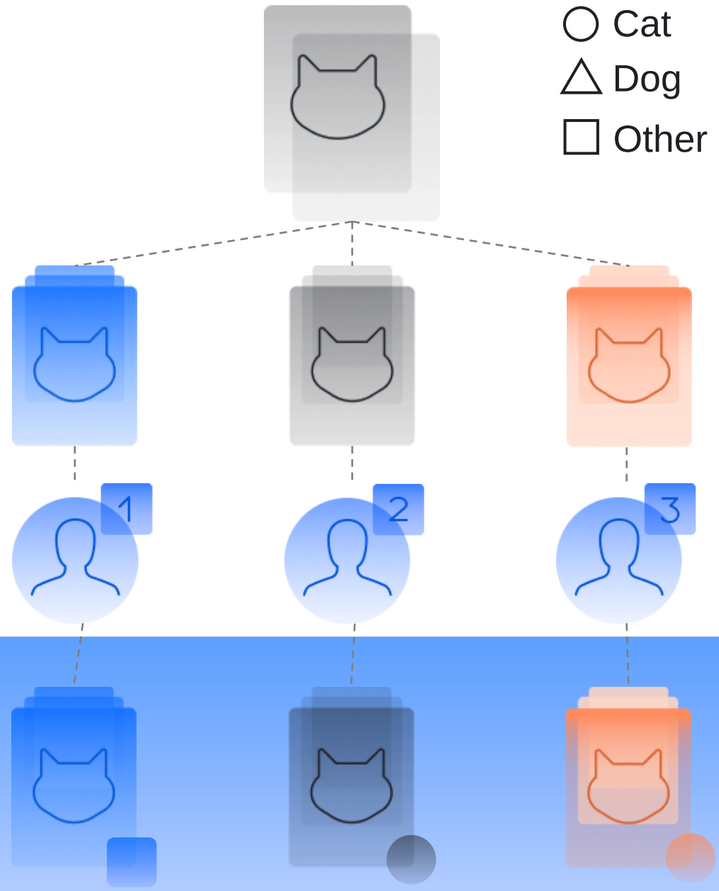
- Control fast responses
- Check whether a link has been visited
- Check whether a video has been played
- etc

Post verification (acceptance)

- A performer gets money only if his answer is accepted
  - Is used when a task is sophisticated (neither golden set nor consensus models work)
  - Can be performed on your own, but

- You can use other crowd performers via a task of different type
  - Thus, you deal with hierarchy of projects (you apply decomposition)

Consensus between performers



Upload multiple copies of each object to label

Performers assign noisy labels to objects

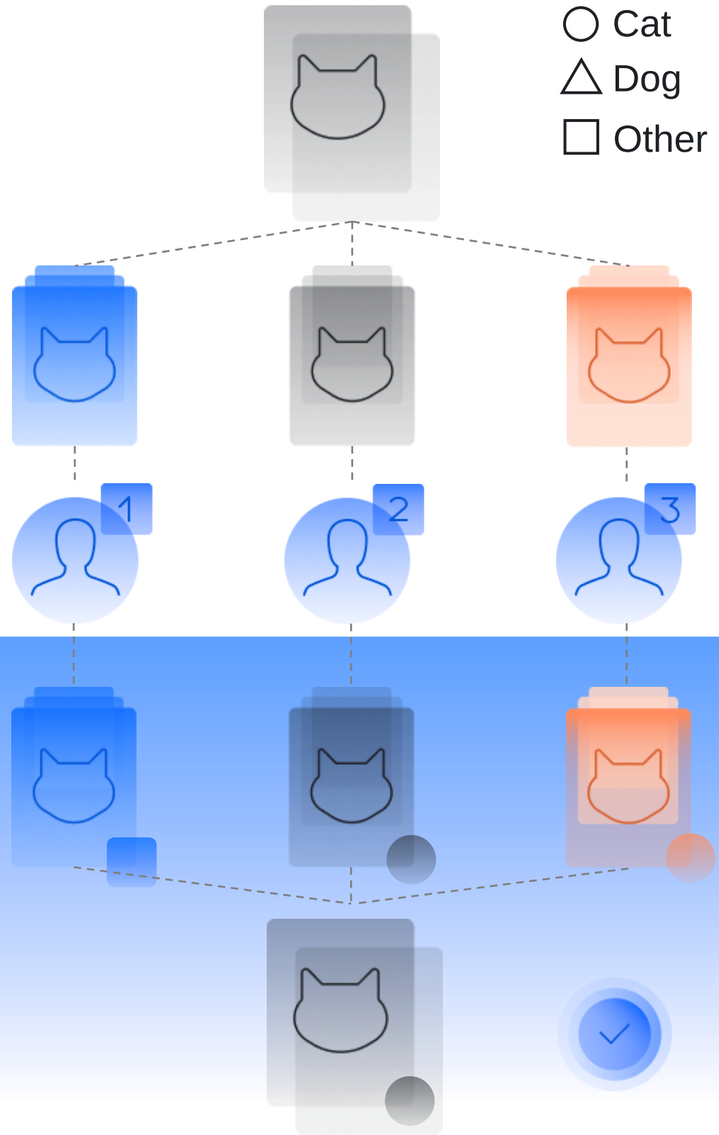
Derive quality from consensus

Works well only if most performers have good quality

Will be discussed in Part VI

Aggregation

Aggregation



Upload multiple copies of each object to label

Workers assign noisy labels to objects

Aggregate multiple labels into a more reliable one

The simplest way:

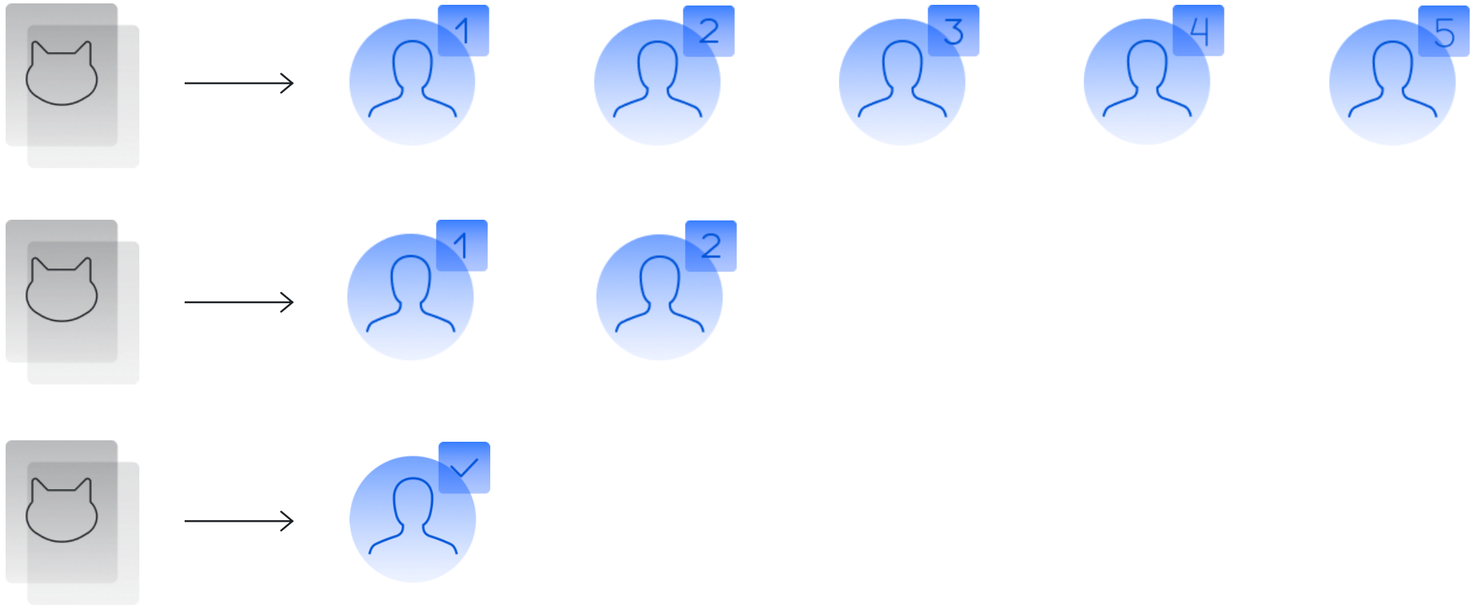
- Assign the most popular answer (Majority Vote)
- There are more sophisticated methods

Will be discussed in Part VI

Incremental relabelling & Pricing

Incremental relabelling

Obtain aggregated labels of a desired quality level using a fewer number of noisy labels



Will be discussed in Part VIII

Pricing depends on

Task design

- Payment is made per a batch of microtasks (aka a task suite)
- Time required to perform a task: control hourly wage

Market economy aspects

- The lower supply of performers is (e.g. due to specific skills), the higher price
- How quickly do you need accomplished tasks (latency)?

Result quality

- Incentivize better performance by a quality-dependent price

Will be discussed in Part VIII

Good decomposition is the key to success

IF

Good decomposition

THEN

Simple instruction

Easy to use task interface

Performers do tasks with better quality

Easy to control quality

Standard aggregation models work well

Easy to control and optimize pricing

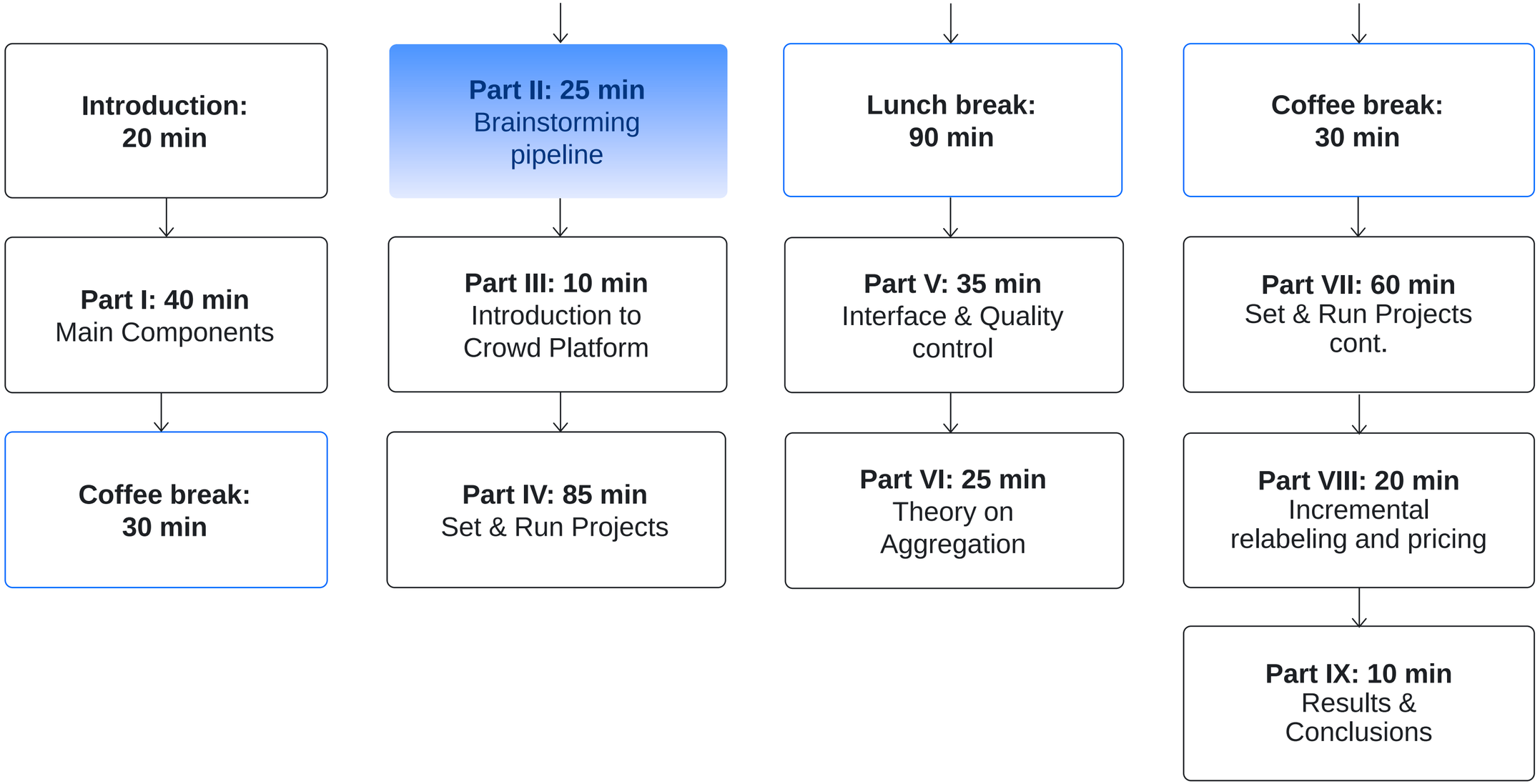
Part II

Label collection projects to be done

Daria Baidakova,
Project Manager

Toloka

Tutorial schedule



Practice session

Our practice session will consist of three parts:

Part I
(now)

Think and discuss in groups how you would design a crowdsourcing pipeline

Part II
(in 35 min)

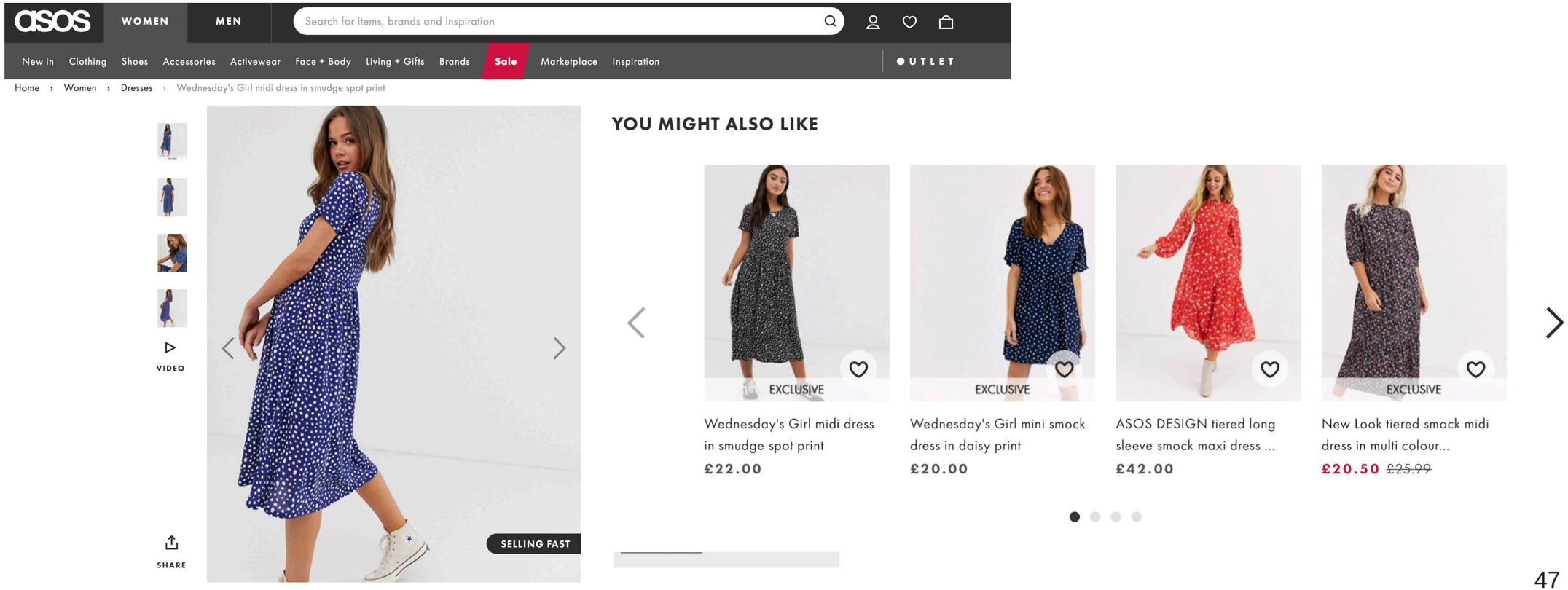
Run the best-practice pipeline on a real crowd on Toloka

Part III
(in 145 min)

Complete the pipeline on Toloka

Practice session: scope

Imagine that you develop a machine learning pipeline to help improve search quality at an online store to find substitutes



Practice session: scope

Imagine that you develop a machine learning pipeline to help improve search quality at an online store to find substitutes

- You have a dataset of pictures with people wearing different clothes
- You need to find a better substitute for the initial item in an image
- These collected data will further be used to train a search algorithm

This is your goal for the practice session of our tutorial

Dataset under study: pictures of people wearing different clothing items

Items to be matched in photos

Each photo may contain clothing items of different types, for example:

- Hats
- Shirts
- Jackets
- Coats
- Jeans
- Pants (trousers)
- Bags
- Sunglasses
- Other items

During your practice:

Choose one type of items you want to find substitutes for in the photos

For example: Shoes

Formal setup: find the best substitute item

- Each clothing item of a selected type in each photo from the dataset needs to be matched by a substitute item
- **Let us do it via crowdsourcing**

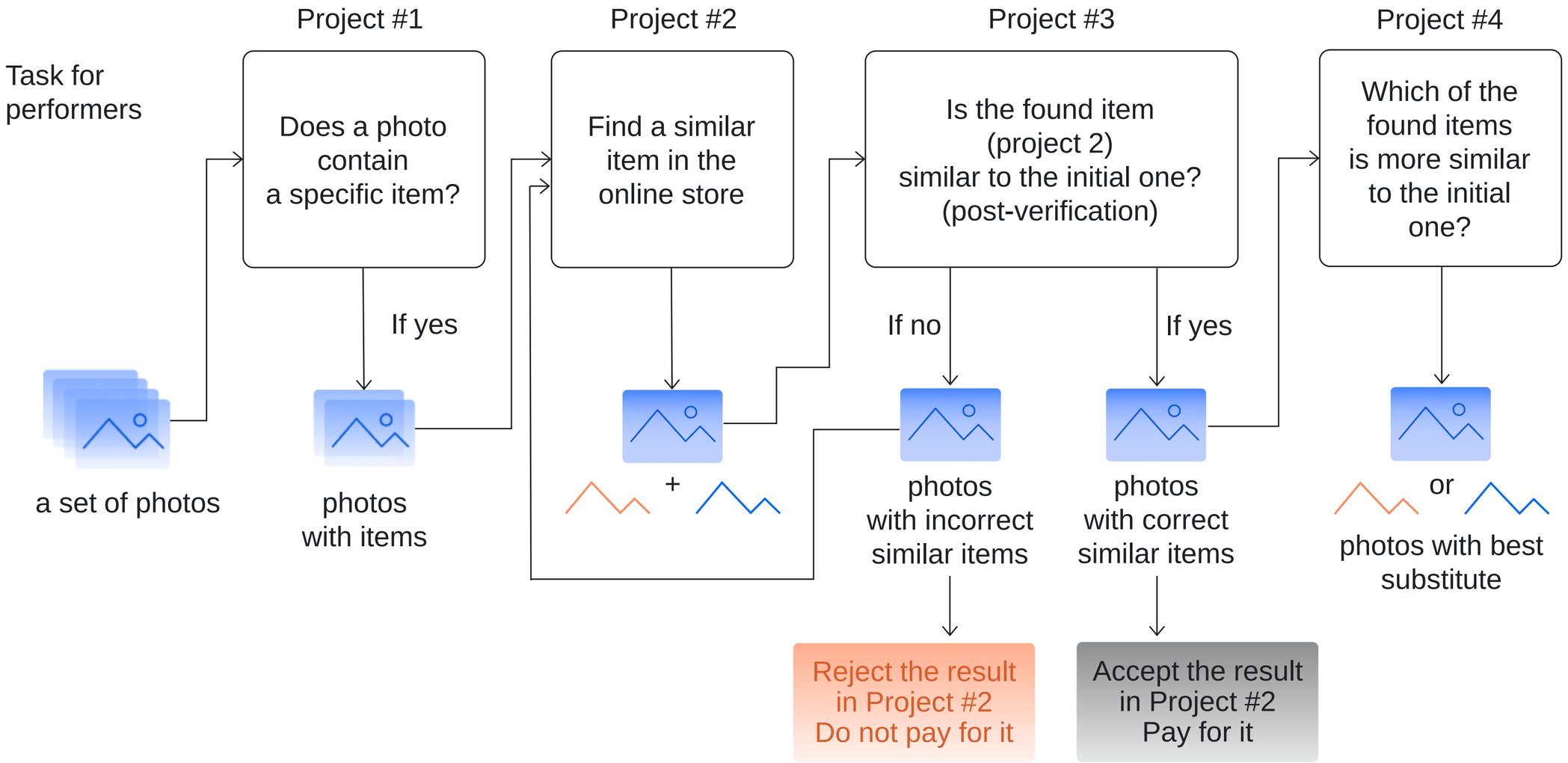
Example: we decide to find the best substitute for the shoes, so our pipeline would be like..

During your practice:

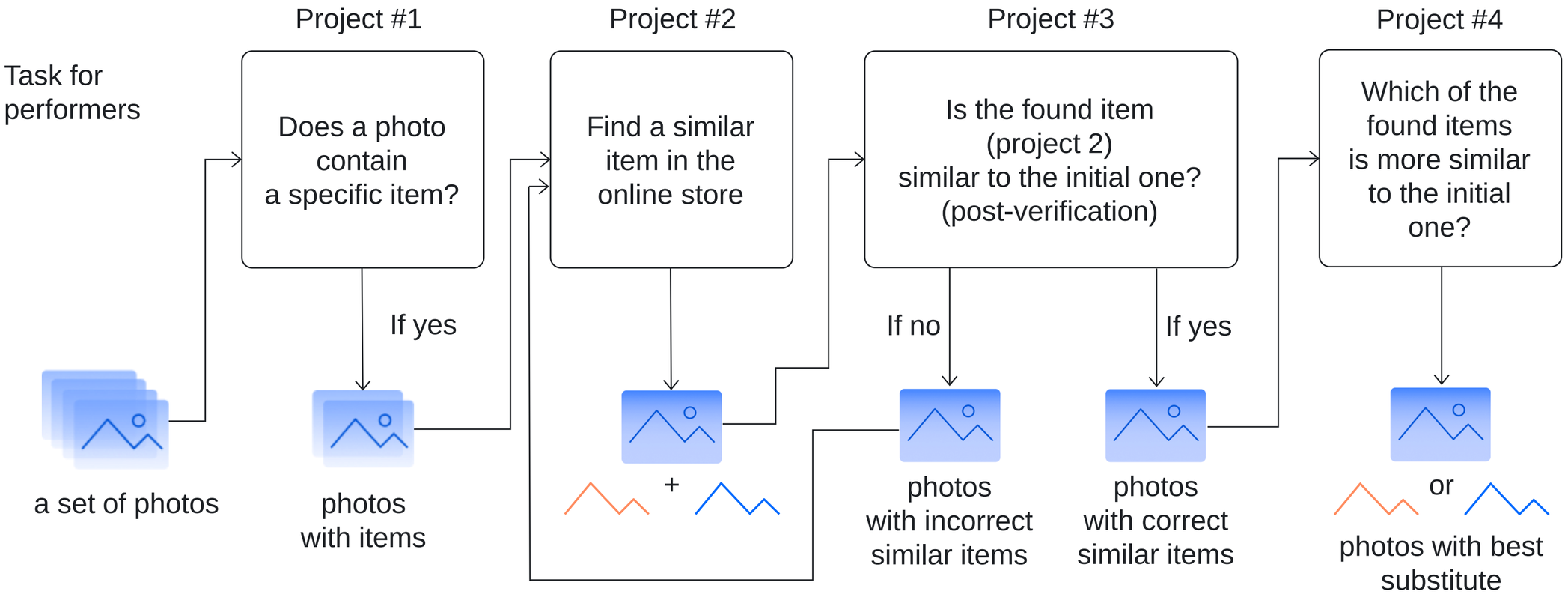
Discuss in groups how you would design a crowd pipeline to find the best substitute!

Suggested pipeline

We suggest the following pipeline



We suggest the following pipeline



During the practical session we will help you implement and run this pipeline

Project #1: Filter out photos without objects

Task

- Does a photo contain an item of desired type?

Key setting

- Type: classification
- Quality control: golden set
- Overlap: 3 answers per photo
- Pay: $0.01 per a suite of 10 photo

Why?

- Save money: no need to process further photos without desired objects

Are there **shoes** in the picture?

Yes No Picture not found

Project #2: Searching for similar items on the online store

Task

- Find a similar item on the internet

Key setting

- Type: product photo search
- Quality control: post verification
- Overlap: 3 answers per photo
- Pay: $0.02 per 1 photo

Peculiar properties

- Hard to use golden set and consensus
- Results will be verified in Project #3

Find the same **shoes** on ASOS

ASOS

Shoes must be the same color and the same style.

Paste the link here

Upload the image here. The image should show the shoes you found.

Project #3: Accept correctness of items found

Task

- Does an image contain a requested item?

Key setting

- Type: classification
- Quality control: consensus
- Overlap: 3 answers per photo
- Pay: $0.01 per a suite of 10 photo

Why?

- Need to verify the results obtained from Project #2

Project #4: Decide which substitute works best

Task

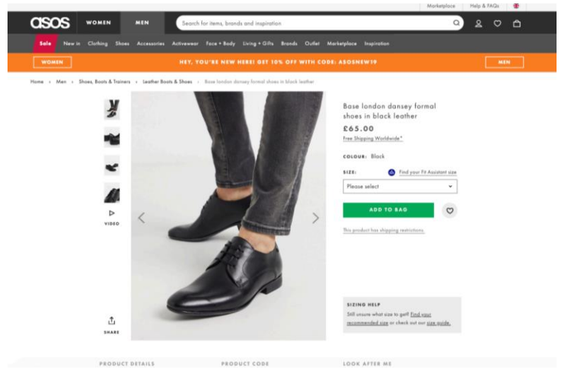
- Which of the items is similar to the initial one?

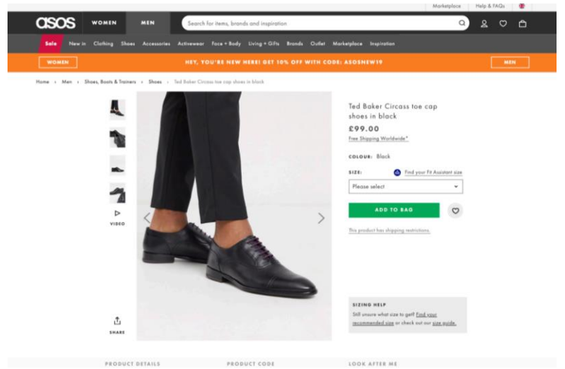
Key setting

- Type: side-by-side image comparison
- Quality control: consensus
- Overlap: 3 answers per photo
- Pay: $0.01 per a task suite of 10 photo

Why?

- Need to understand which substitute fits best



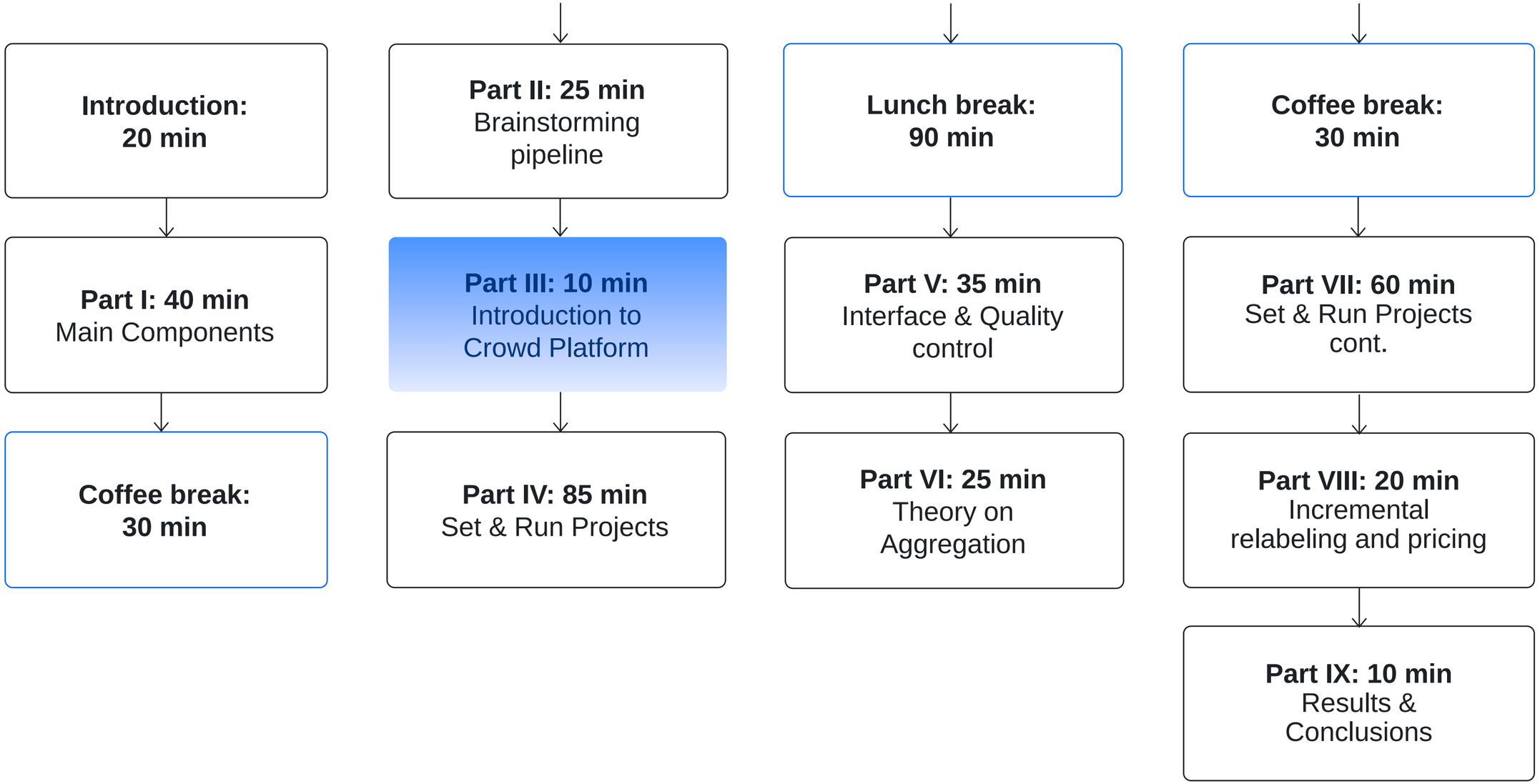


Part III

Introduction to Toloka for requesters

Evfrosiniya Zerminova,
Head of Data Analysis and Research Group, Toloka

Tutorial outline



Key types of instances in Yandex.Toloka

Project

- Defines the structure of tasks
- Defines how to perform them

Configure in a project

- Input and output data types
- Task interface
- Task instruction

Pool

- Is a batch of tasks
- Defines access of performers

Configure in a pool

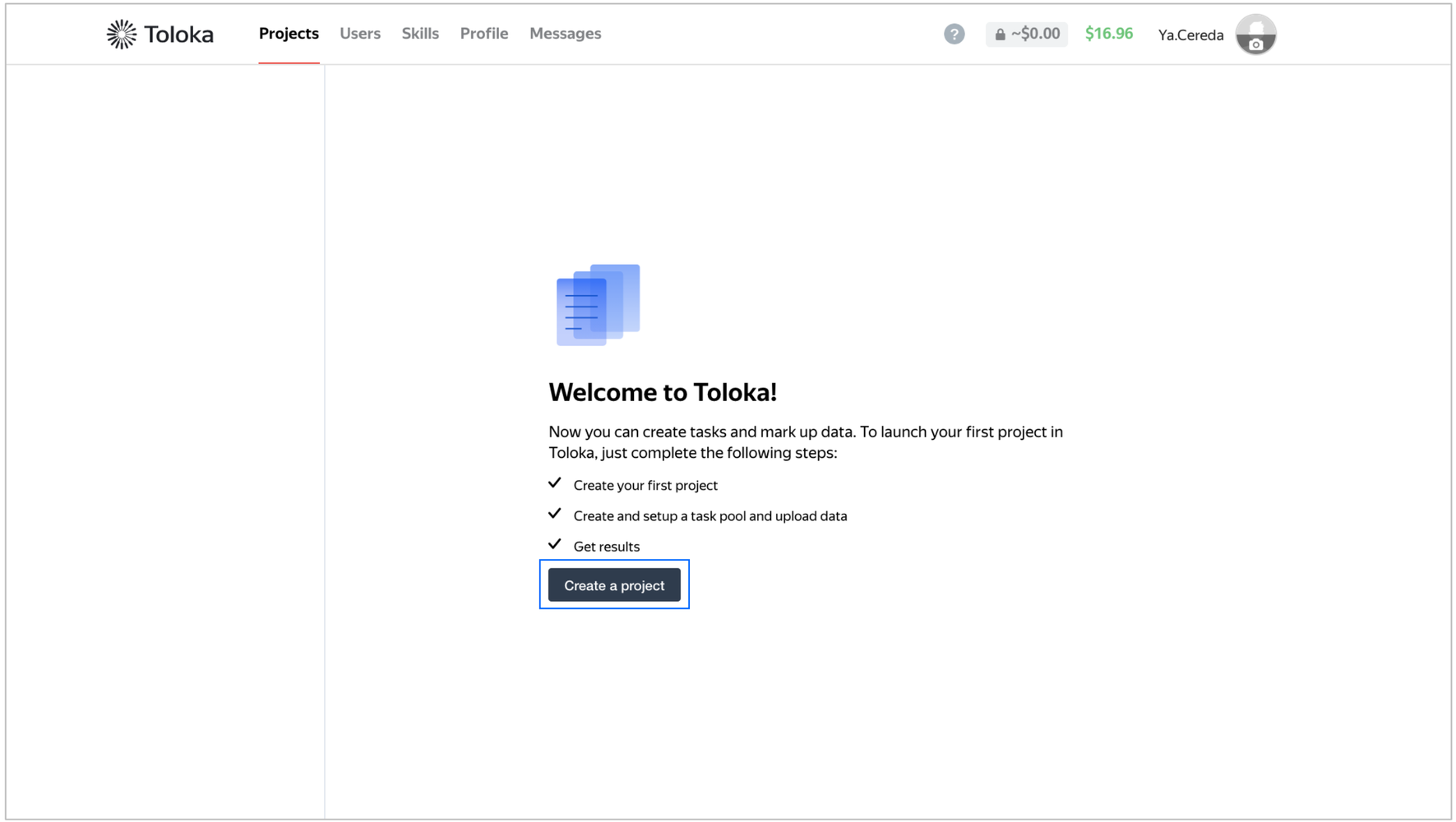
- Performer filters
- Quality control mechanisms
- Overlap settings

Task

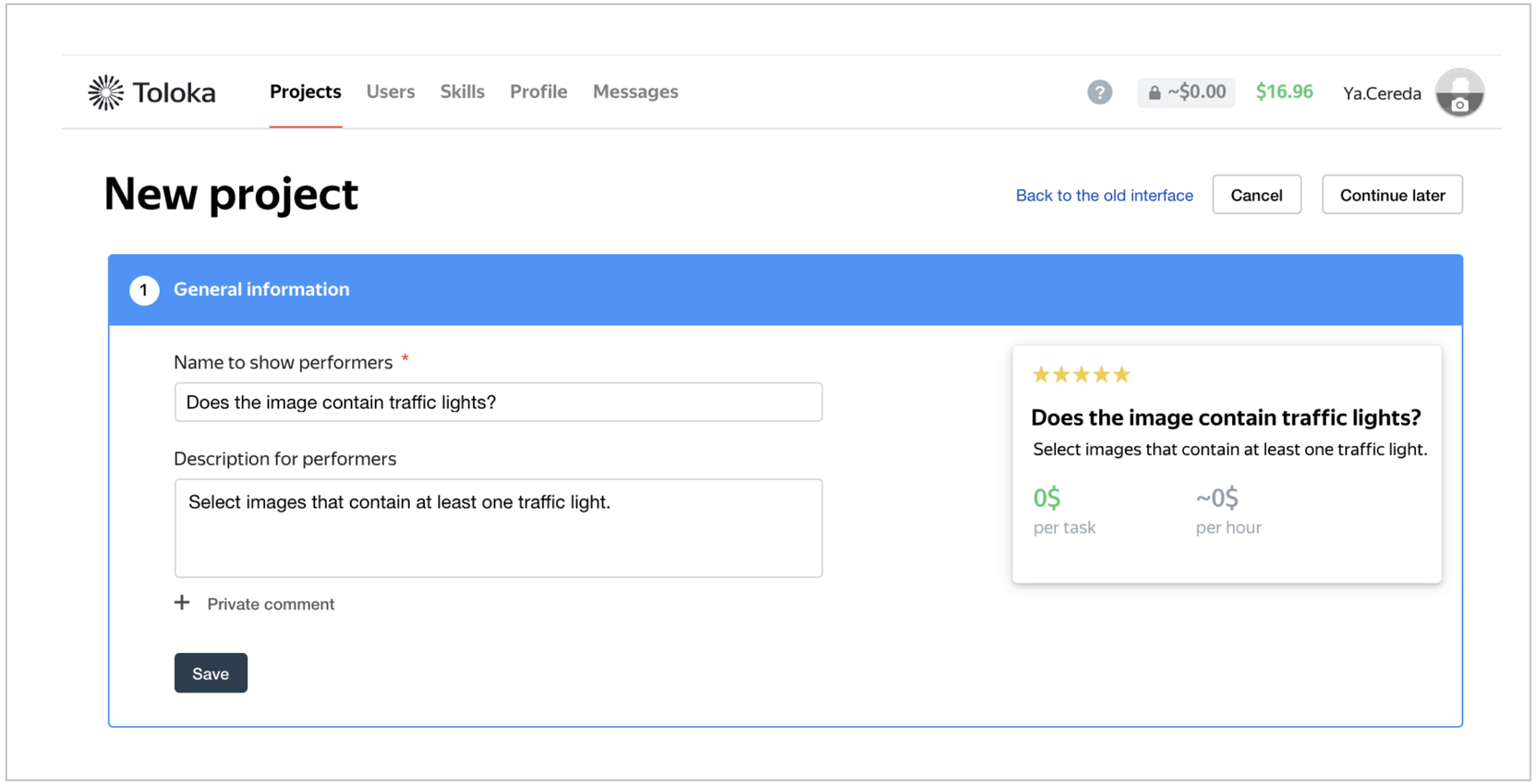
- A particular input data
- Results for it from performers

Project:
creation & configuration

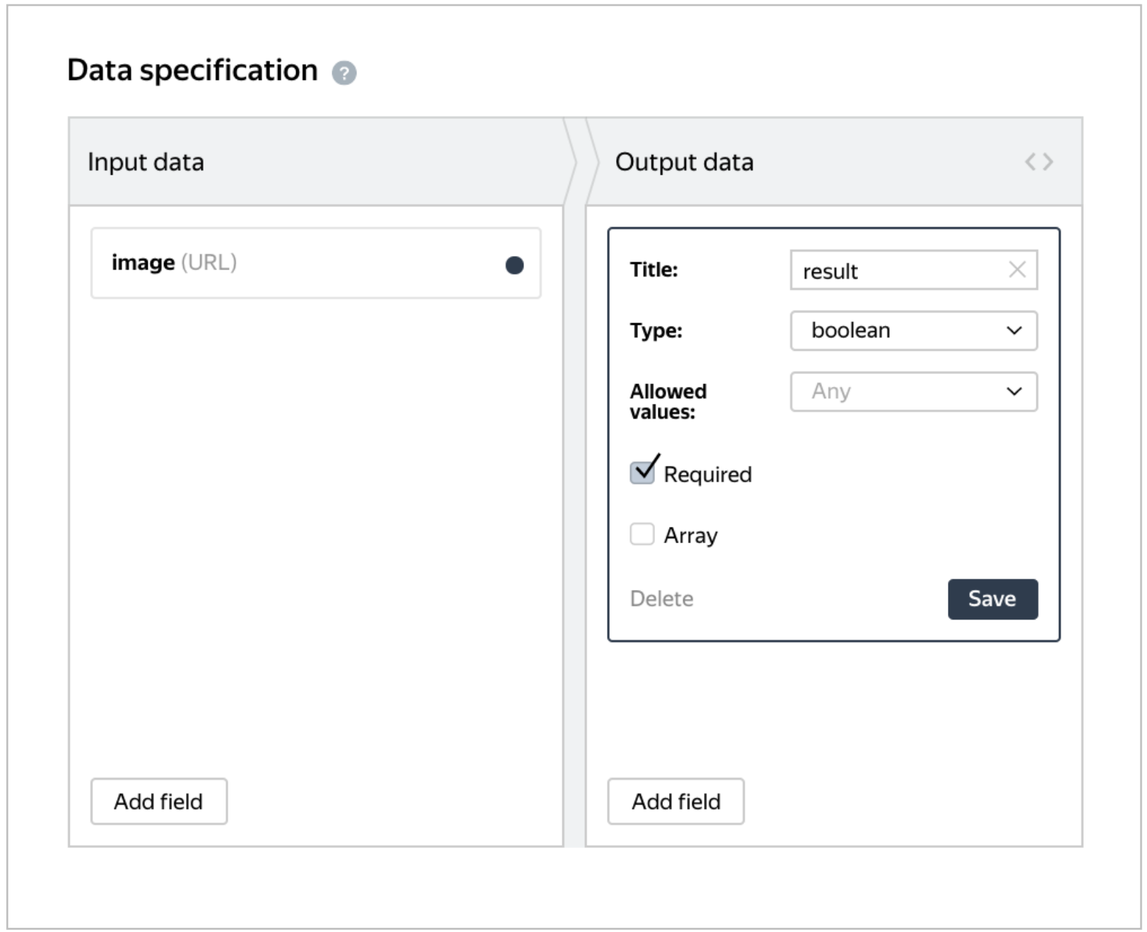
Project: creation



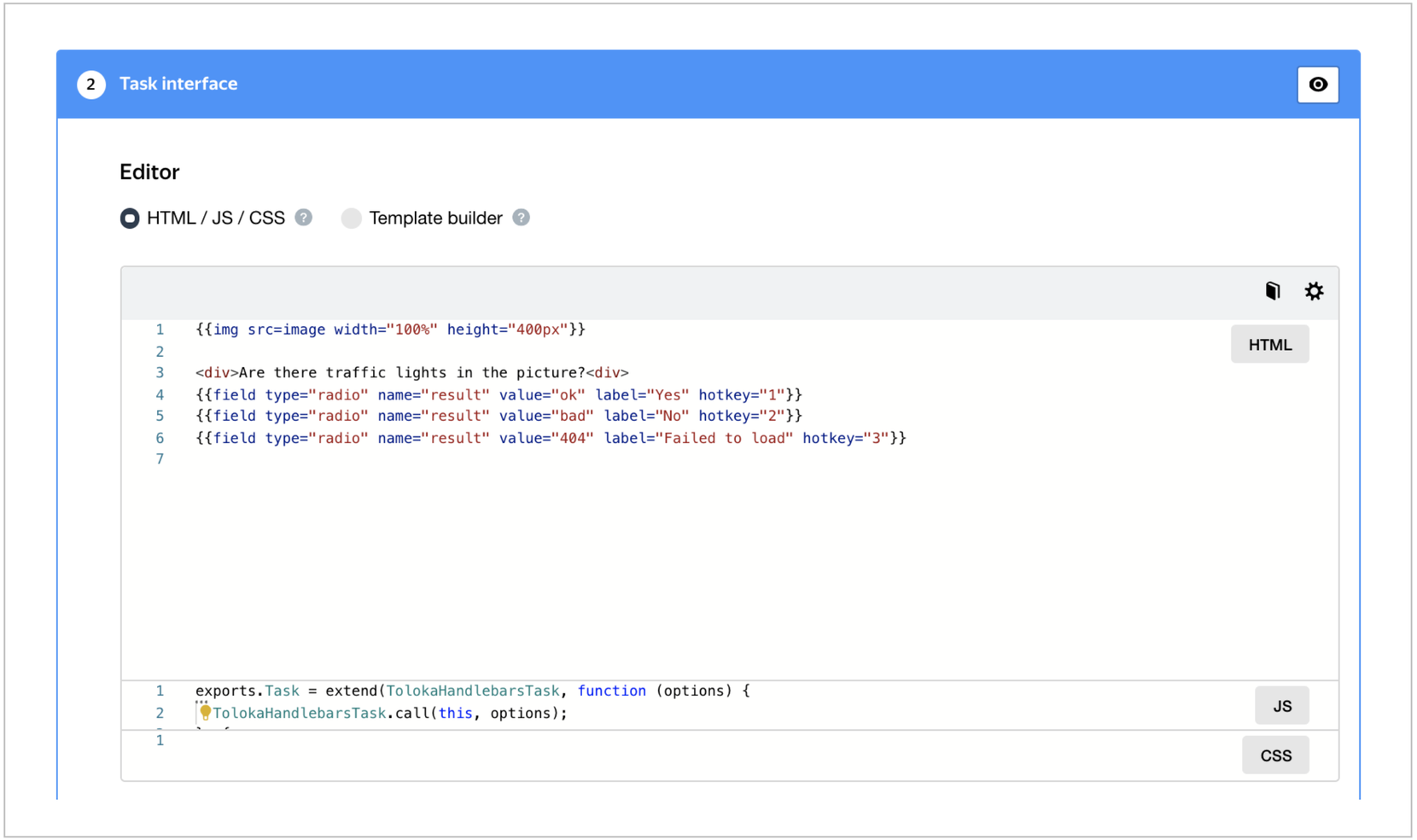
Project: configure name and instruction



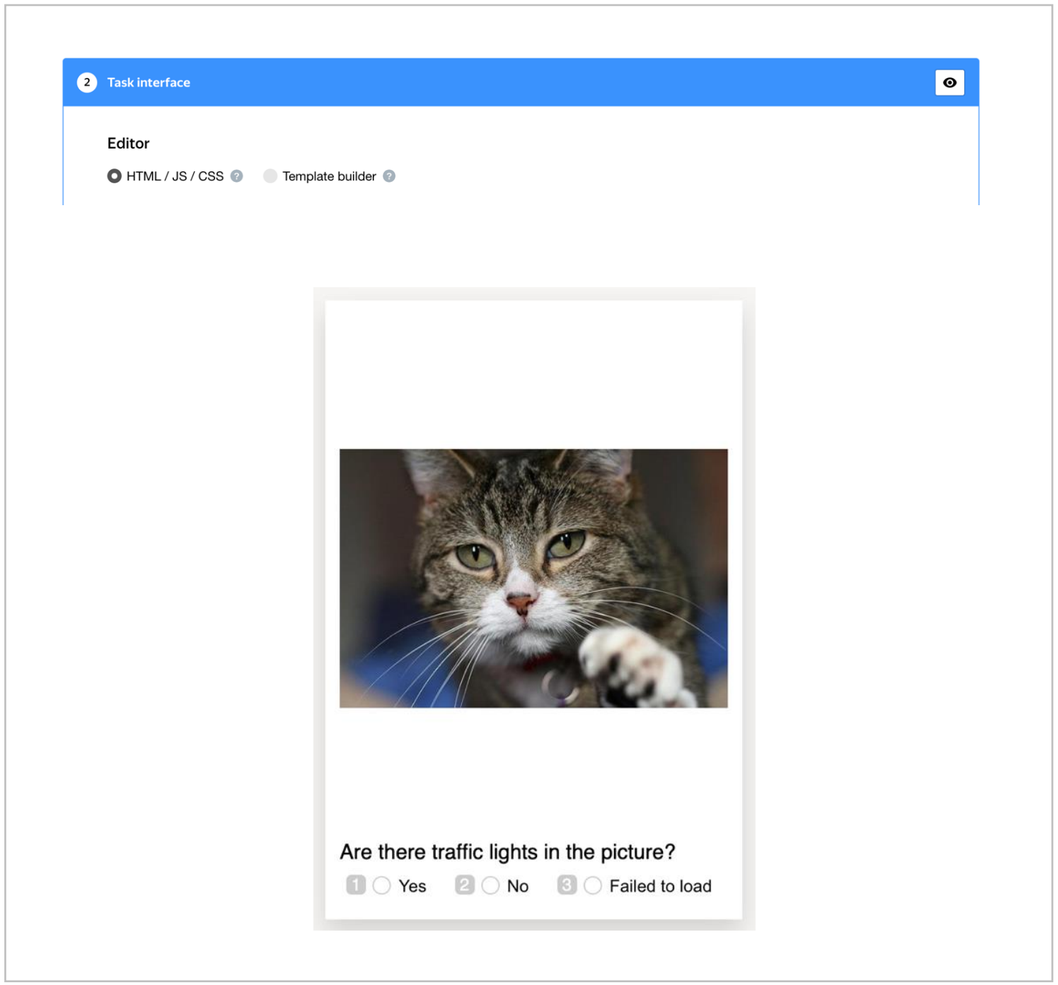
Project: configure in/out data types



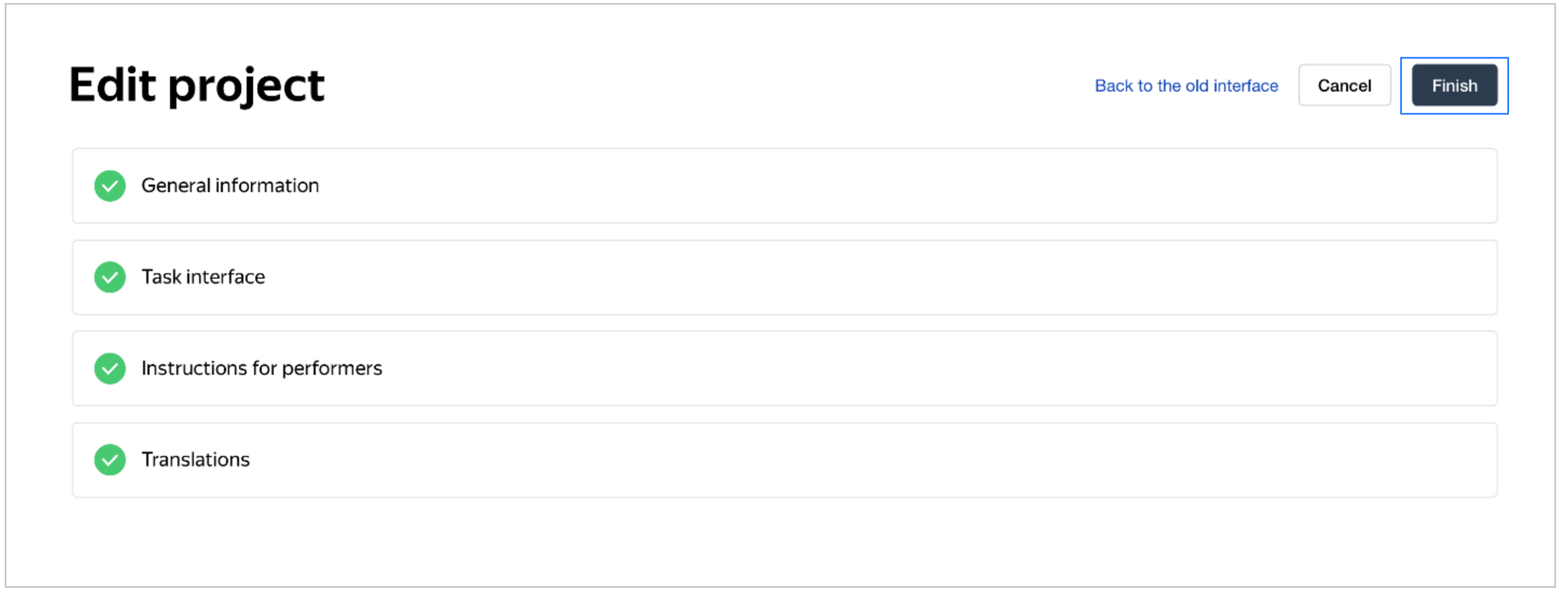
Project: configure interface



Project: use the task preview to see how it works

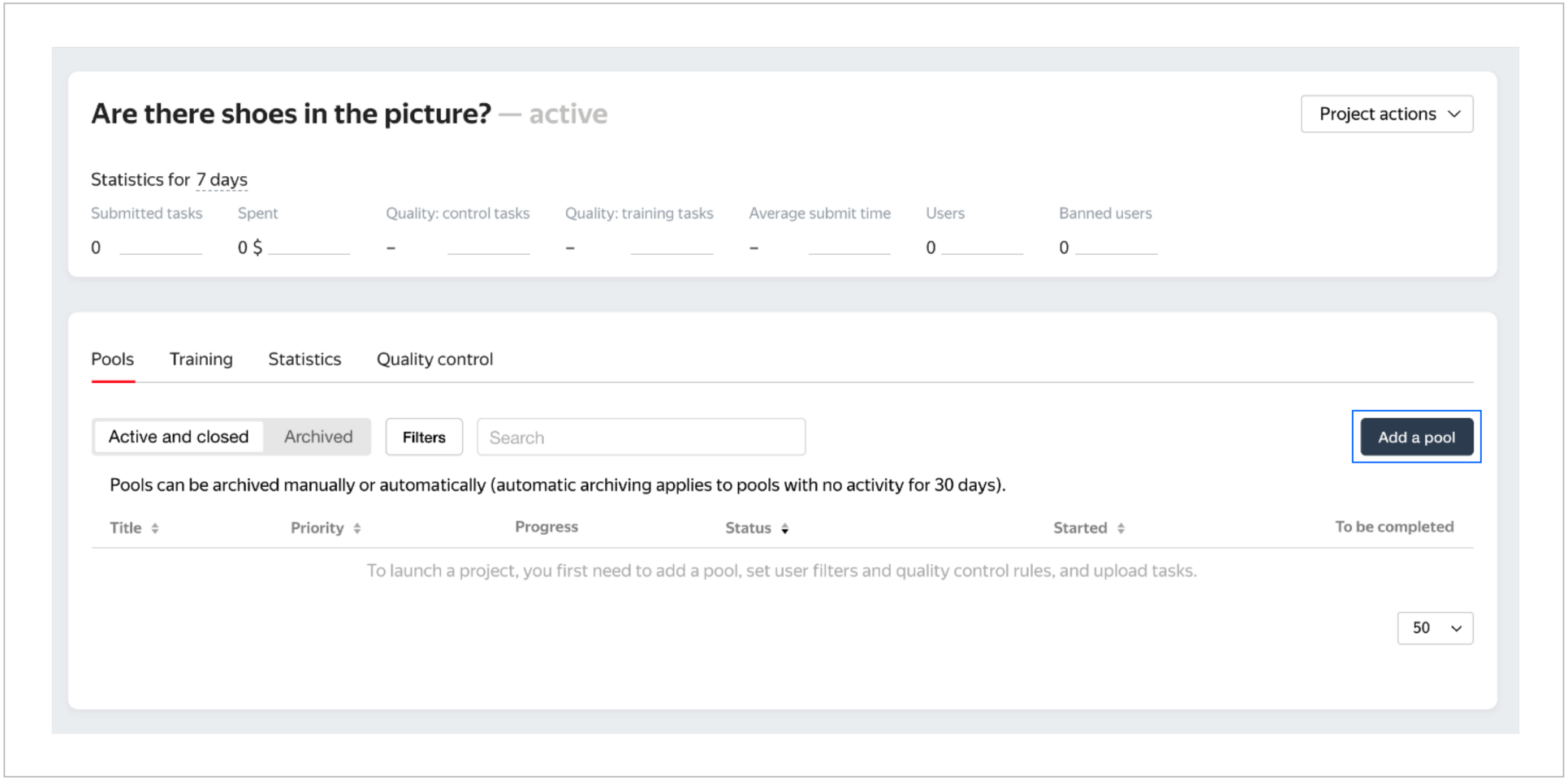


Project: saving

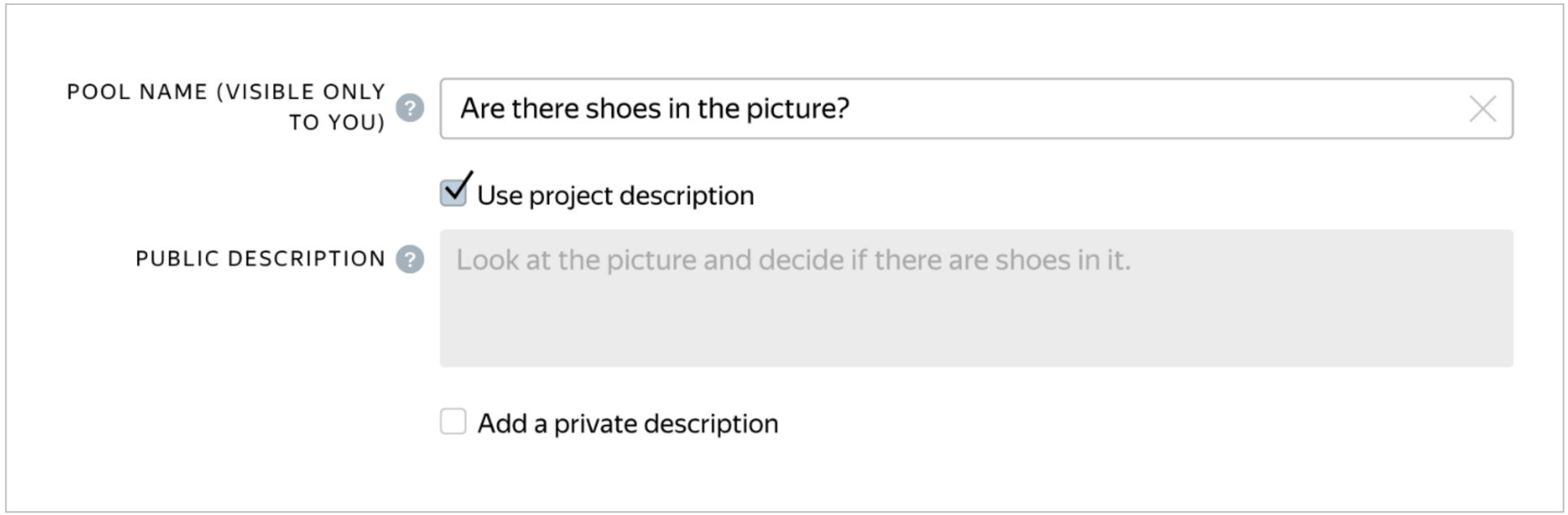


Pool:
creation & configuration

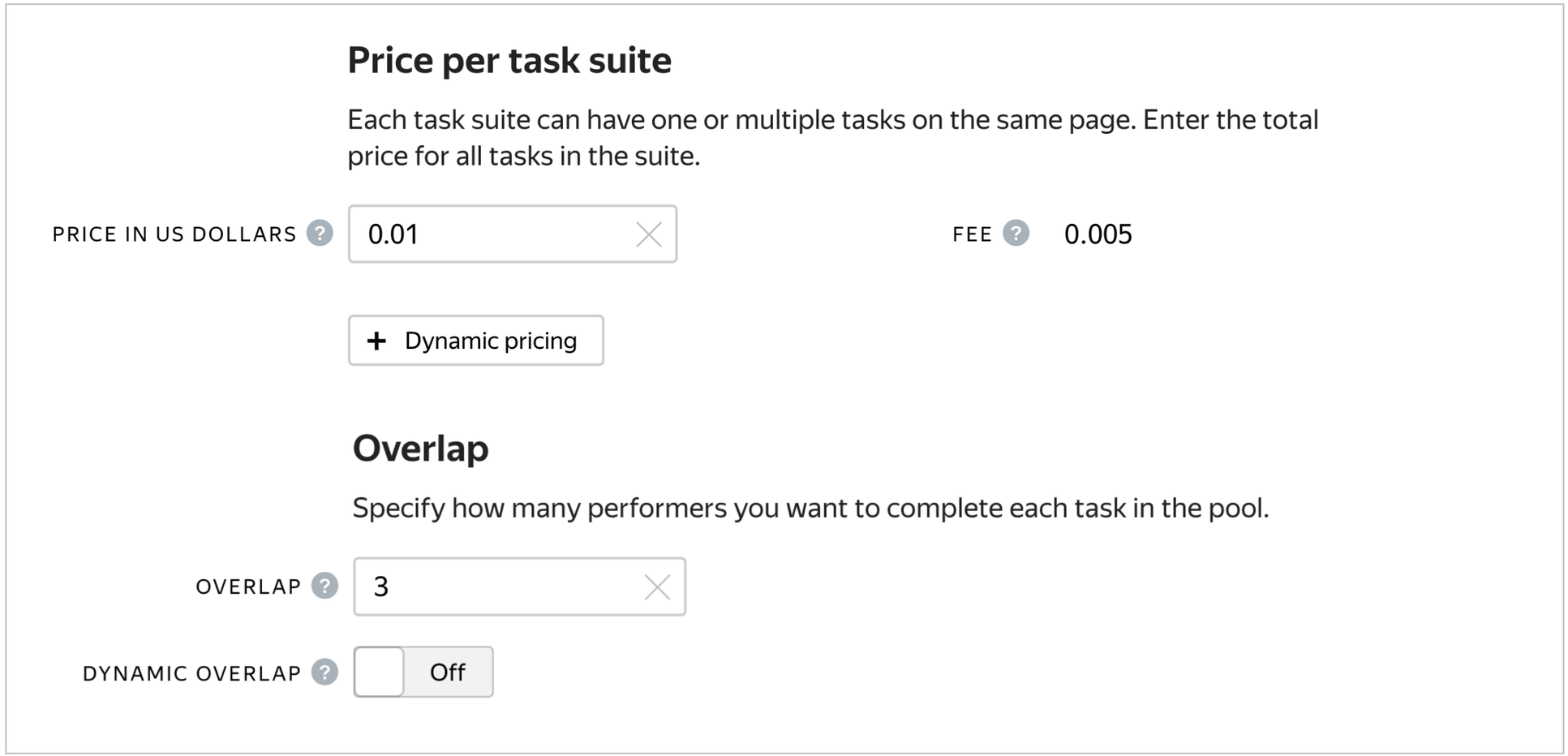
Pool: creation



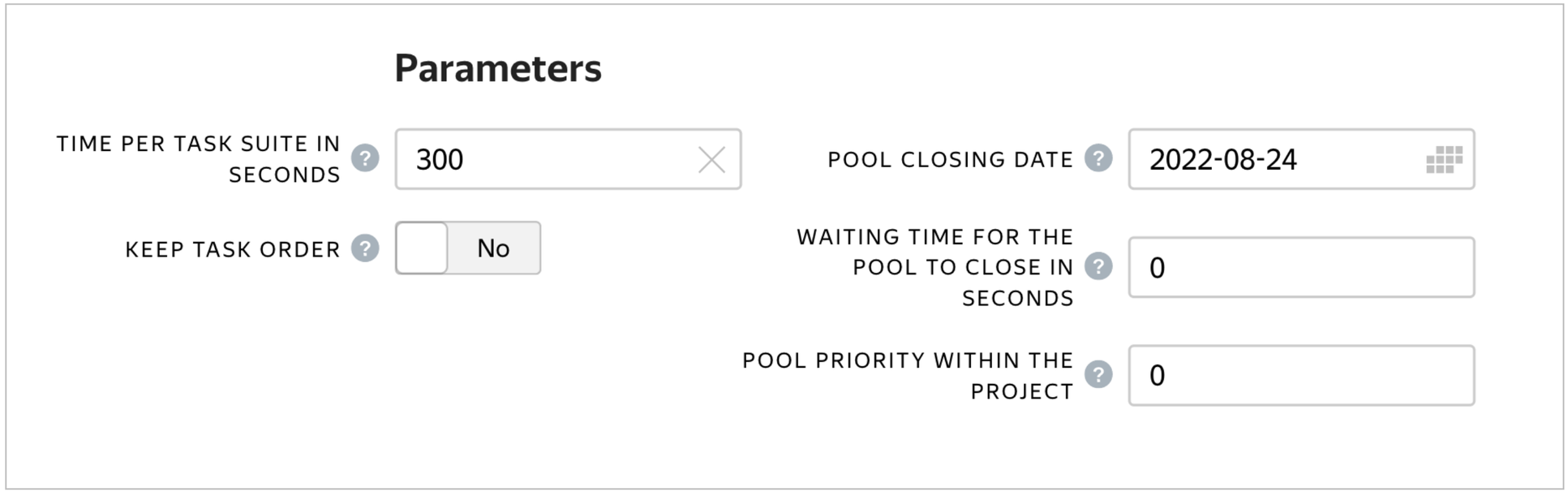
Pool: configure name and description



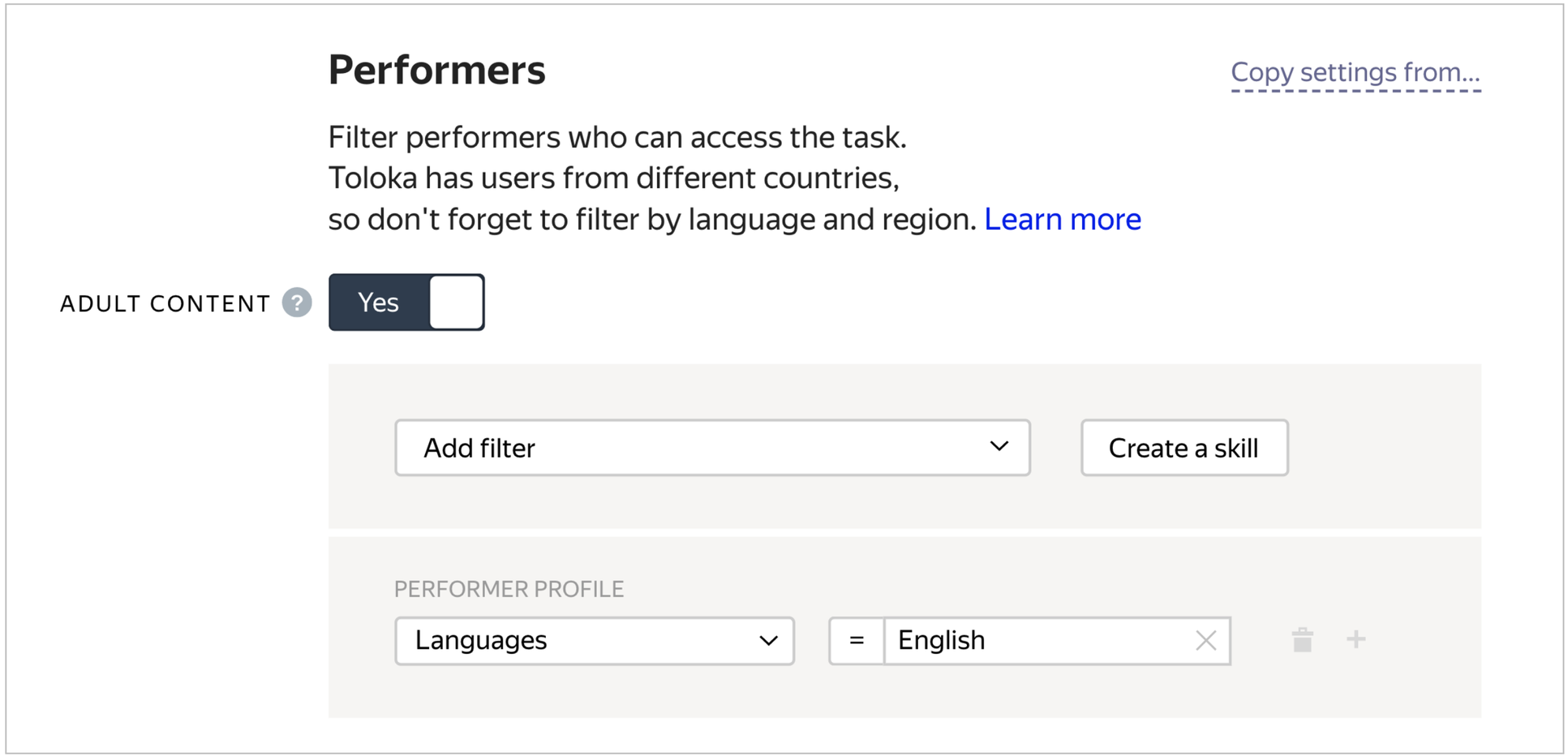
Pool: configure pricing and overlap



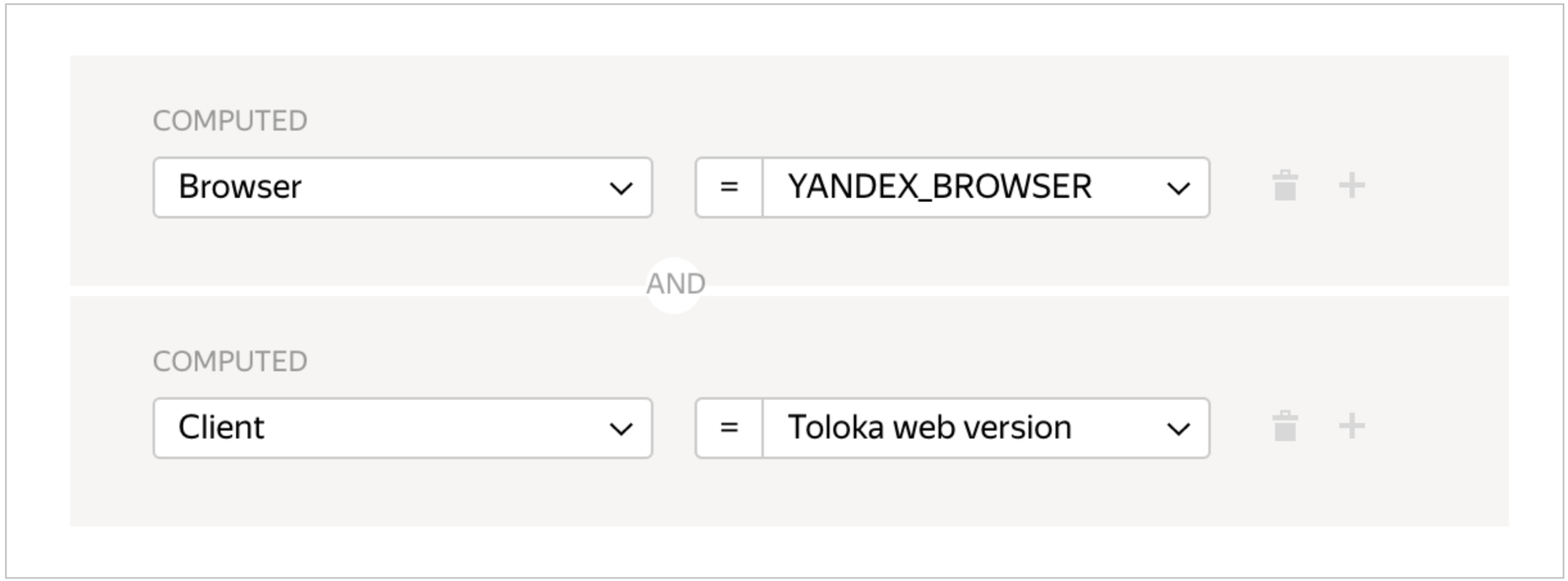
Pool: configure timing and post verification



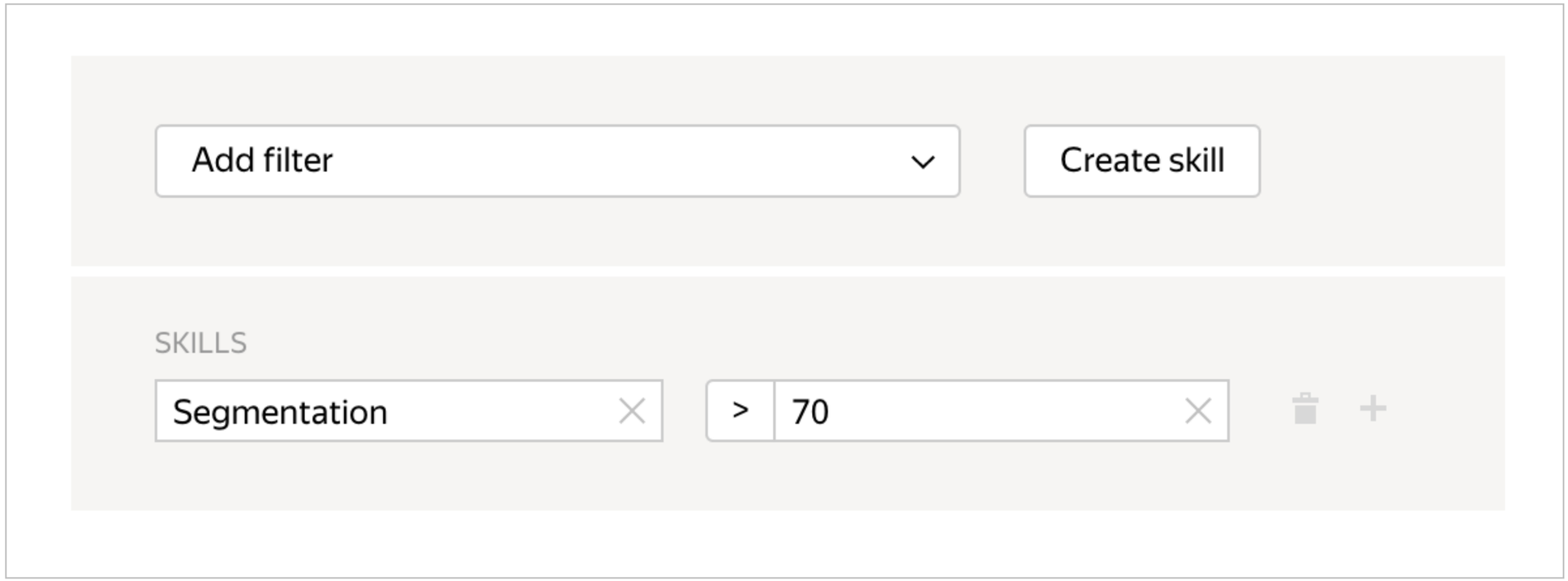
Pool: filter performers by their profiles



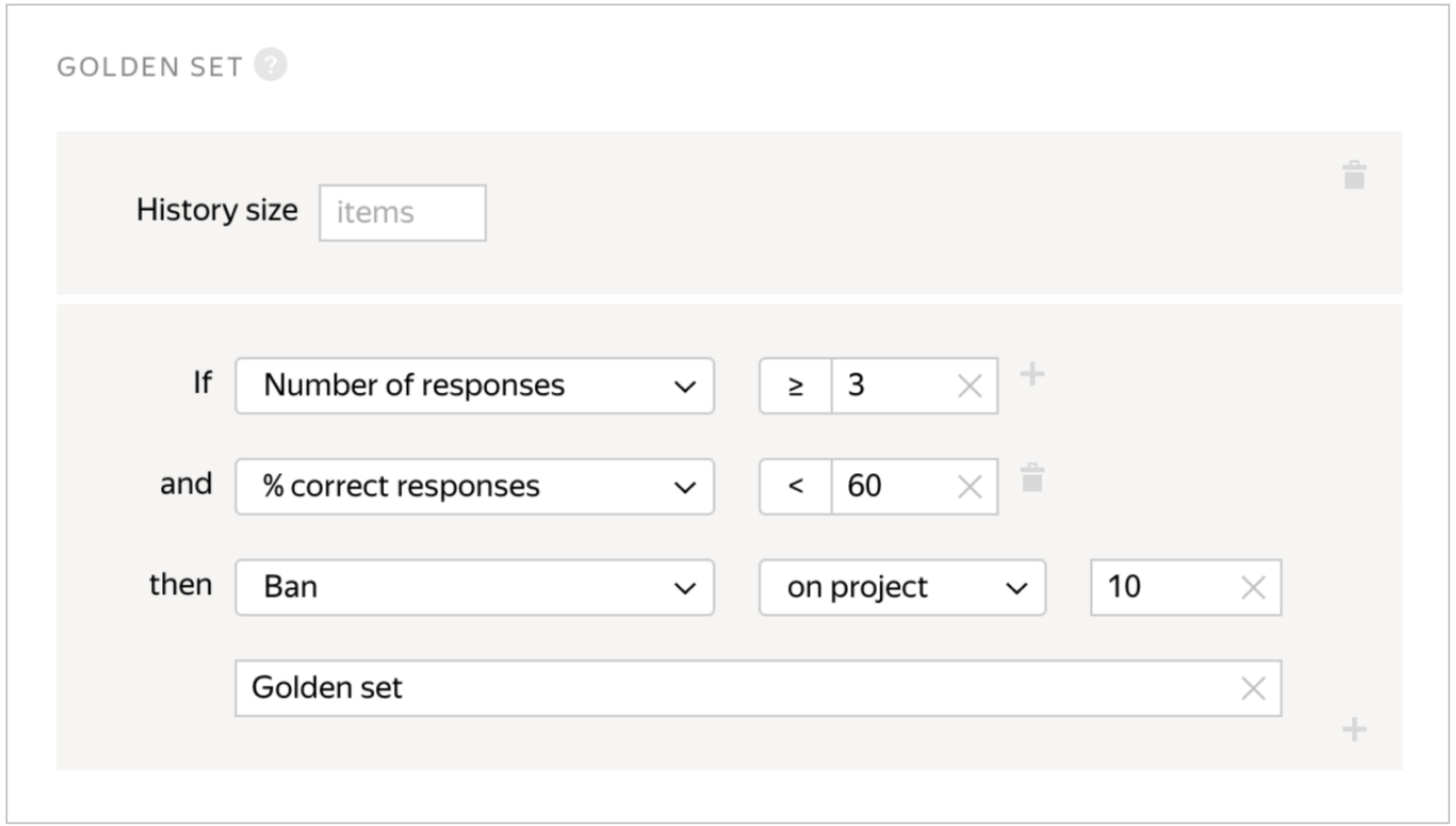
Pool: filter performers by computed properties



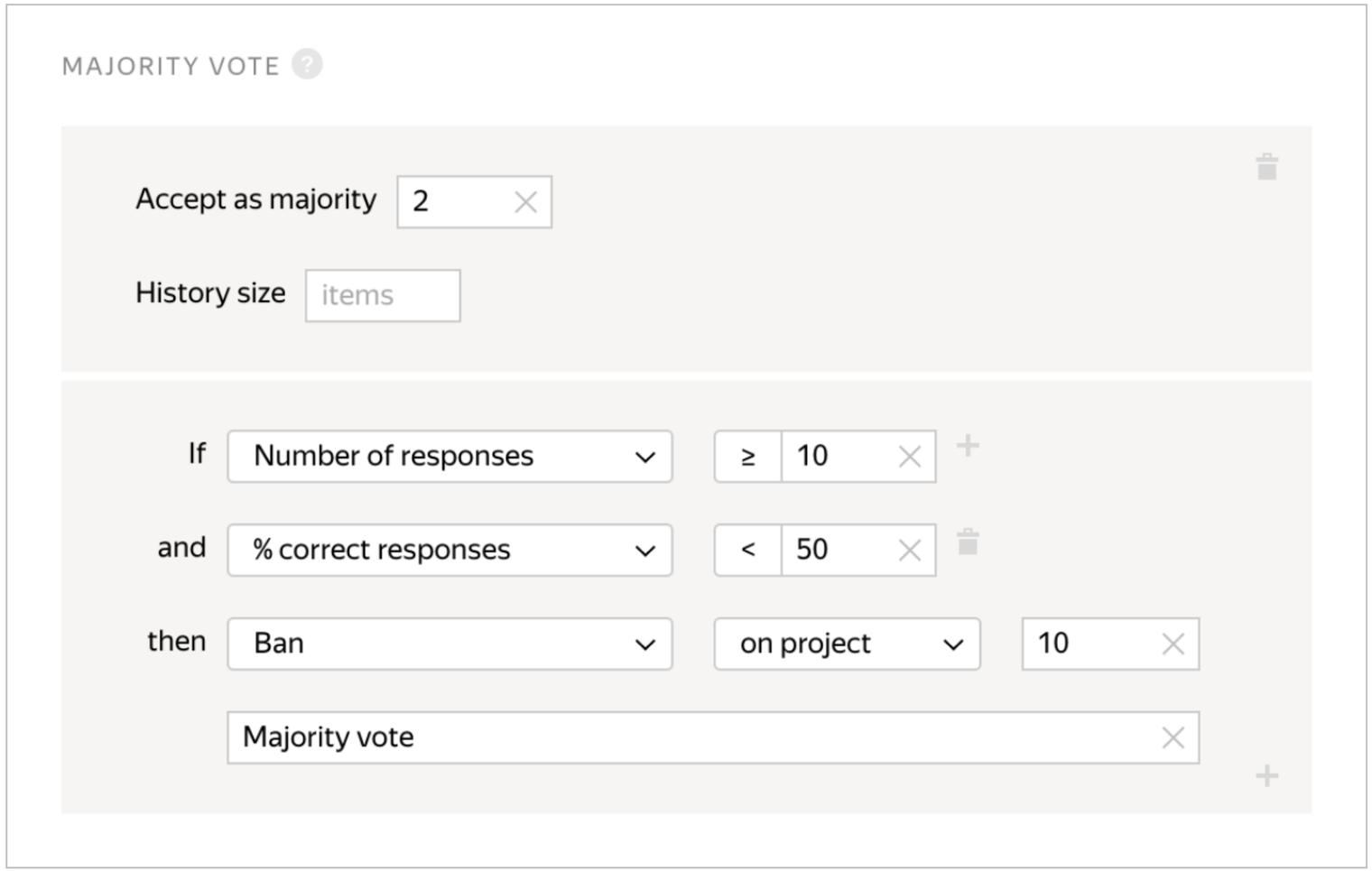
Pool: filter performers by custom skills



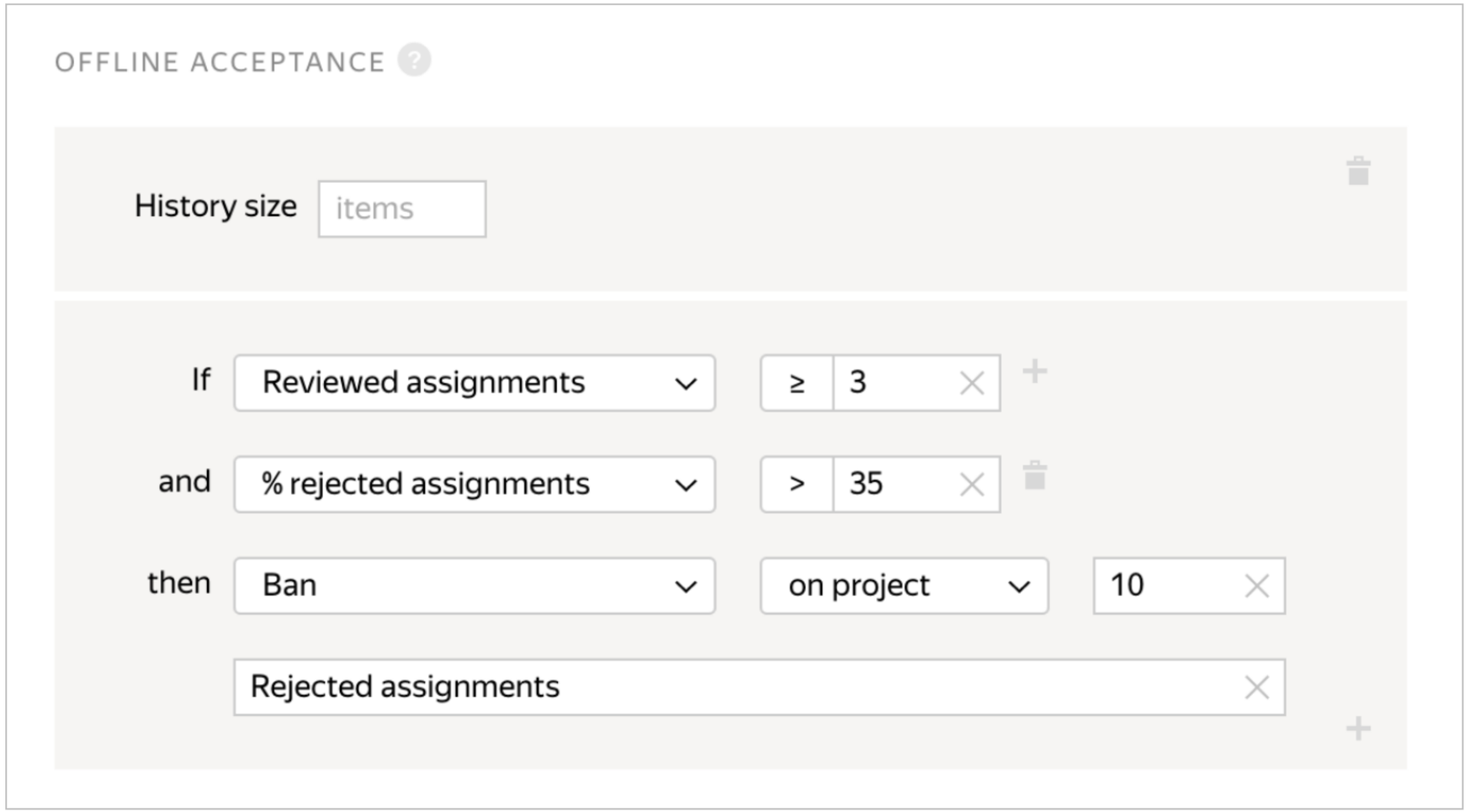
Pool: control quality via a rule on golden set



Pool: control quality via a rule on majority vote

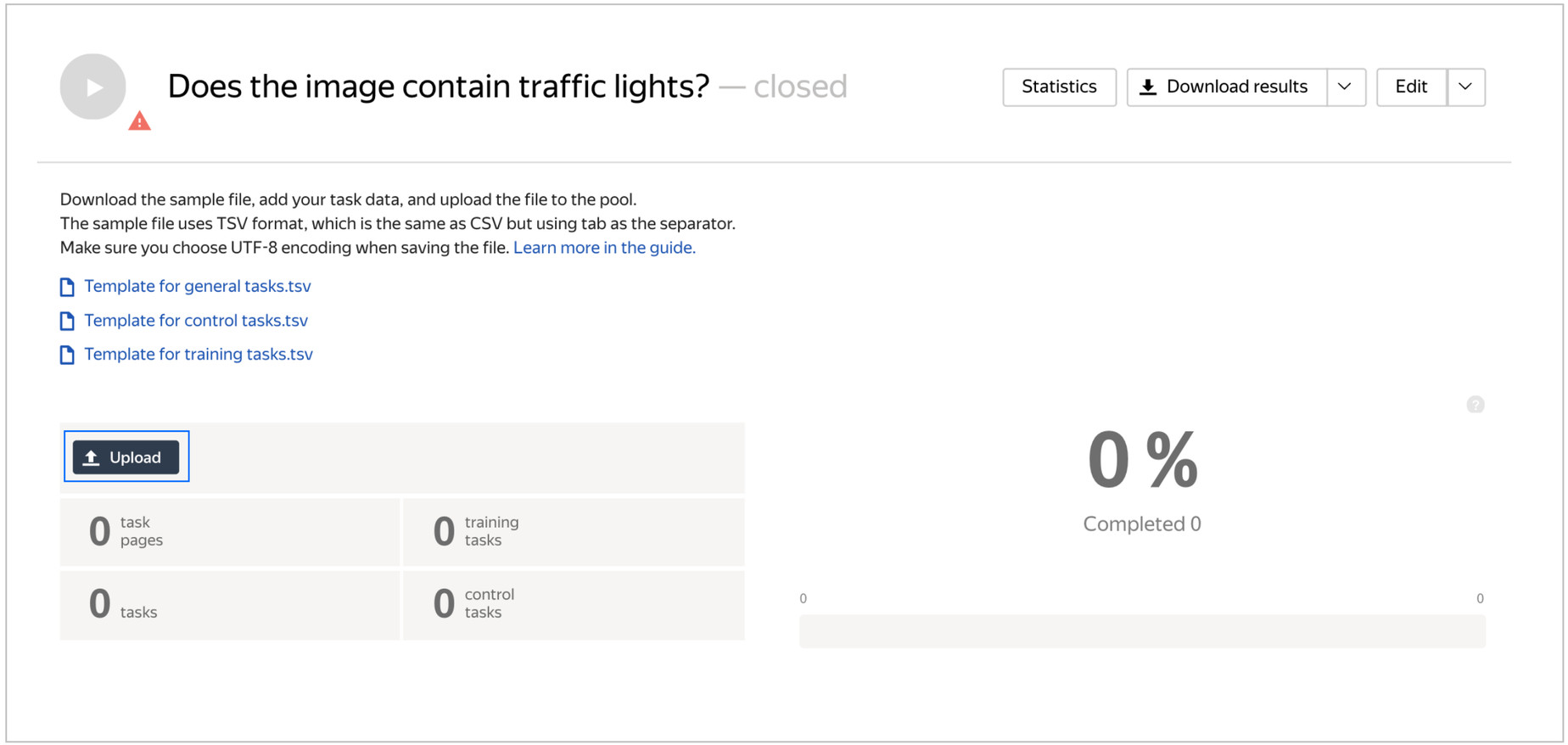


Pool: control quality via a rule on post verification

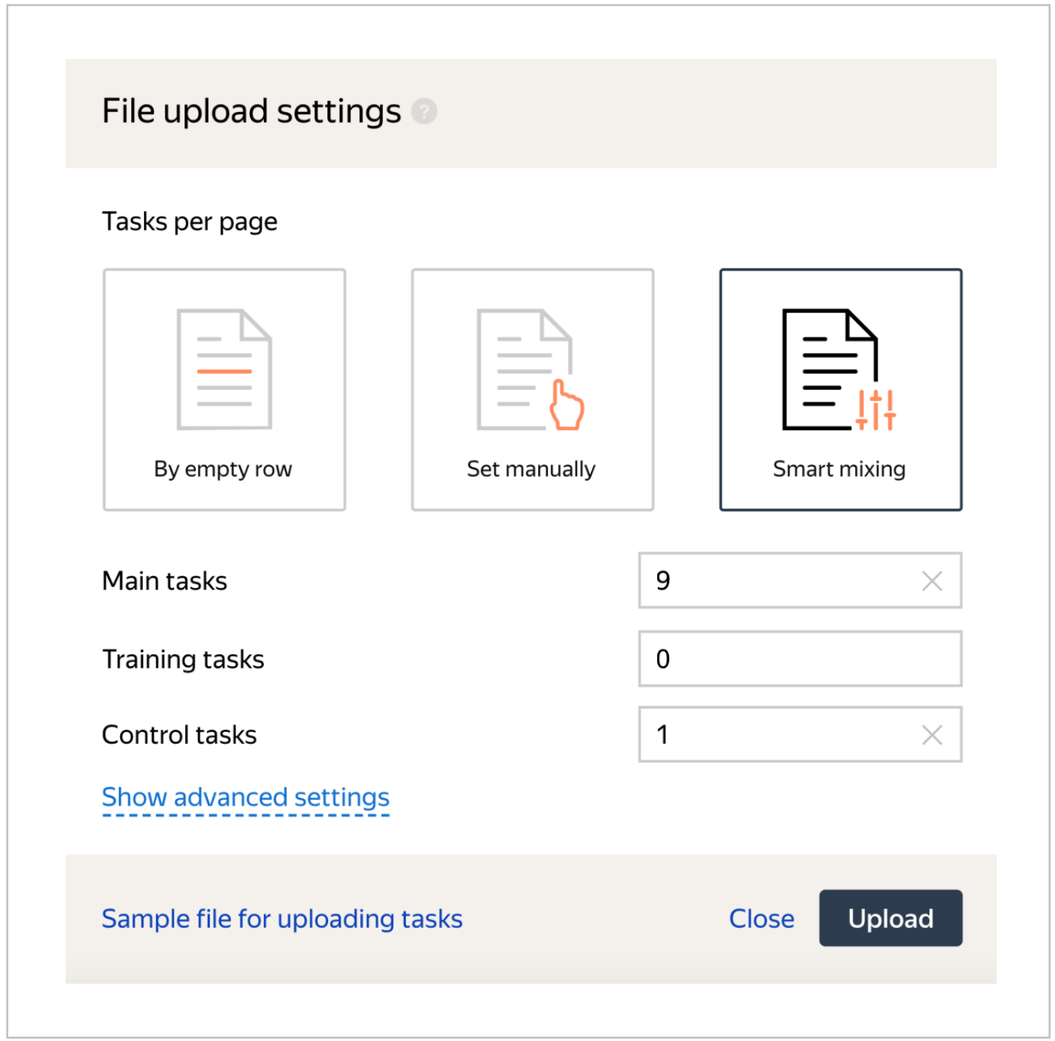


Task uploading & golden set creation

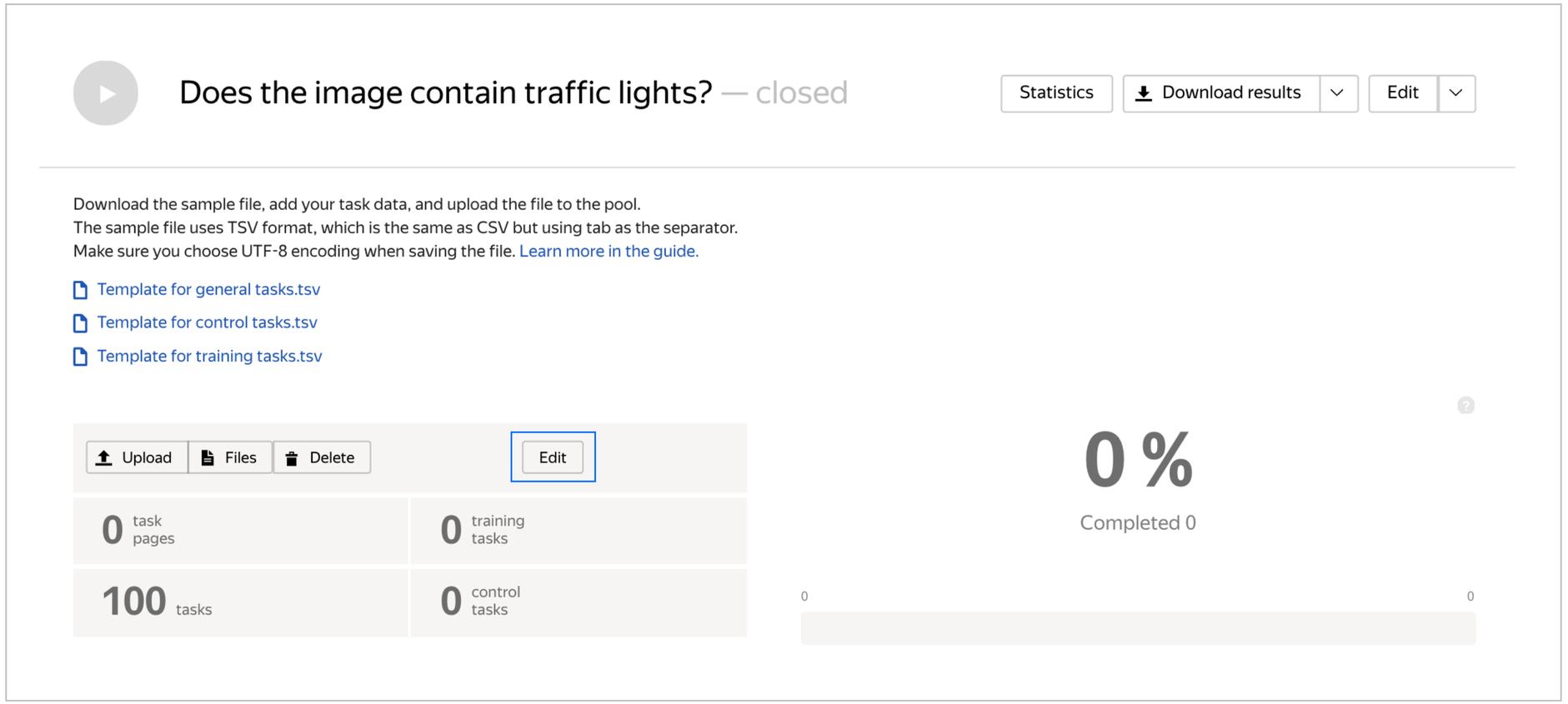
Task: uploading



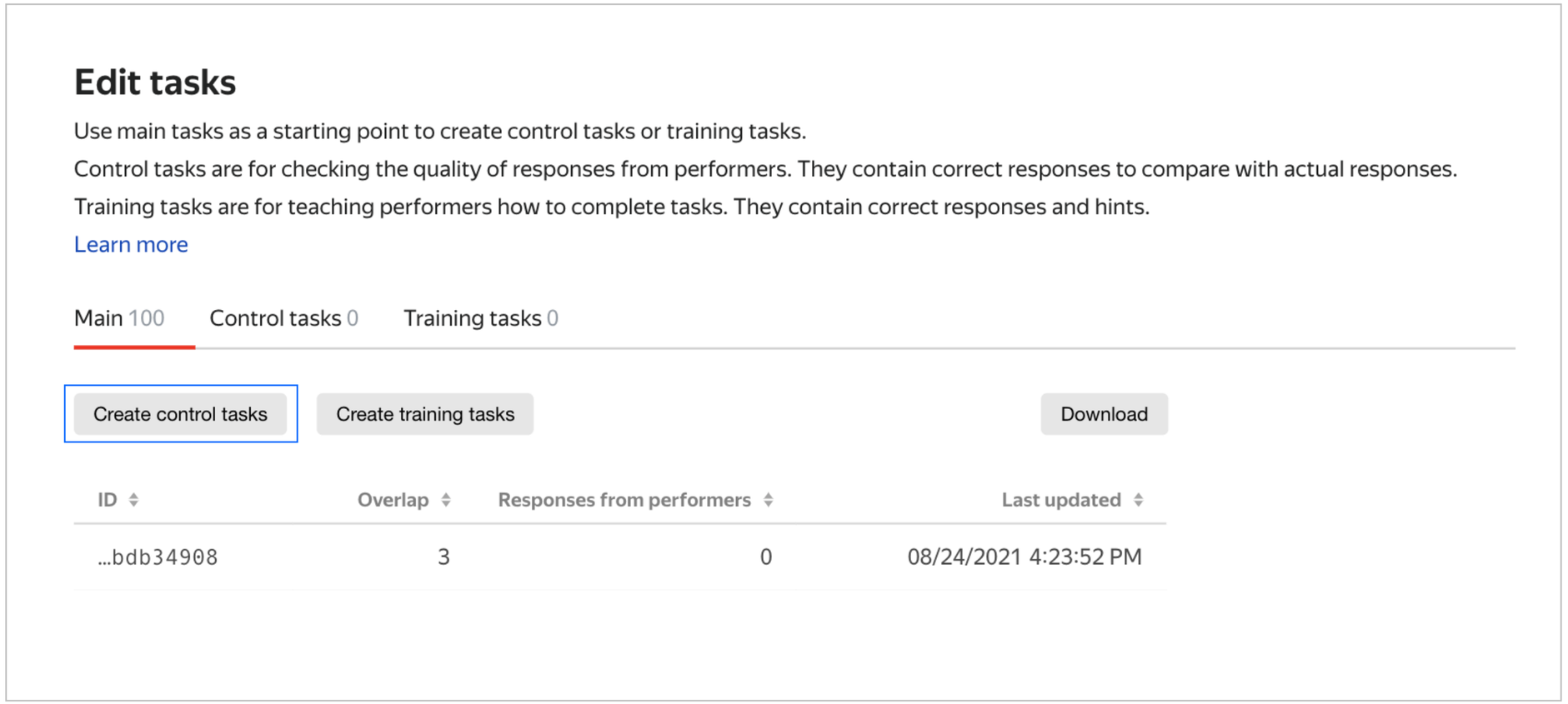
Task: uploading



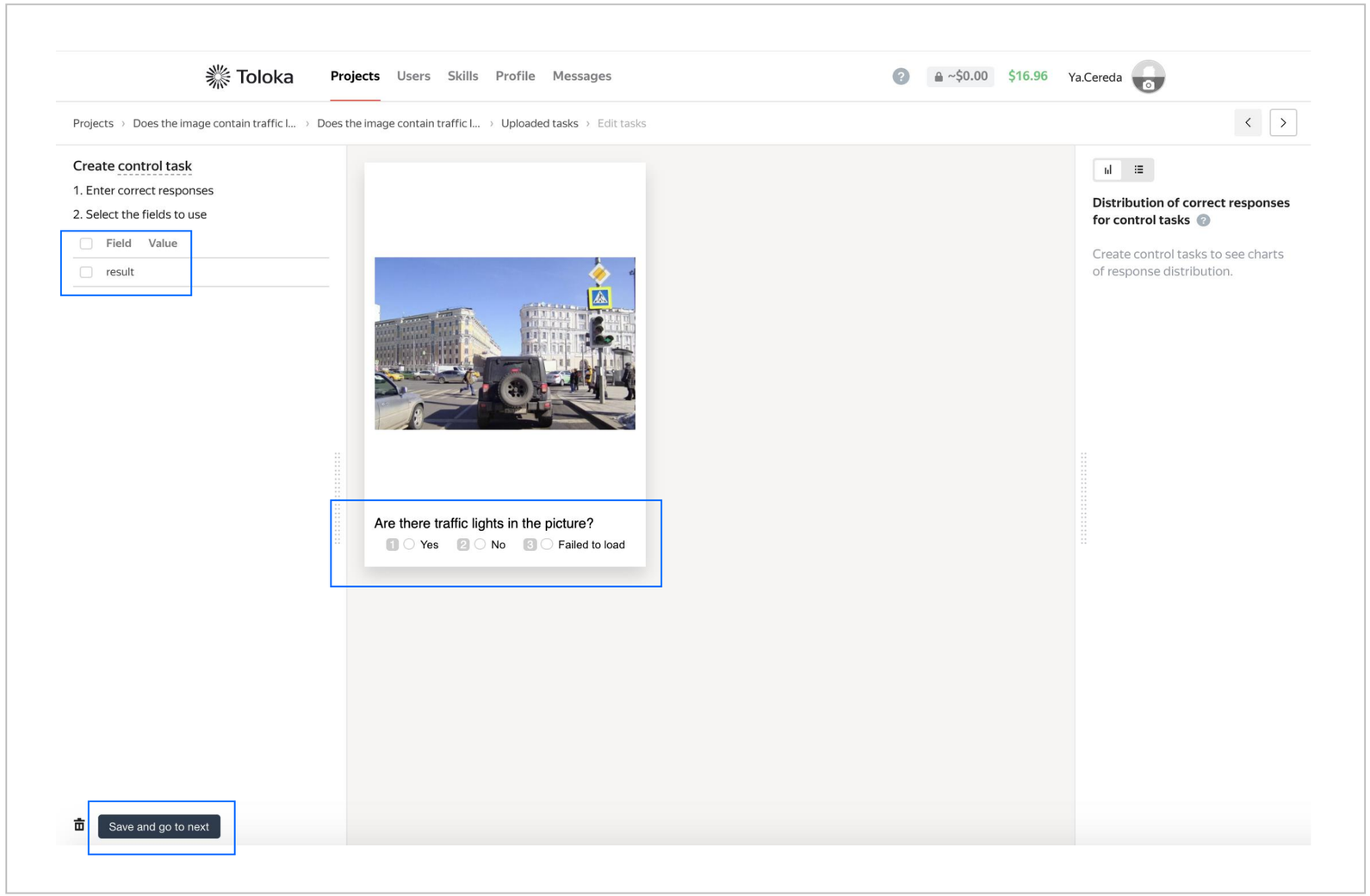
Task: edit for creation of control tasks



Task: control task creation

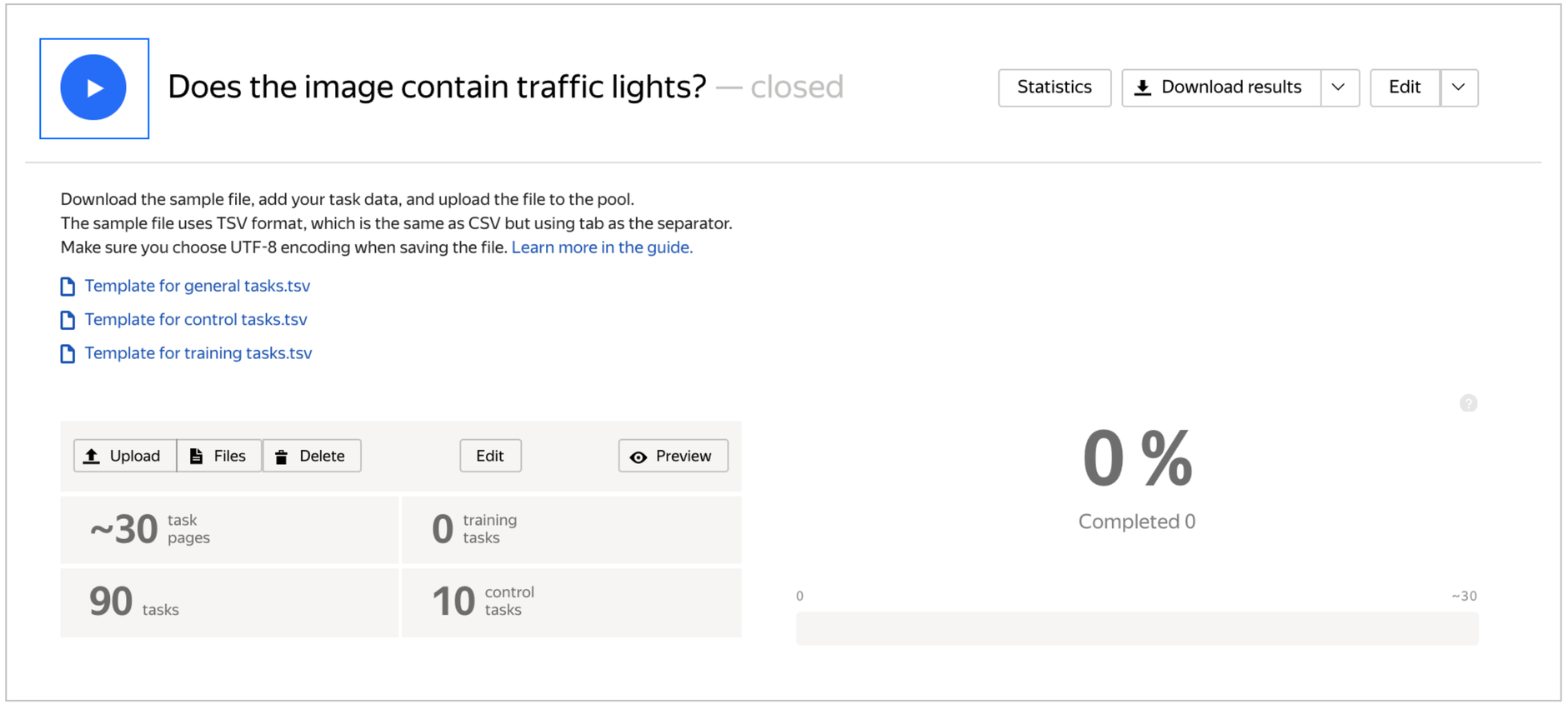


Task: create a control task by answer selection

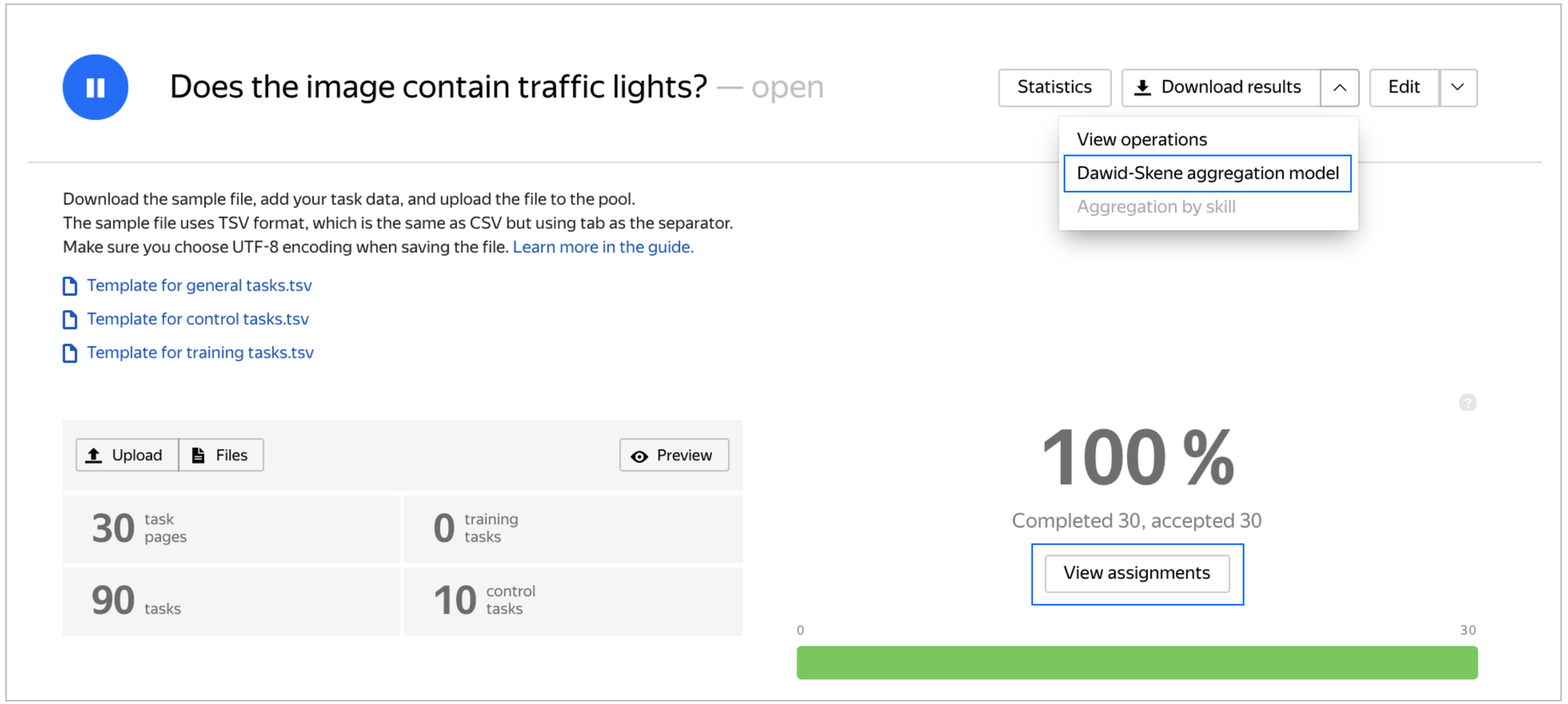


Run the pool & result downloading

Pool running



View and aggregate the results of your tasks



Monitor the statistics on how tasks are performed

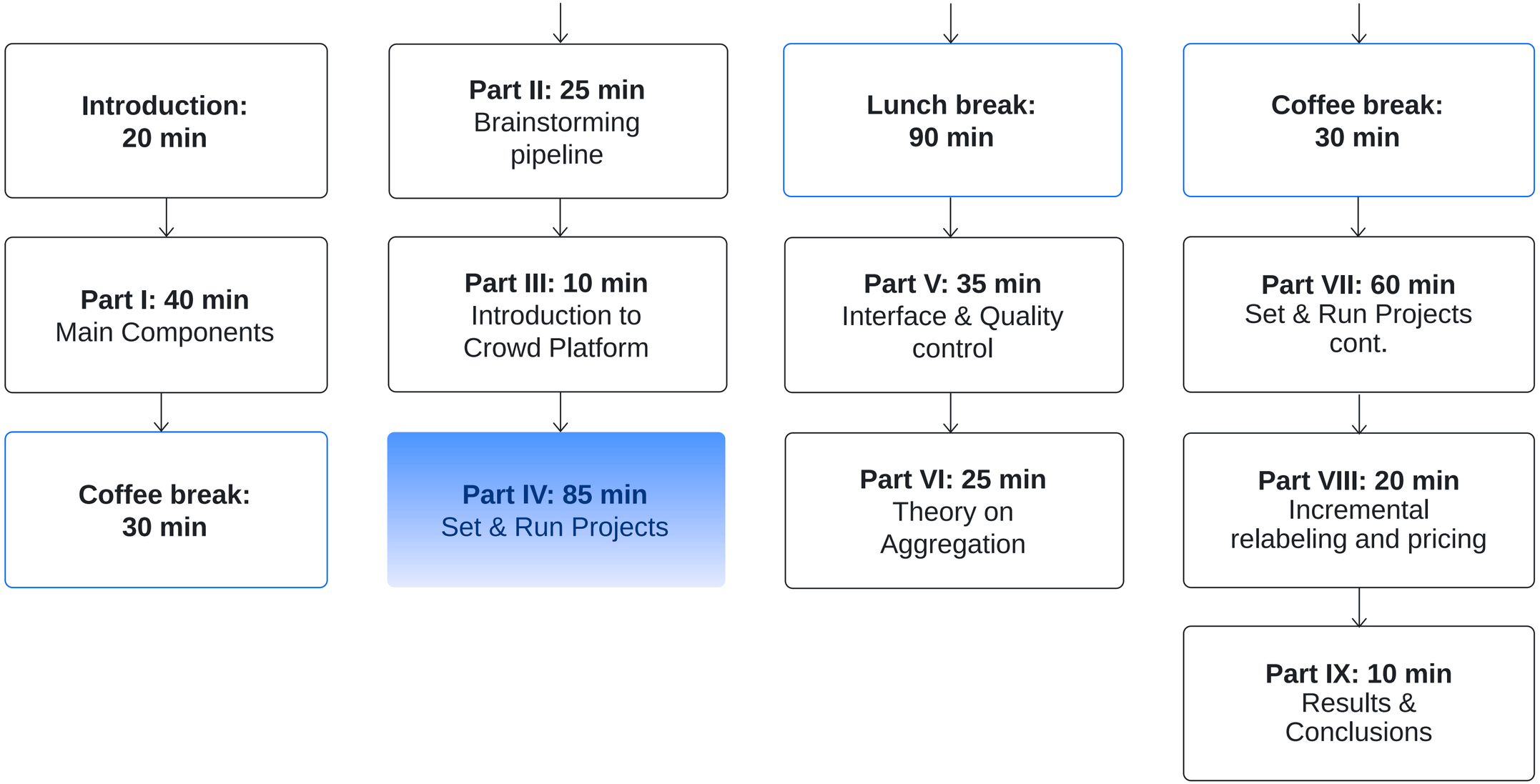
Part IV

Setting up and running label collection projects

Daria Baidakova,
Project Manager,

Toloka

Tutorial outline



What you need for the practice session

We are starting the practice session

We give you a card with information and links to:

- A step-by-step instruction to configure and run our crowd projects
- A dataset with photos that should be labeled
- Login+Password to sign in Toloka as a requester

We also provide several copies of a printed version of the instruction

Did everybody receive this card?

Requester account that you received

You have Login+Password to sign in Toloka as a requester

The same account is given for several participants (a group)

- So, you can divide work on the project configuration within this group
- Or, each member of a group may work individually and create the whole pipeline by her/himself

Sign in Toloka as a requester

1. Go to https://toloka.ai
2. Click on “Sign in” in the top-right corner
3. Use received Login+Password to sign in

Requester account that you received

You have Login+Password to sign in Toloka as a requester

The account of this requester has money

- So, you will run your tasks on real crowd performers!

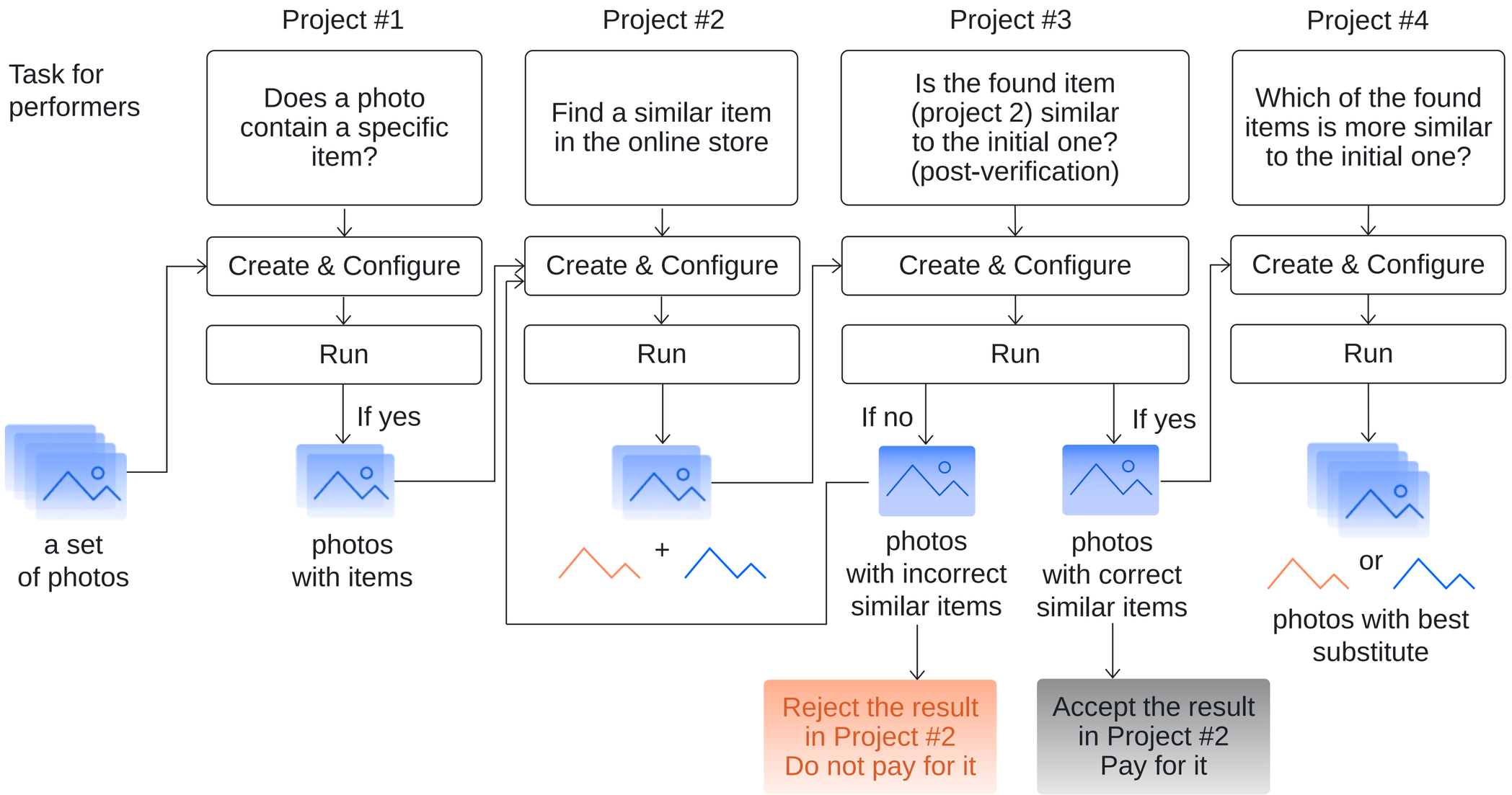
Practice: creating a real crowdsourcing pipeline

Now we will create a real simplified crowdsourcing pipeline

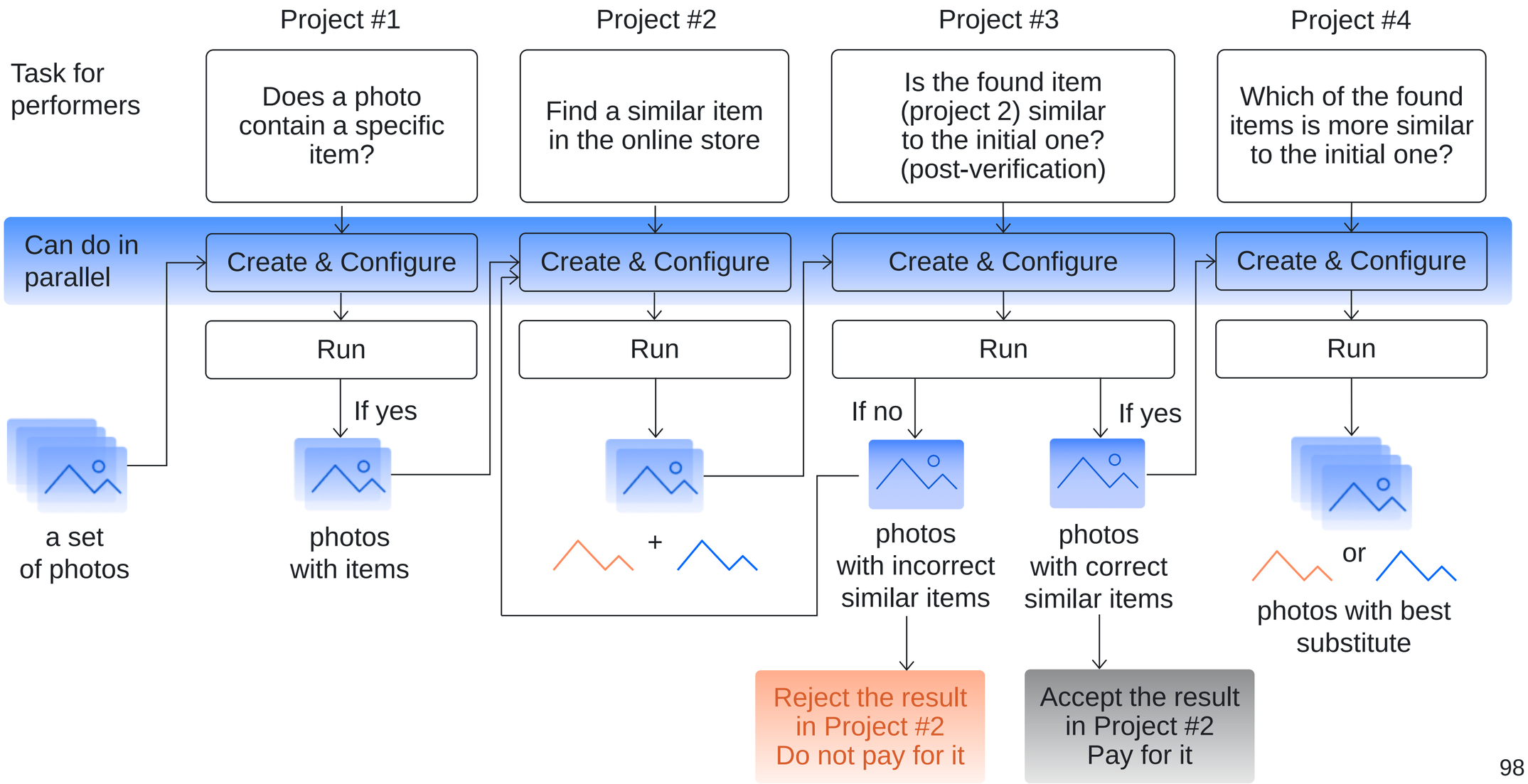
To simplify the task, we ask you to:

- Finding a substitute for **one type** of item
- Choose any item you want to find the best substitute for. For example, Shoes

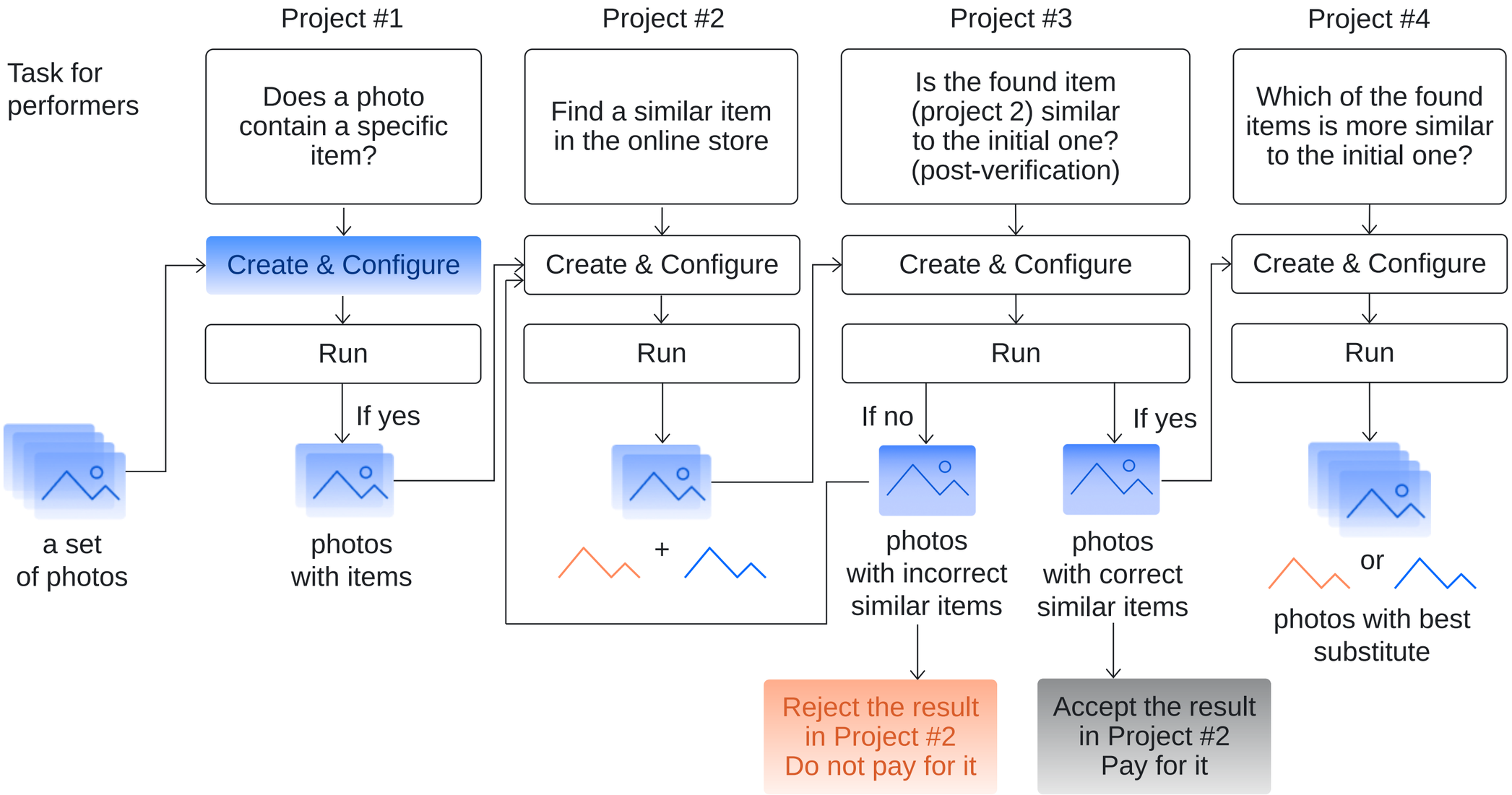
Reminder: we implement and run our pipeline



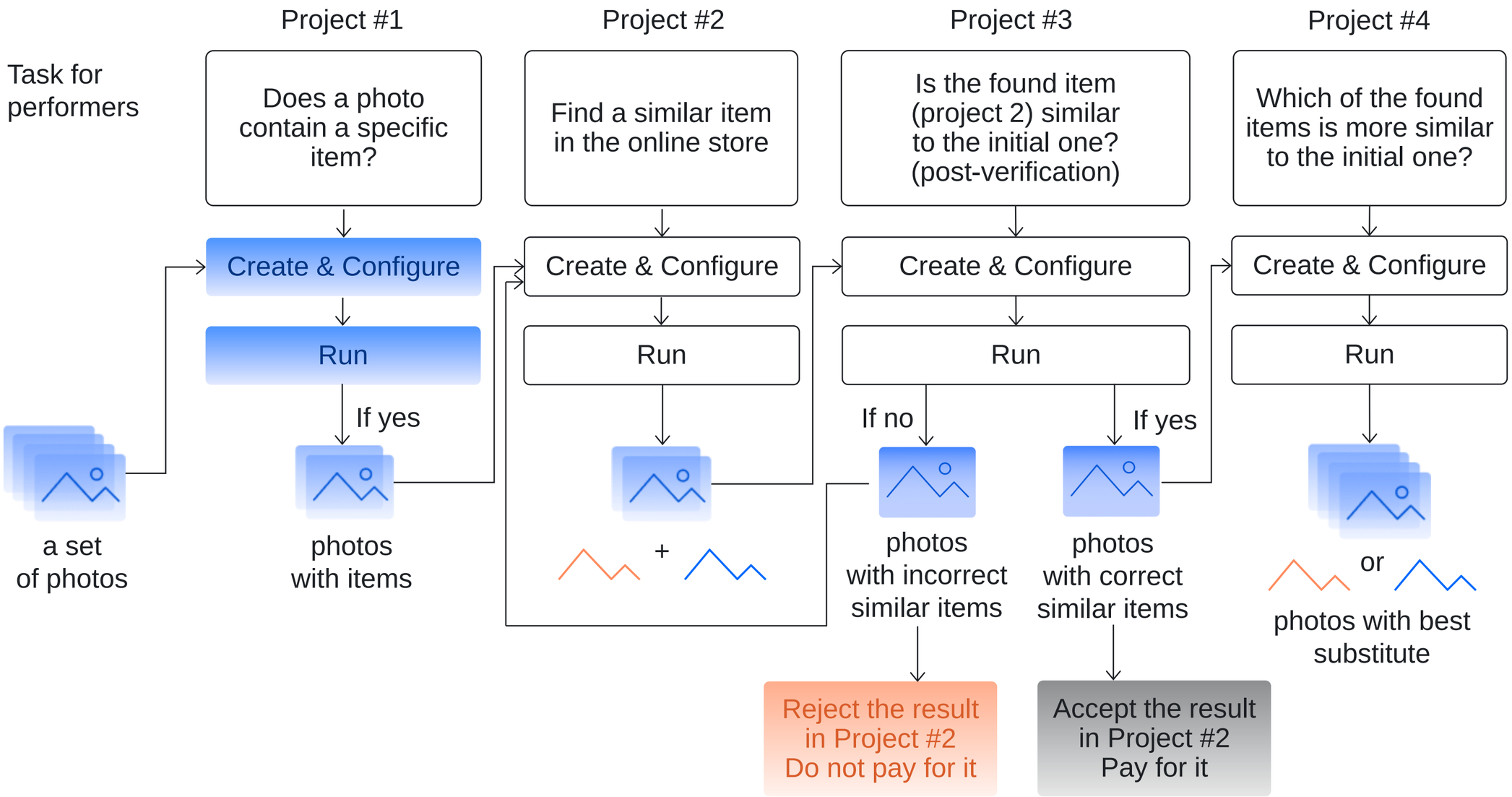
You can divide work within a participant group



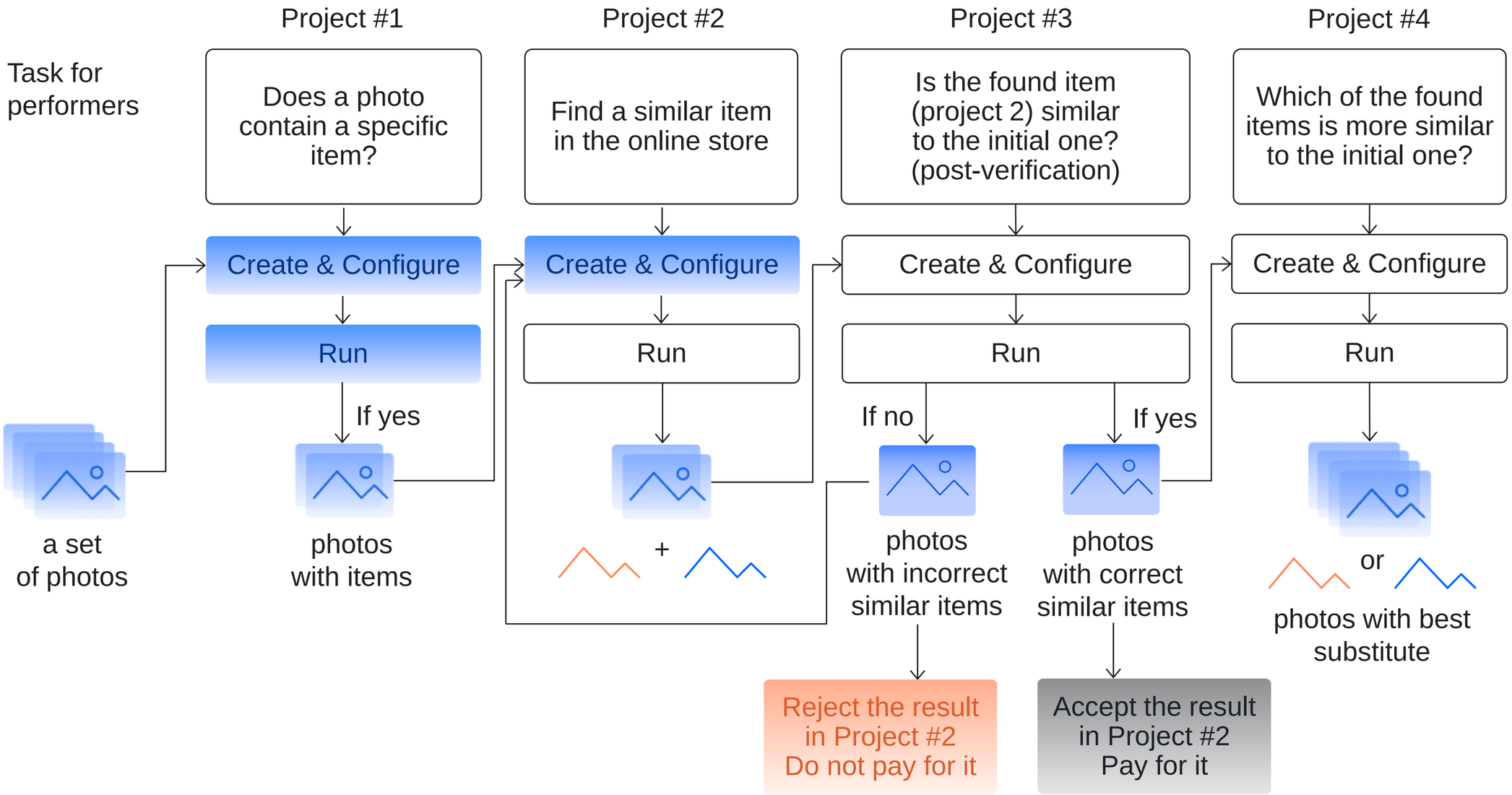
Step #1


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

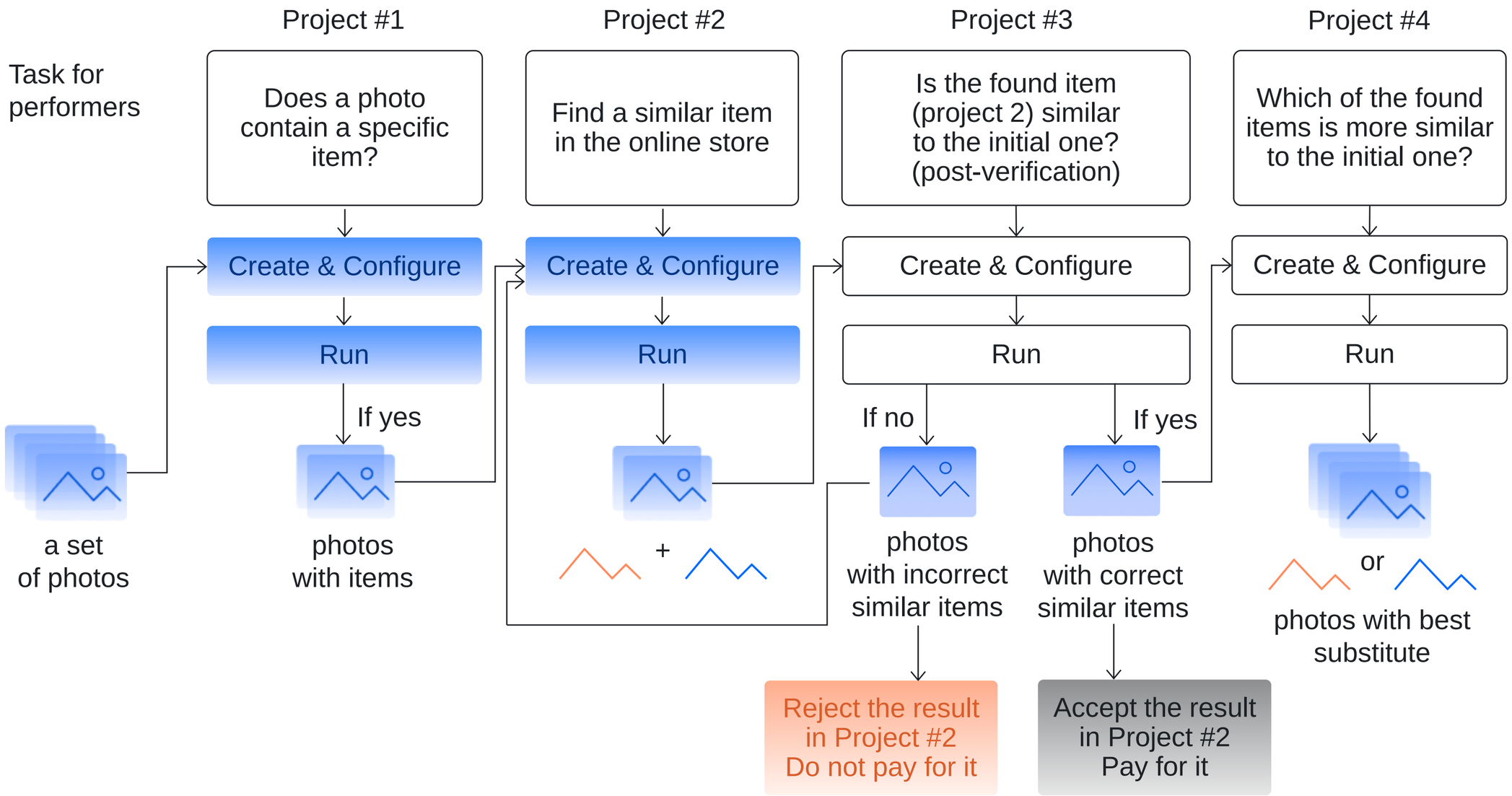
Step #2


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

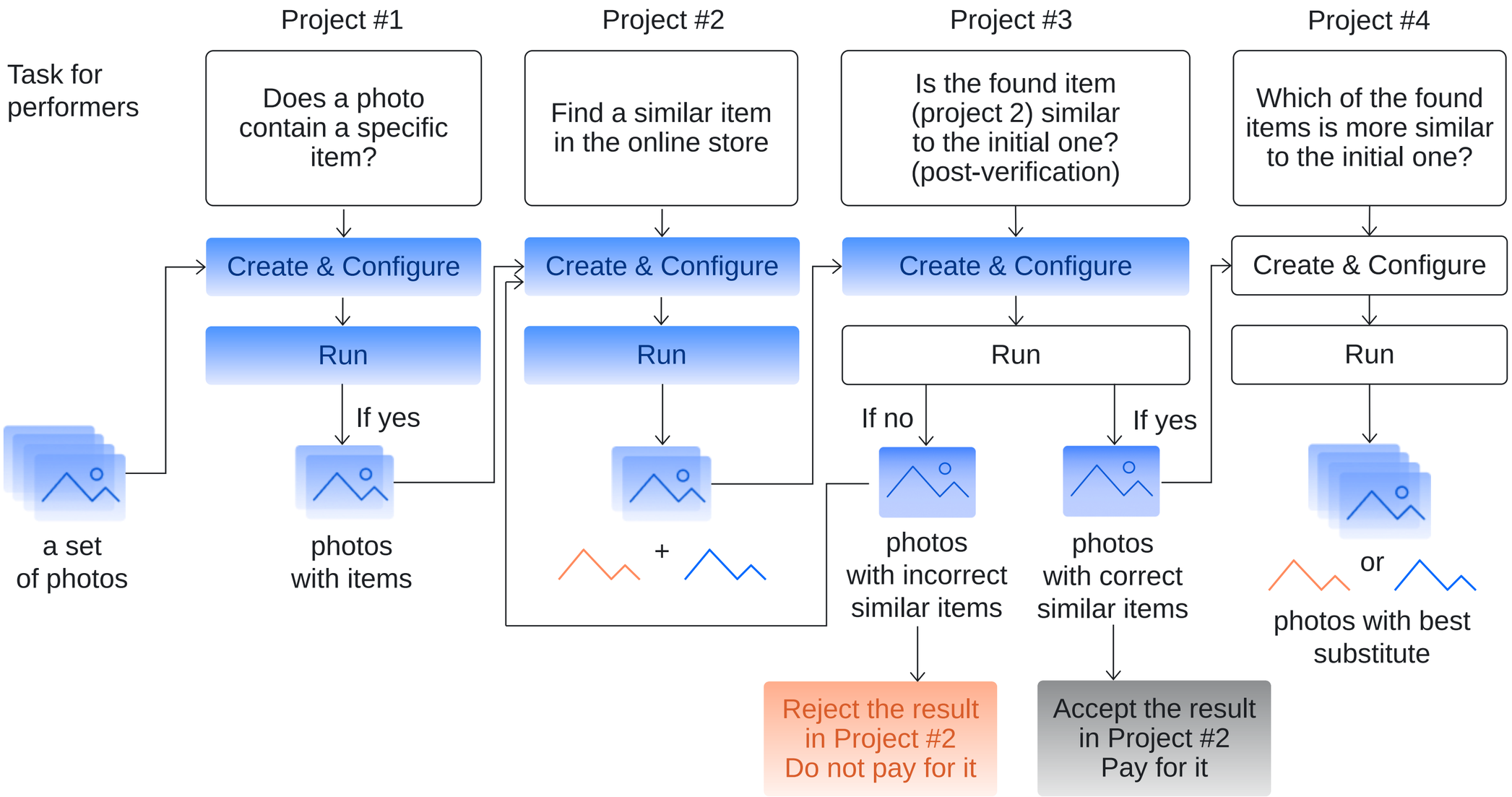
Step #3


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

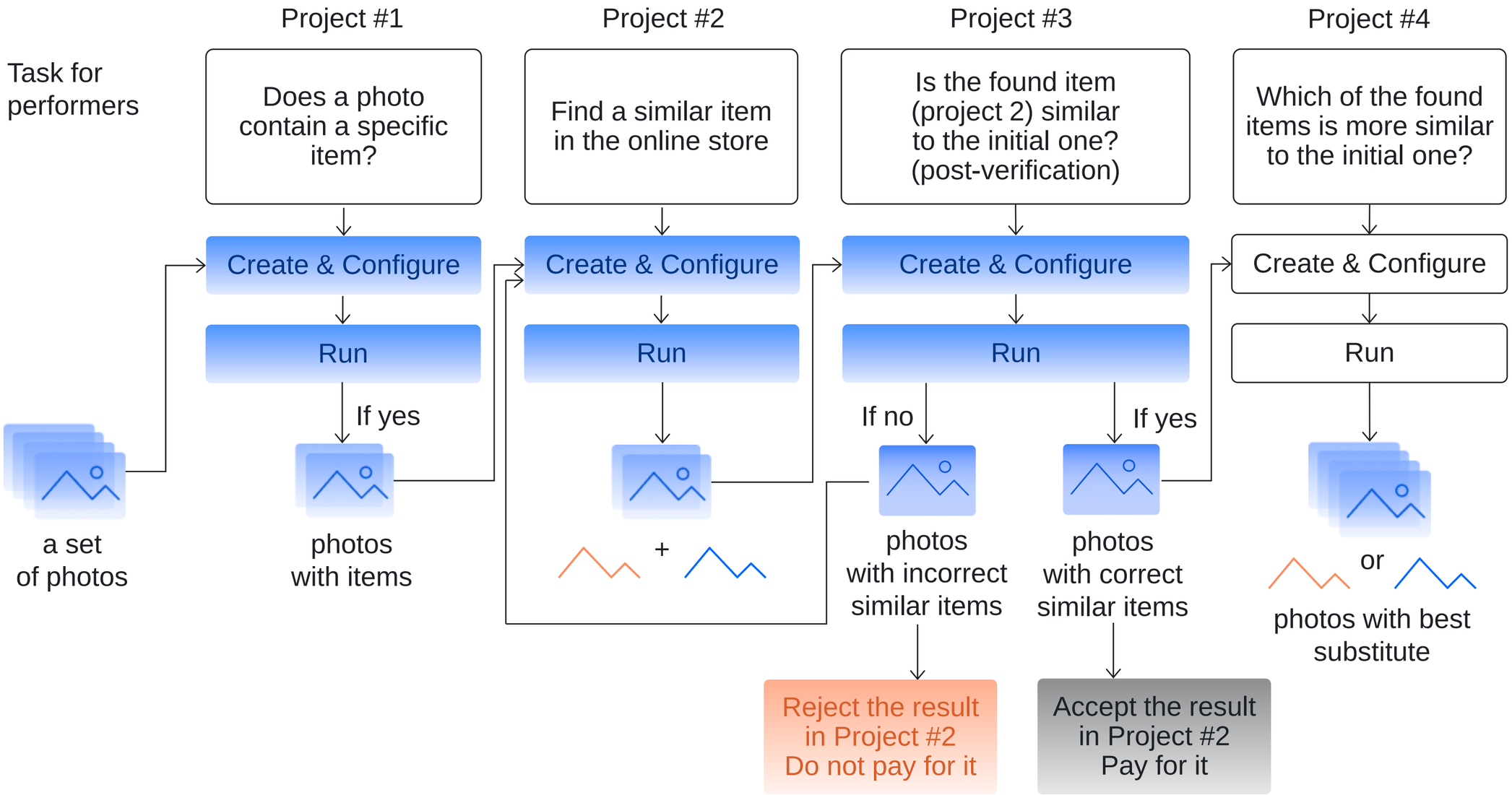
Step #4


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

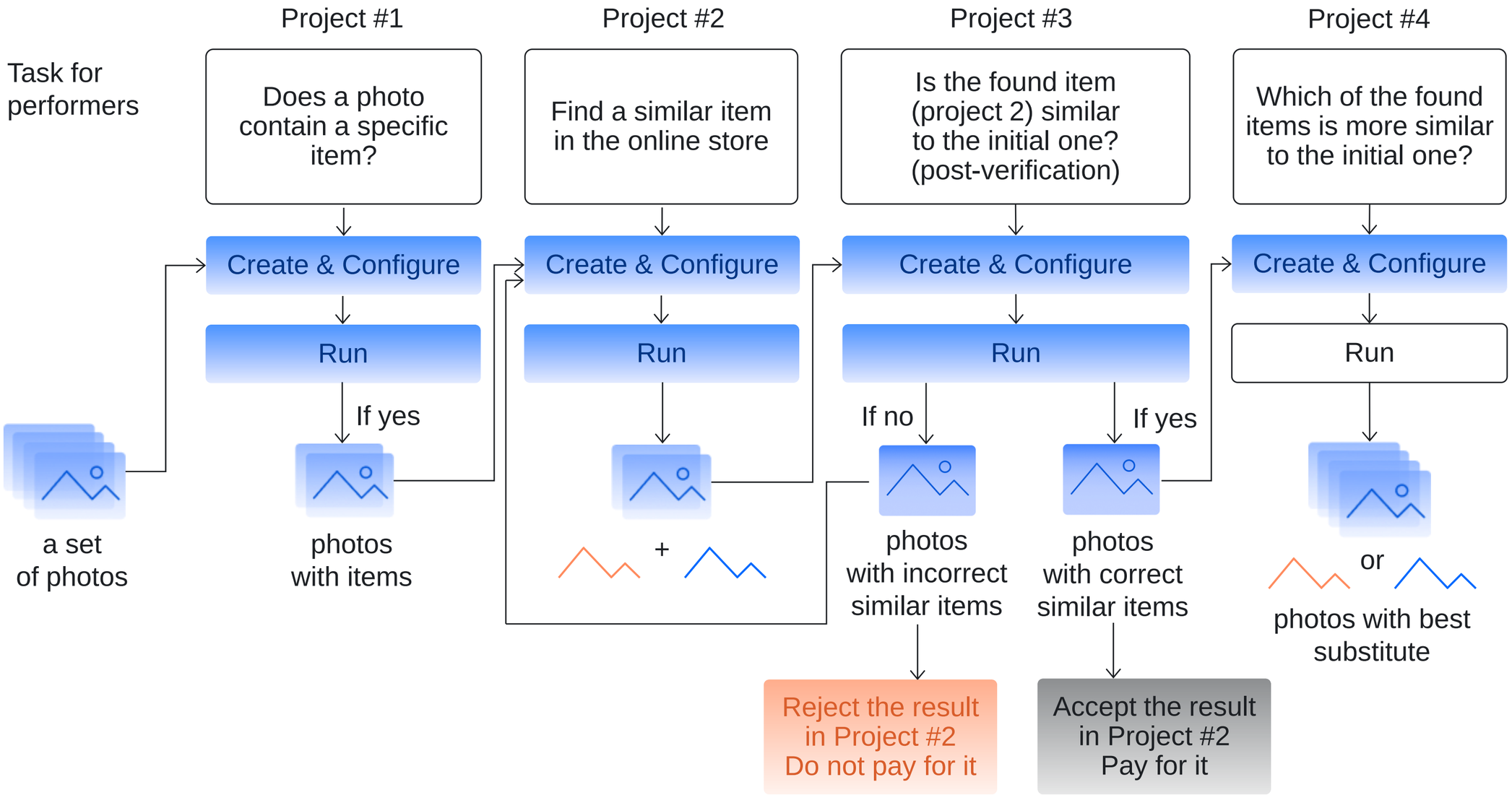
Step #5


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

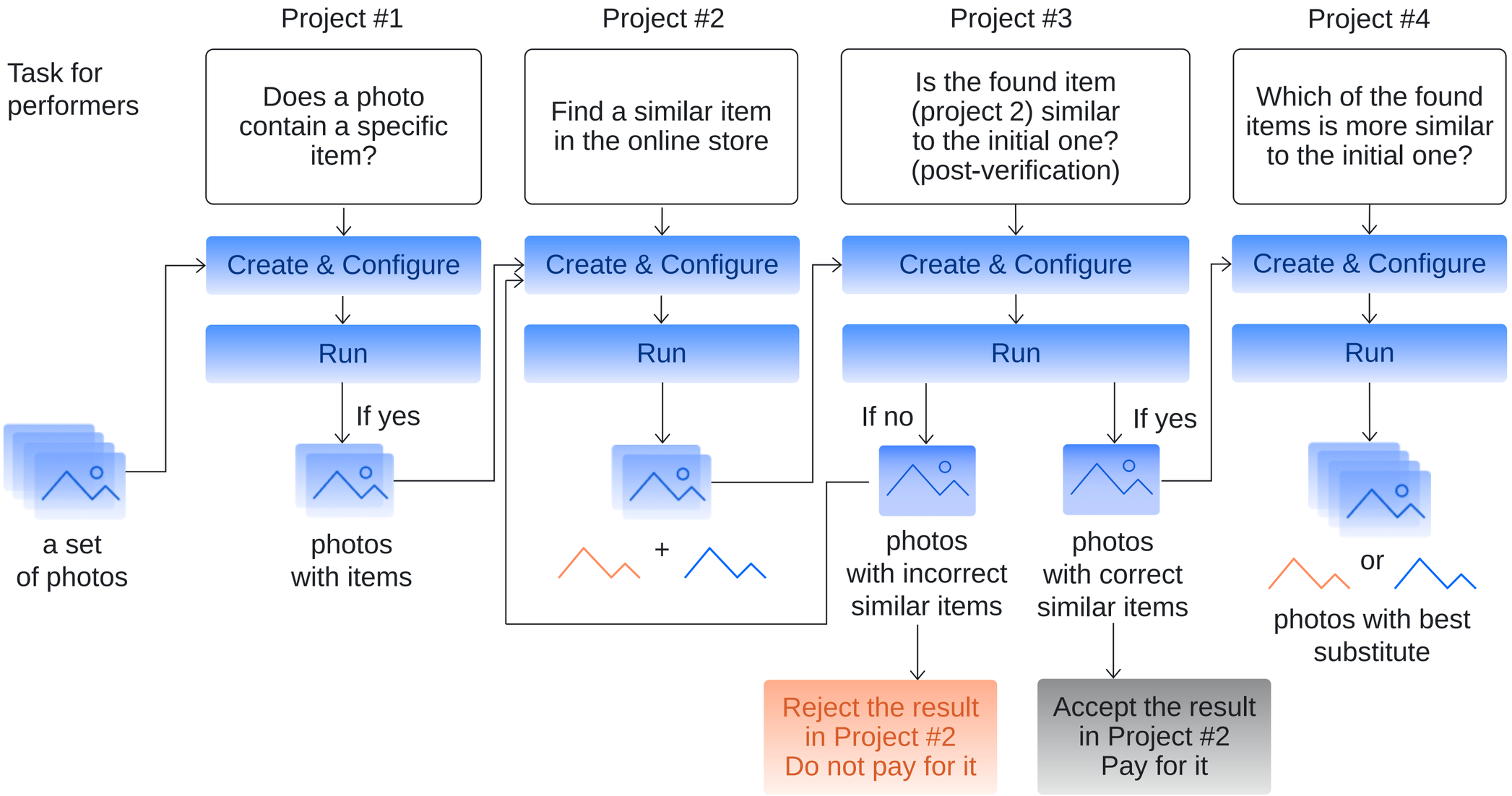
Step #6


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

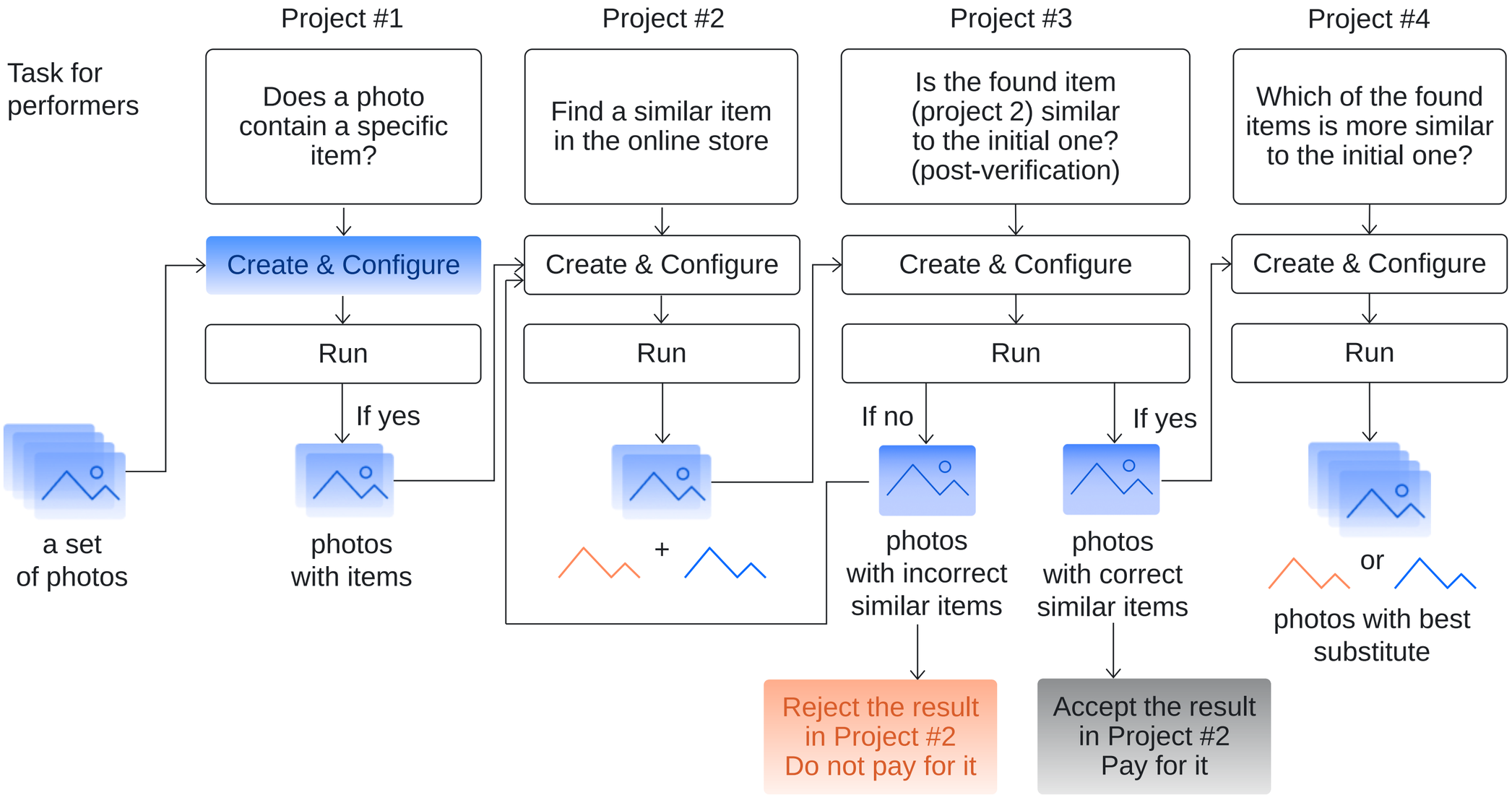
Step #7


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Create & Configure
Create & Configure
Create & Configure
Run
Run
Run
Run
If yes
If no
If yes
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

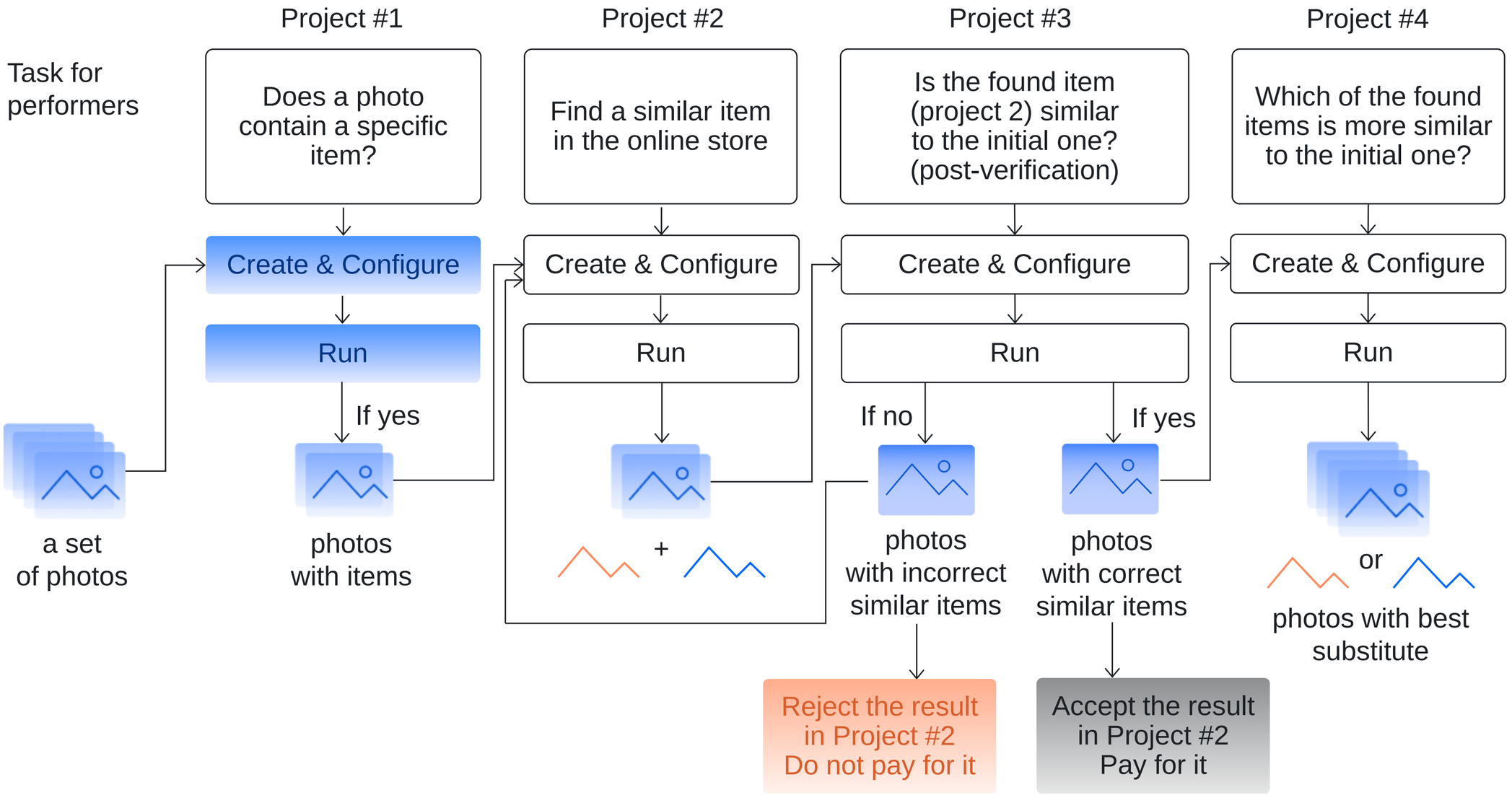
Step #8


Project #1
Project #2
Project #3
Project #4
Task for performers
Does a photo contain a specific item?
Find a similar item in the online store
Is the found item (project 2) similar to the initial one? (post-verification)
Which of the found items is more similar to the initial one?
Create & Configure
Run
If yes
If no
a set of photos
photos with items
+
photos with incorrect similar items
photos with correct similar items
or
photos with best substitute
Reject the result in Project #2 Do not pay for it
Accept the result in Project #2 Pay for it

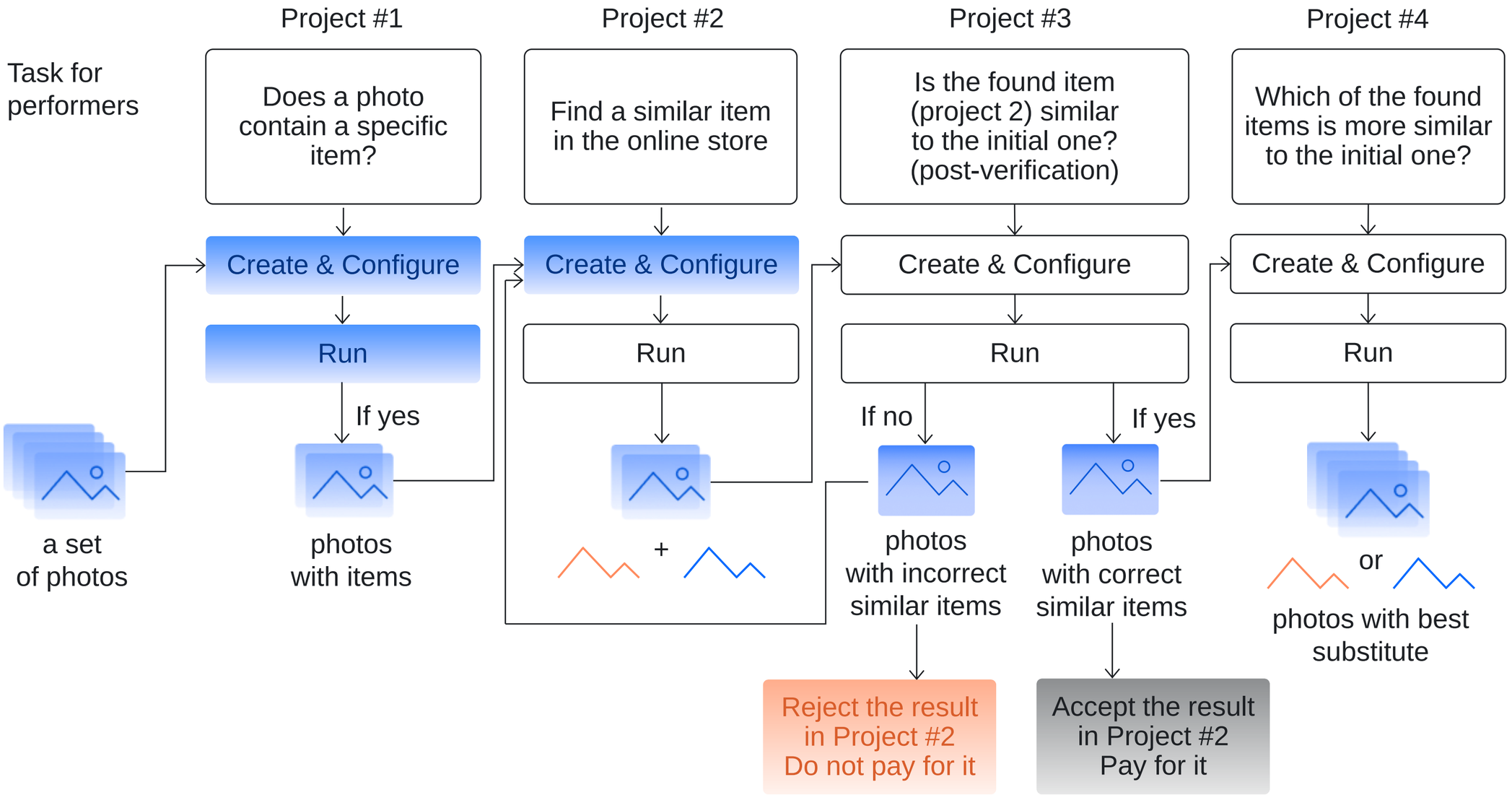
Most of us are at this step



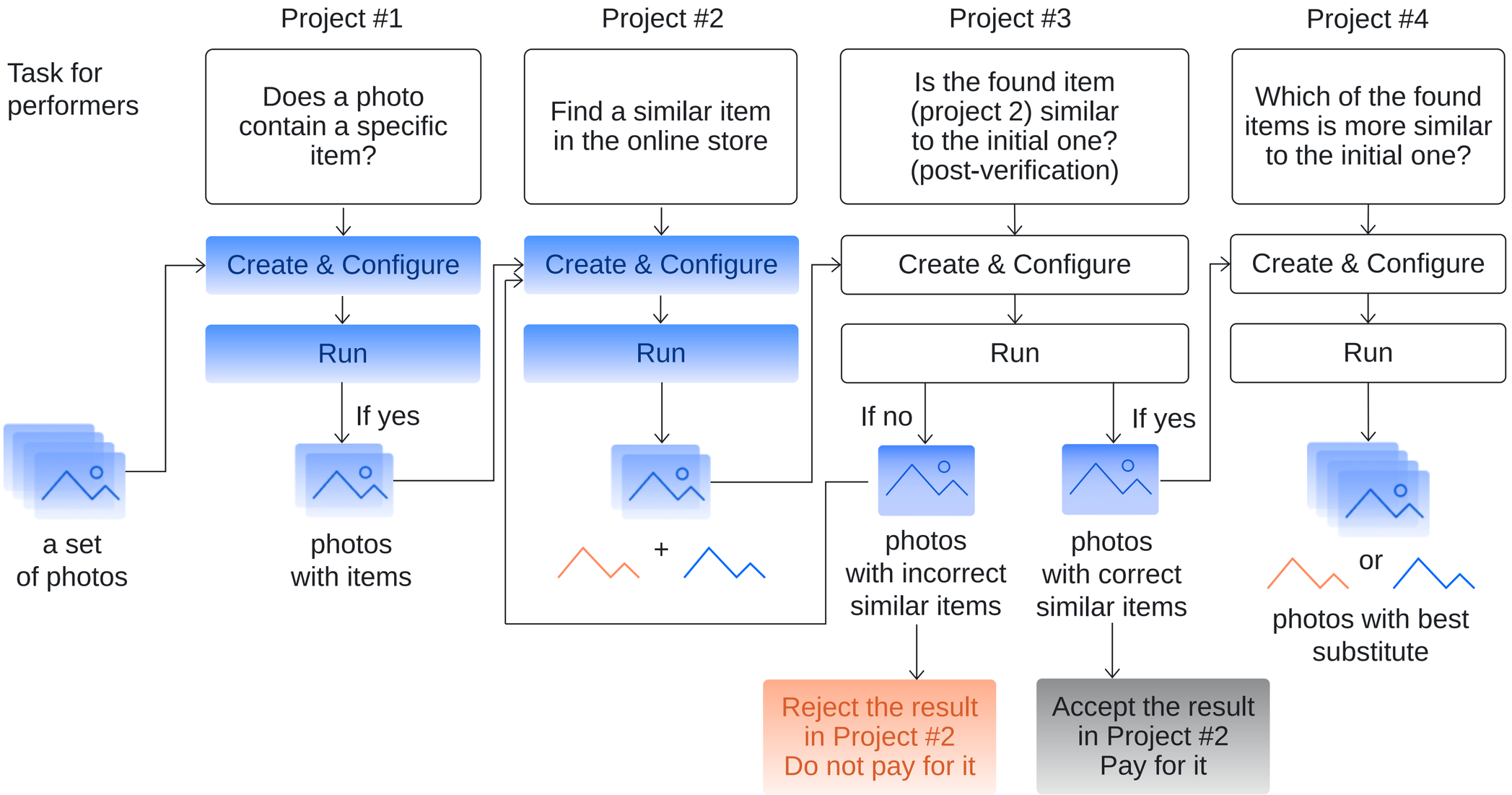
Most of us are at this step



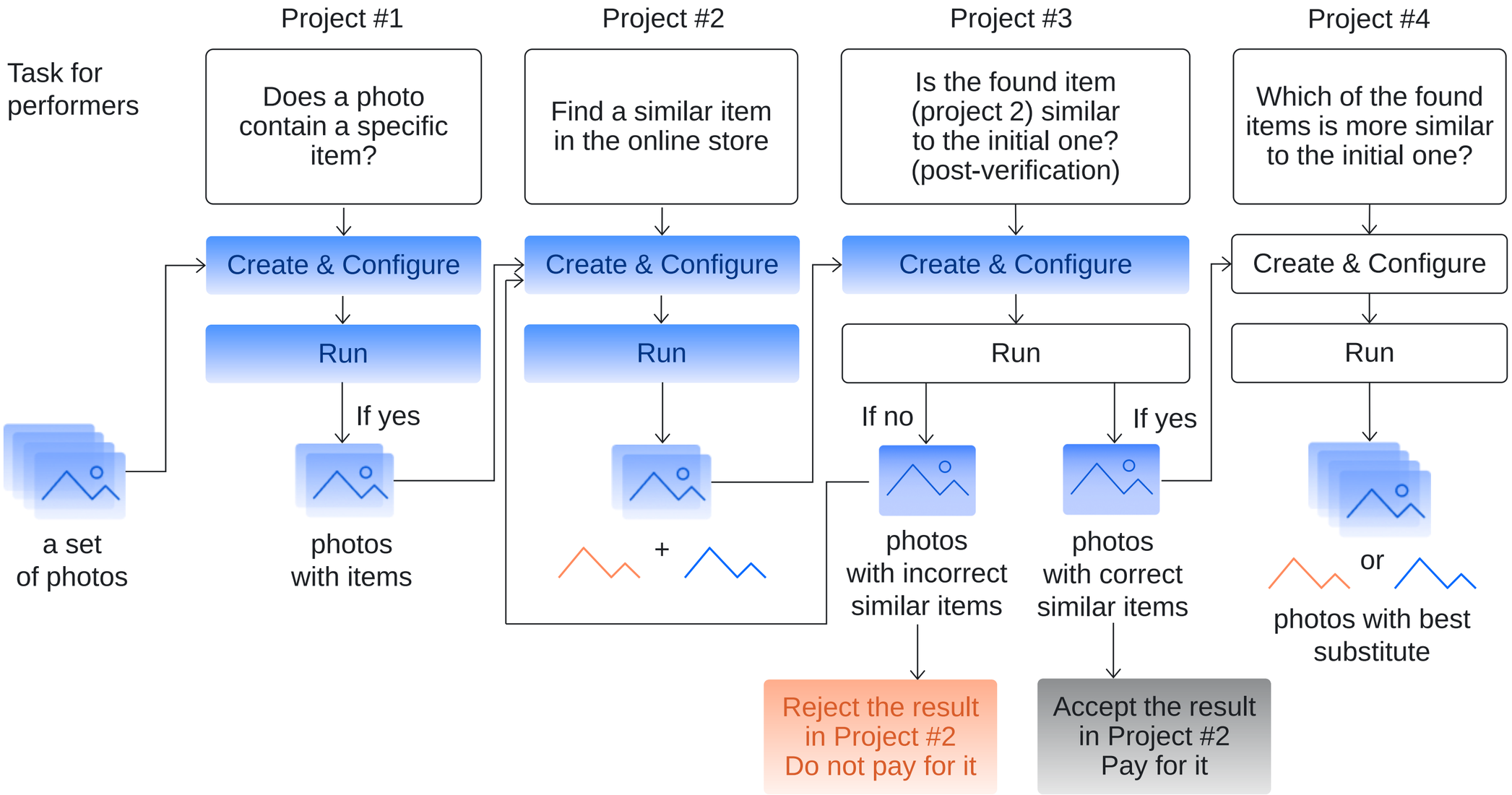
Most of us are at this step



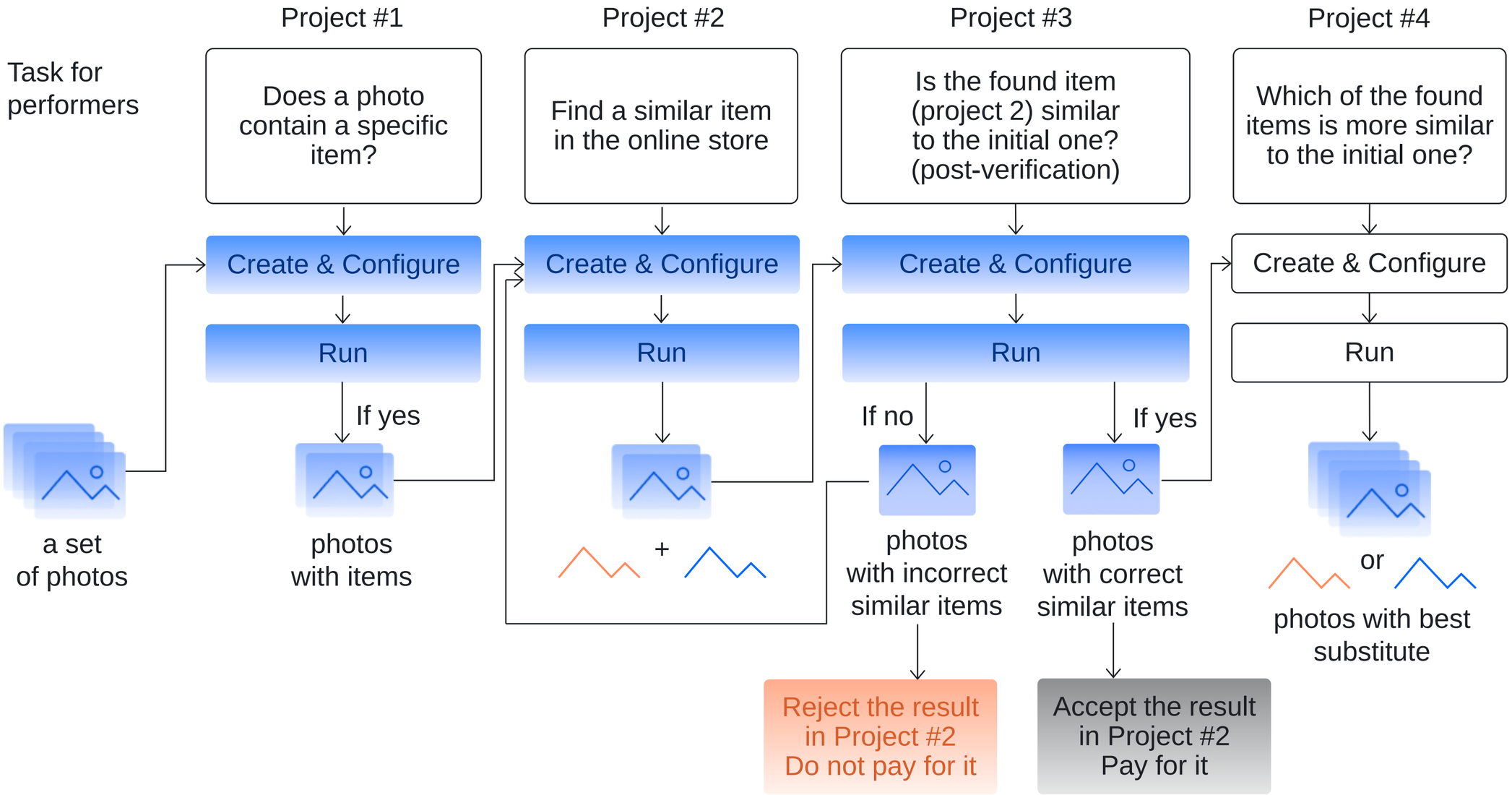
Most of us are at this step



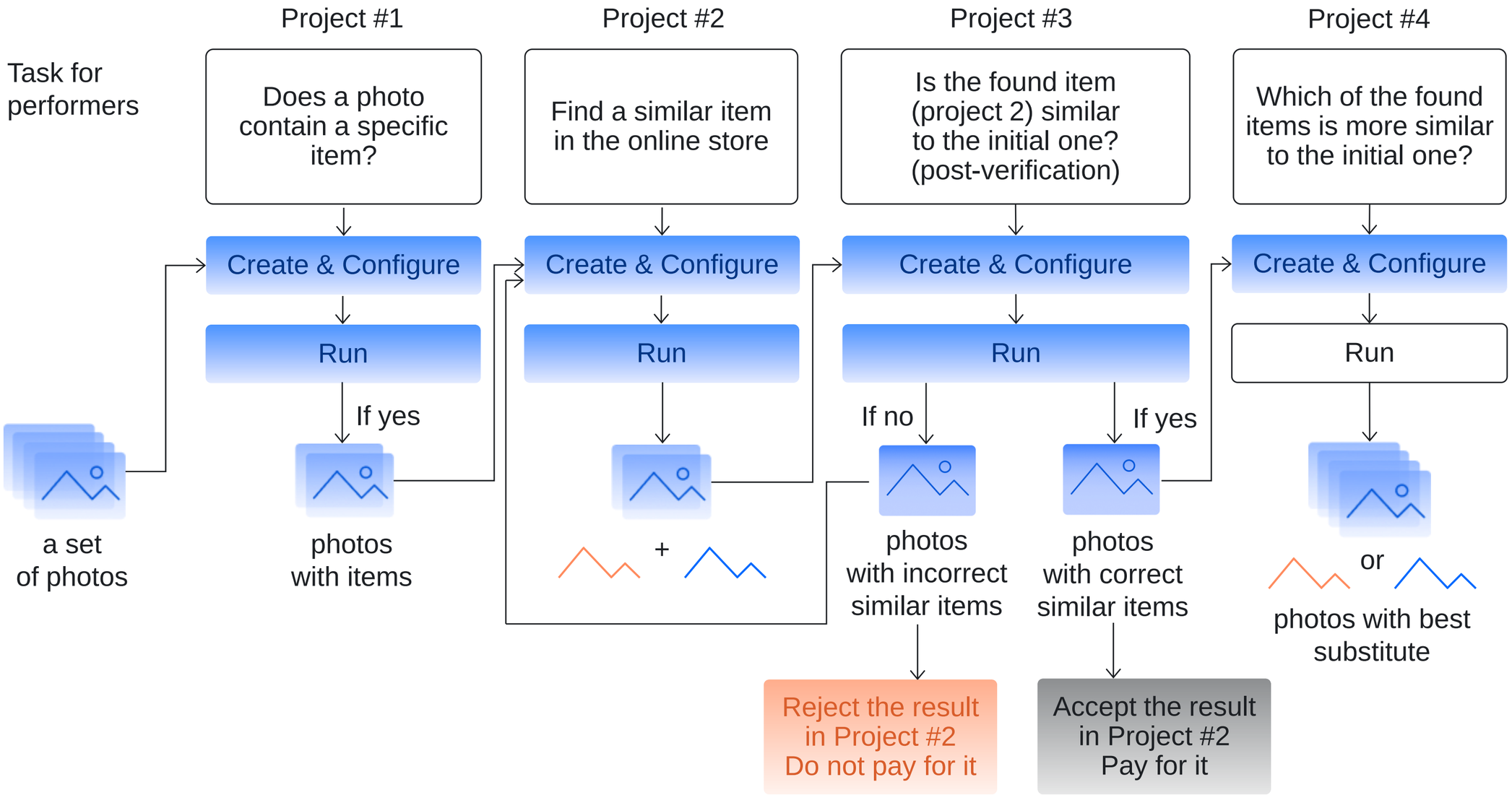
Most of us are at this step



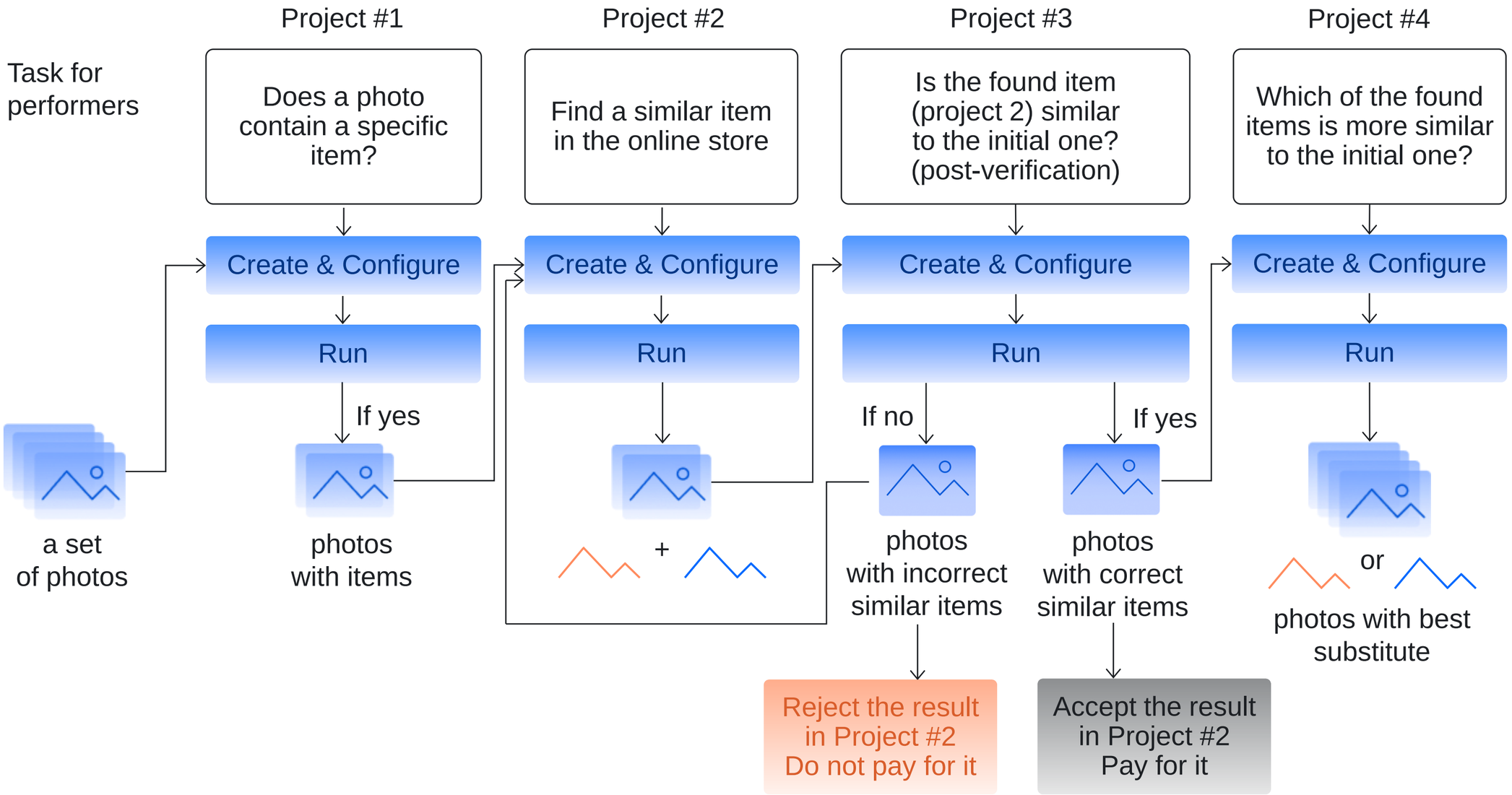
Most of us are at this step



Most of us are at this step



Most of us are at this step

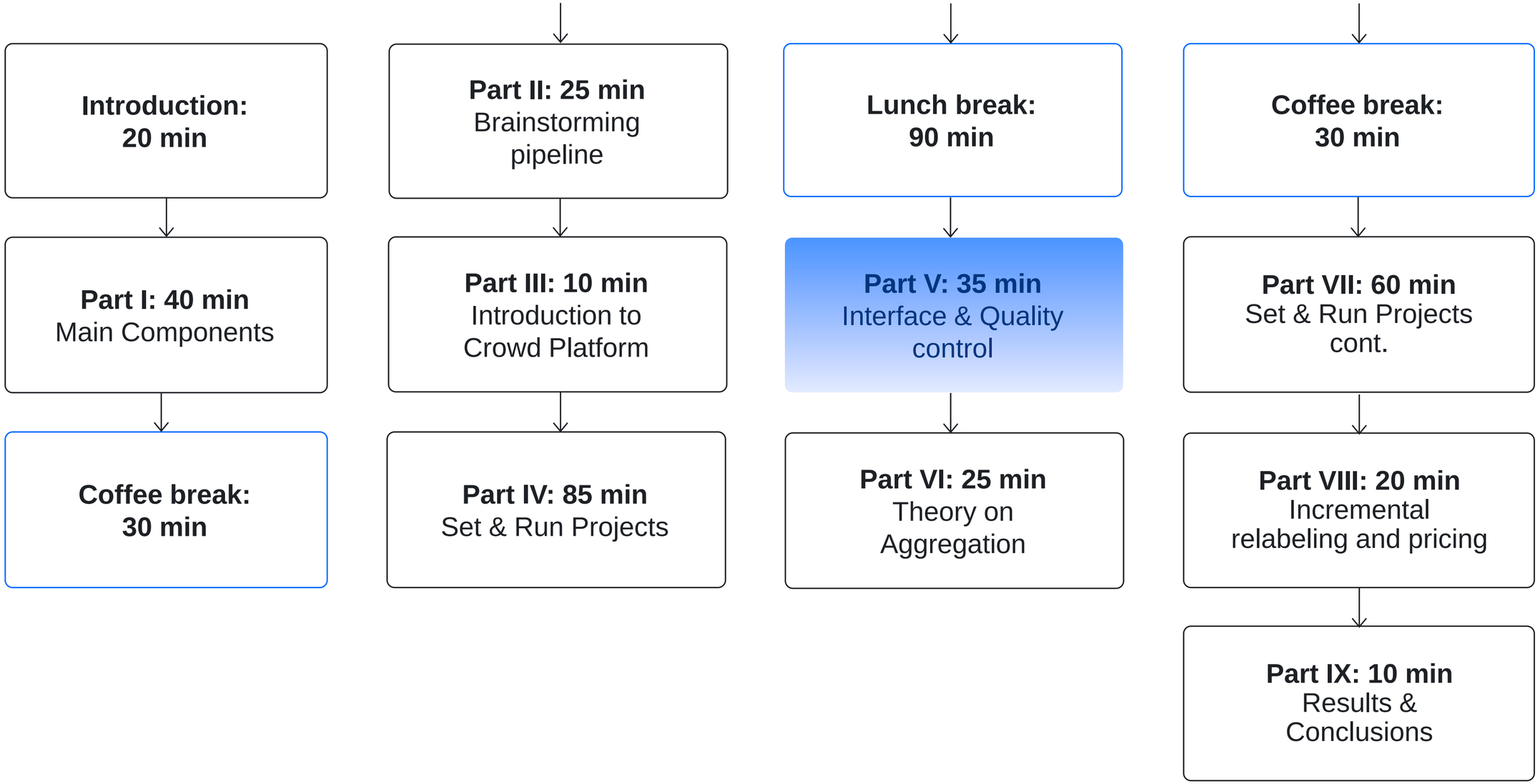


Part V

Effective quality control and task interface: details

Alexey Drutsa,
Head of Efficiency and Growth Division, Toloka

Tutorial schedule



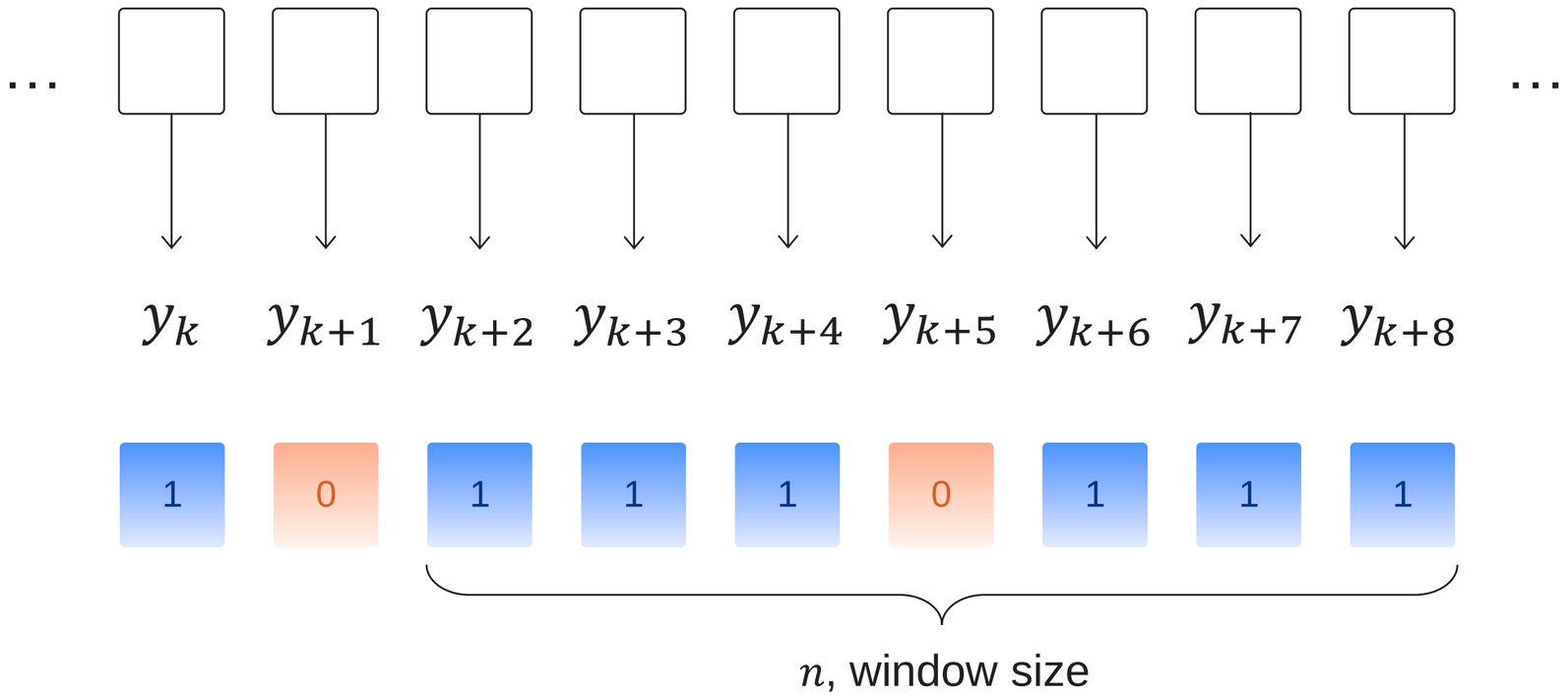
Quality control: the rate of correct answers

Task sequence

Tasks executed by a performer

Signals of answer correctness

For instance, binary, y, $\in \{0,1\}$



Estimation of correctness rate

To estimate the probability of a correct answer use

$$\mathbb{P}(\text{correct}) \approx \frac{1}{n}\sum_{i=1}^{n} y_i \pm \frac{1}{2\sqrt{n}}$$

Window size (n) is a balance between

- Accuracy of the estimate

  and
- Fast reaction to changes in performer quality

Sources for correct answer signal

How can we get y_i?

- Control tasks
- Agreement with aggregated answer (e.g., Majority Vote)
- Post-verification

Control tasks

Pros

- Signal is obtained instantly
- Signal has high confidence on tasks where obtained

Cons

- Tasks for labelling do not provide this signal (→ signal for a fraction of tasks)
- Creation and maintenance of a set of control tasks

Costs (extra charge for quality control)

- Control task creation
- Depends on the frequency of control tasks occurred in the task sequence

You can apply adaptive frequency to optimize costs

Agreement with aggregated answer

Pros

- Easy to implement

Cons

- Signal is obtained with latency
- Works well only if most workers have good quality
- Works well for tasks with small # of answer variants (e.g., classification)

Costs (extra charge for quality control)

- Multiplied by the overlap used

You can apply incremental relabelling to optimize costs

Agreement may fail against coordinated attacks

$$\mathbb{P}(\#m_{bad} > \frac{n}{2}) = \sum_{k=\lceil \frac{n}{2} \rceil}^{n} C_n^k \, p^k (1-p)^{n-k}$$

p is the fraction of coordinated spammers among performers

n is the overlap for Majority Vote model

For instance:
If n = 3 and p = 0.1

The probability of majority with an incorrect answer is 2.8%

in fact, is larger since other performers may accidentally agree with spammers

Post-verification

Pros

- Can be applied to any task type (even with a sophisticated answer)

Cons

- Signal is obtained with latency
- Requires efforts to construct a pipeline

Costs (extra charge for quality control)

- Cost of verification tasks

You can apply selective verification to optimize costs

Non-binary penalty

You can set different penalty $y_i \in [0, 1]$ for different signals

For instance:

- Task consists of several answers of different importance
- Level of confidence of the aggregated answer
- Level of expertise of the performer who post-verifies

Quality control: undesired behavior

Performer behavior

Correct answers to your tasks are not the sole signal of performer quality

For instance, take care of such characteristics:

- Time of task execution
- Usage of UI control elements within task execution
- CAPTCHA

Use them to filter out (ban) performers with low quality of high confidence

Fast responses

There is a lower bound on time required to execute your task with good quality

- Estimate this time based on behavior of a set of performers
- Calculate the number or the rate of tasks executed too fast

Verification of action execution

Some tasks require usage of certain UI control elements

For instance:

- Check whether a link has been visited
- Check whether a video has been played

CAPTCHA

Instead of revoking access to your tasks, you can ask crowdsourcing platform to show CAPTCHA to a performer

You get an additional signal to decide whether you face a robot or not

Quality control: skills

Skill is a variable assigned to a performer

Can be used to automatically calculate

- Answer correctness rates (via control tasks, agreement, post-verification)
- Behavioral features (e.g., fast response rate)
- Binary information on execution of particular projects
- Any their combinations and other features

Can be used for automatic decision making

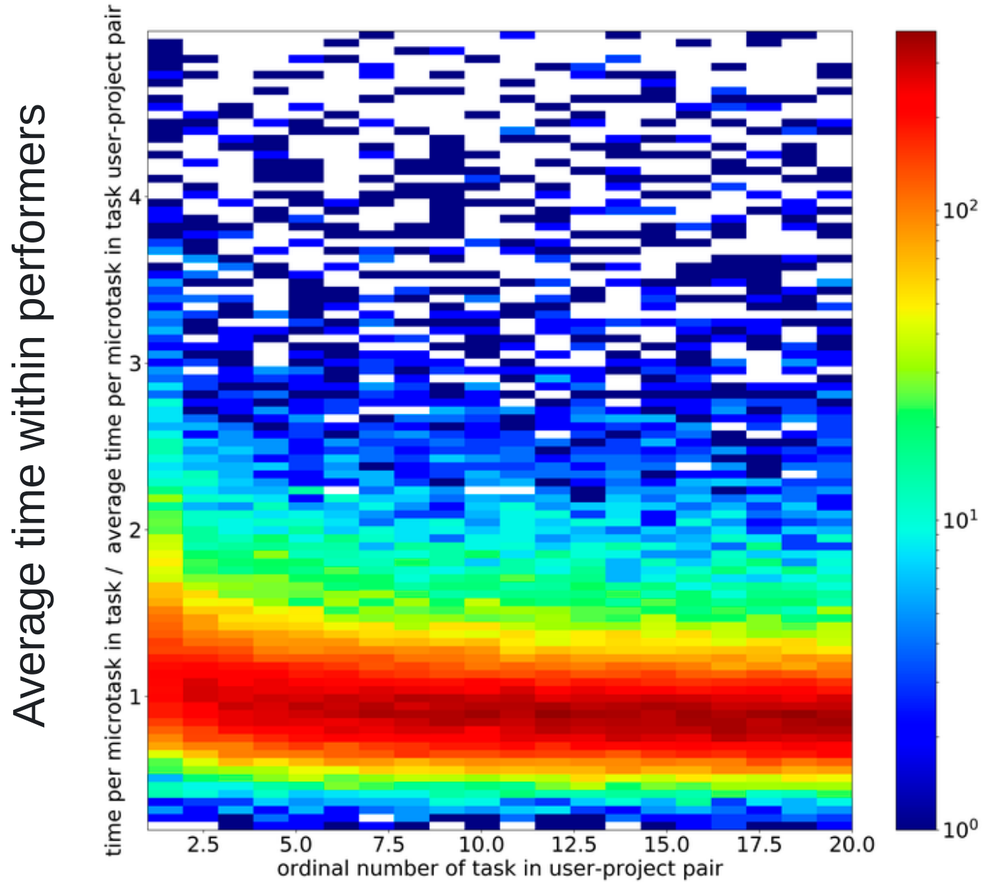
- Access control to certain projects and tasks
- e.g., revoke access to your tasks if a skill becomes too low

Thinking (cogitation) vs reflexes

Skills based on a single signal are easy to game

It is difficult to force
a performer to think (cogitate)
instead of to use/train reflexes

A representative crowd project



tasks made by a performer

Best practice for a good skill

Combine different signals to get a skill robust to gaming

- Combine agreement signal with control tasks or post-verification
- Add behavioral information: execution time, CAPTCHA, etc.

Use this skill in quality-based pricing

Quality control: performer life cycle

Training task

Train performers to execute your tasks

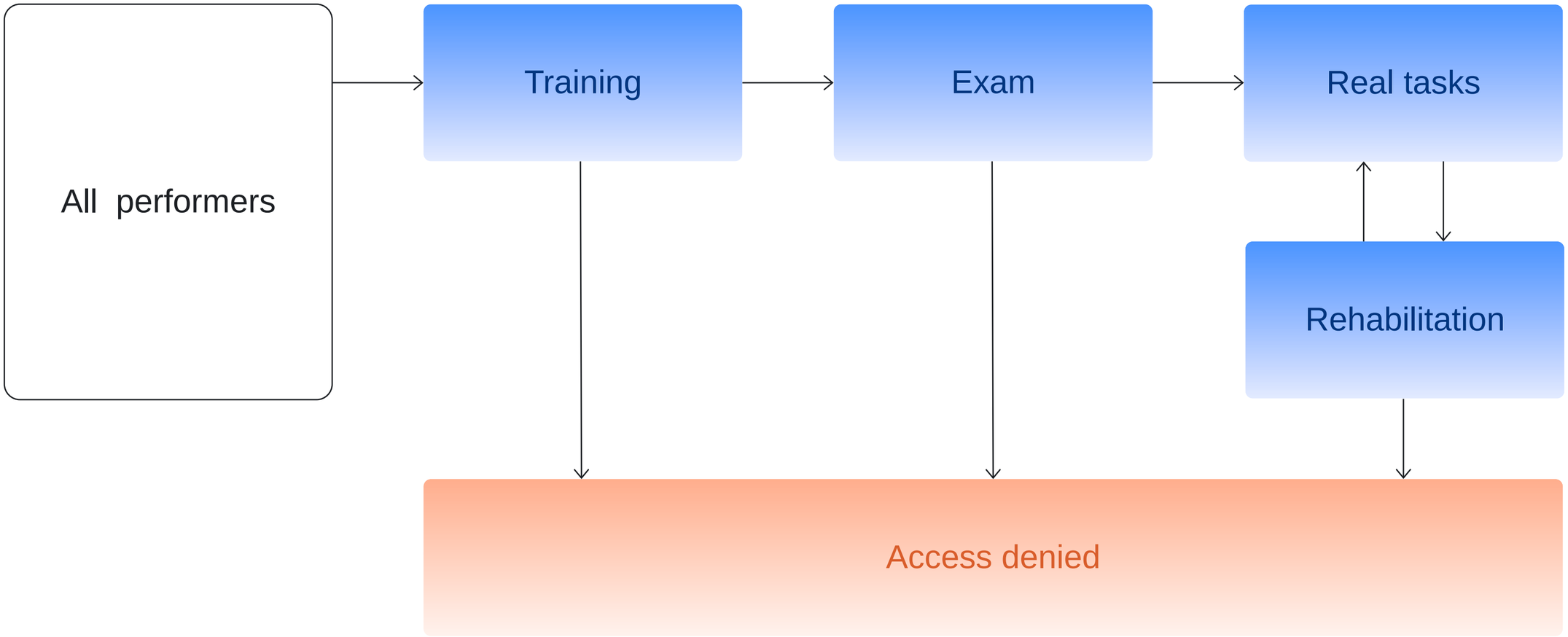
- All tasks are control ones
- There are hints that explain incorrect answers

Exam task

Control the results of training

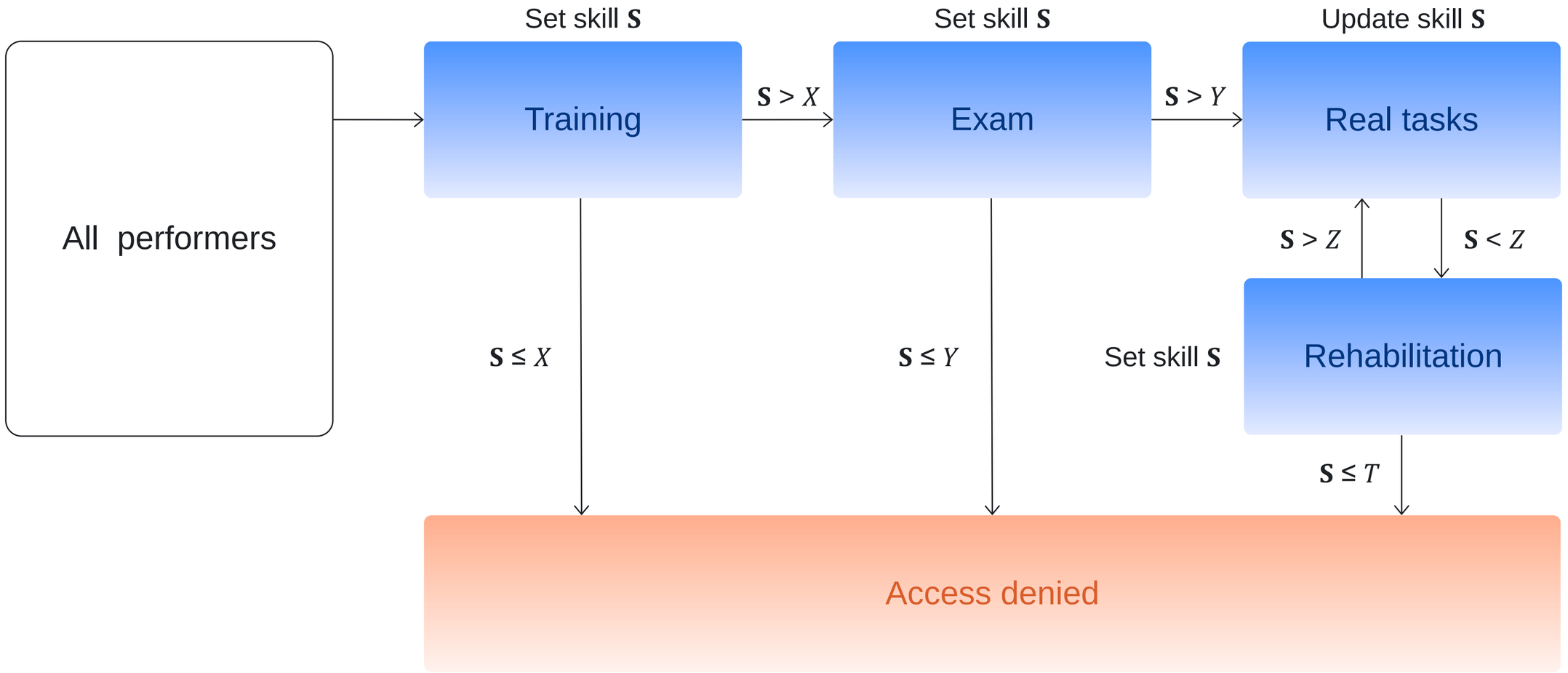
- All tasks are control ones
- No hints and explanations
- A good exam should be:
  - Passable
  - Regularly updated
  - Small

Recommended life cycle of performers



Recommended life cycle of performers

Let quality be controlled by means of a skill **S**



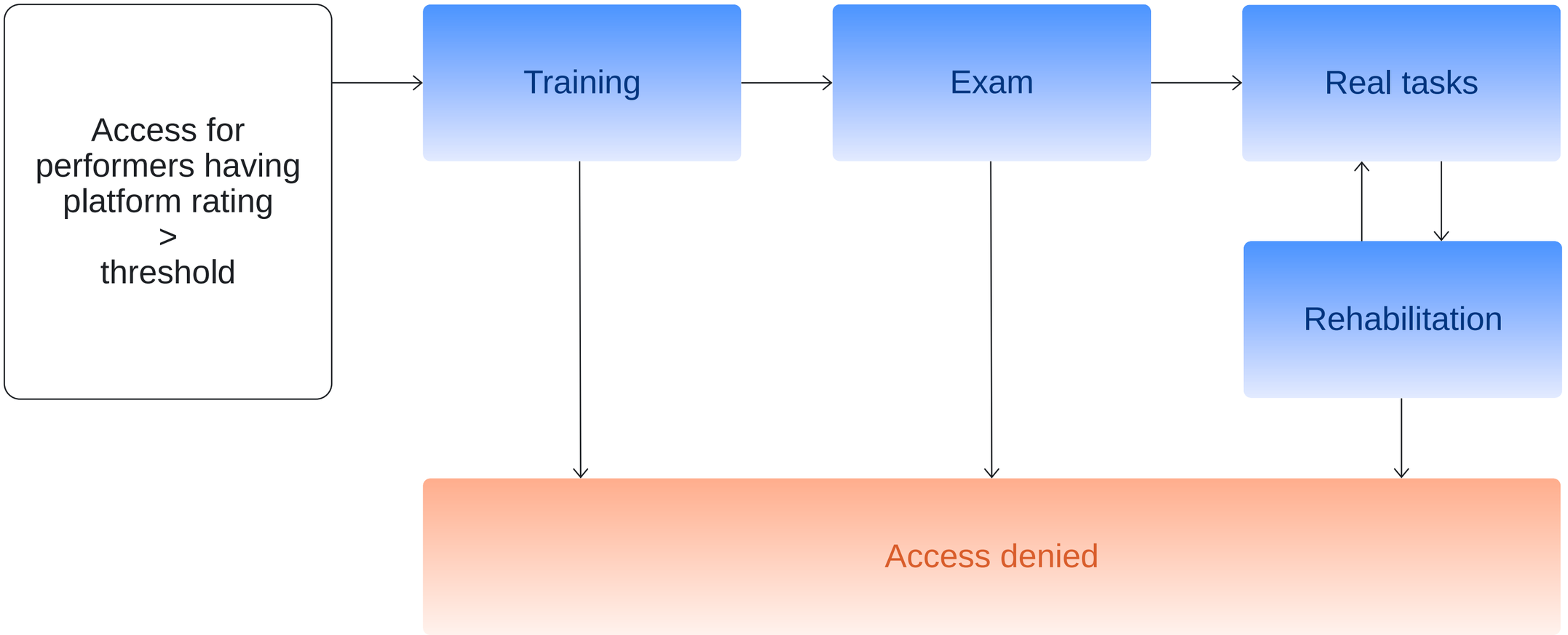
Rehabilitation task

Give a change to those who failed the skill threshold accidentally

- Rehabilitation is similar to an exam task, but with another access criterion
- Remind that there is a chance to observe low quality of a good performer

$$\mathbb{P}(\text{correct}) \approx \frac{1}{n}\sum_{i=1}^{n} y_i \pm \frac{1}{2\sqrt{n}}$$

Grant initial access to top performers



Platform rating*

is calculated based on performer behavior on all existed tasks within the platform

* is available on Toloka

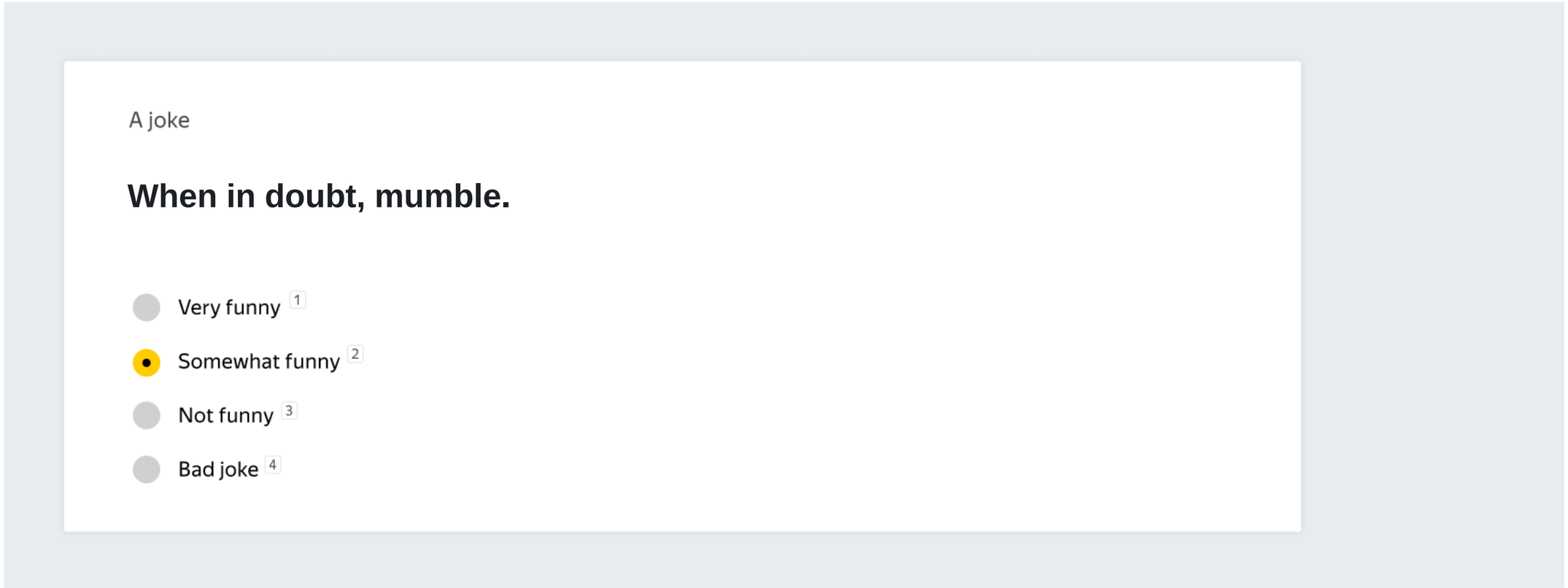
Interface. Introduction

Task in the eyes of the performers

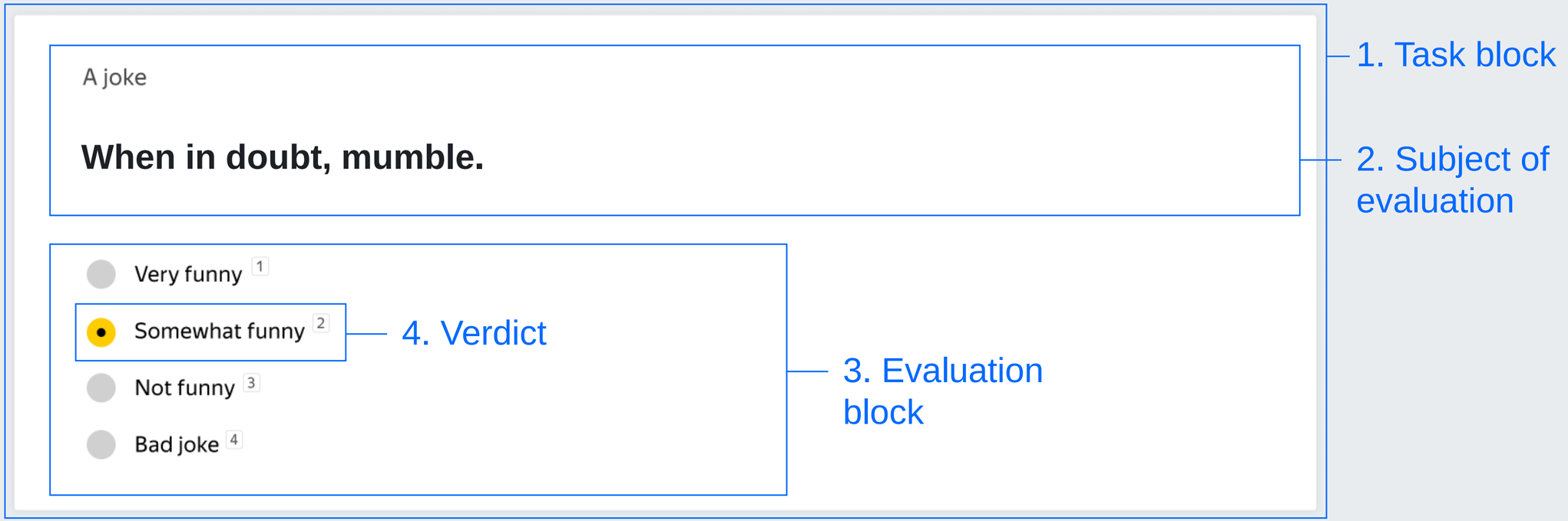
Web-page with specific features

- Long run time
- Repetitive actions
- Concentration
- Speed

Structure of a task interface



Structure of a task interface

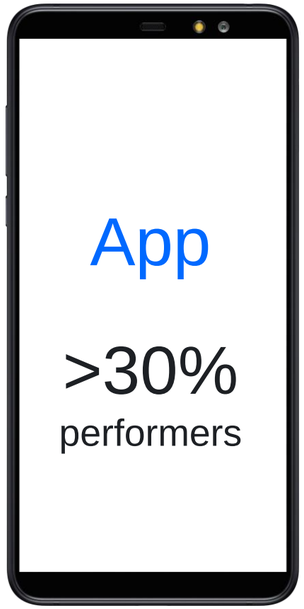


9 golden rules of interface structure

Why is it important?

- Performer’s time
- Speed and data labelling volumes
- Manager’s time
- Quality of the results
- Project’s rating
- Task simplification thanks to the interface

Rule #1. Cross-platform compatibility



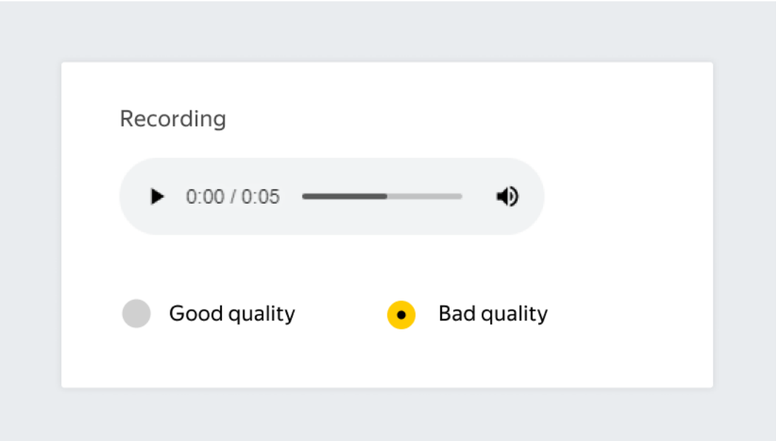
Possible limitations for mobile services:

- Task difficulty
- Media Content, Devices, and Browsers

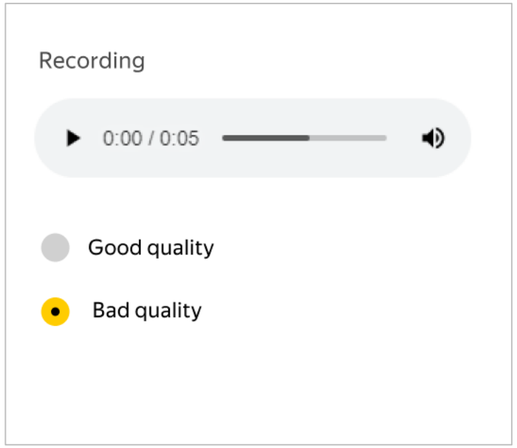
Rule #1. Cross-platform compatibility

Task: evaluate sound quality in wav audio files

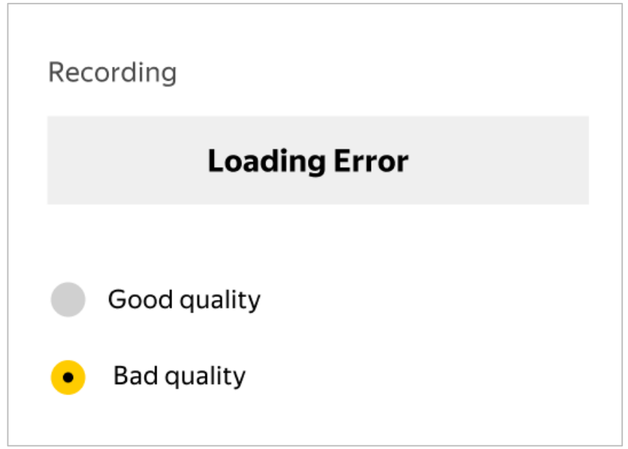
Web version



Android App



IOS App



Rule #1. Cross-platform compatibility

Task: draw a polygon around every road sign

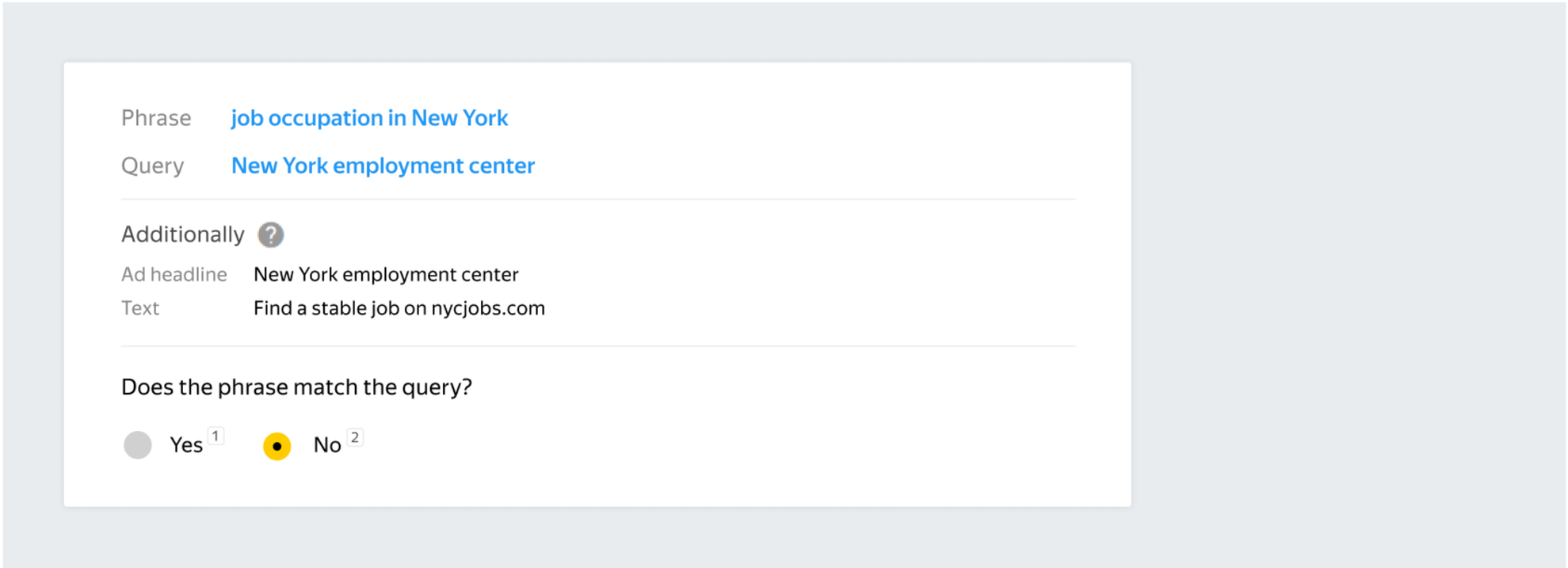
Rule #1. Cross-platform compatibility

Task: draw a polygon around every road sign

Challenge: to outline every single road sign

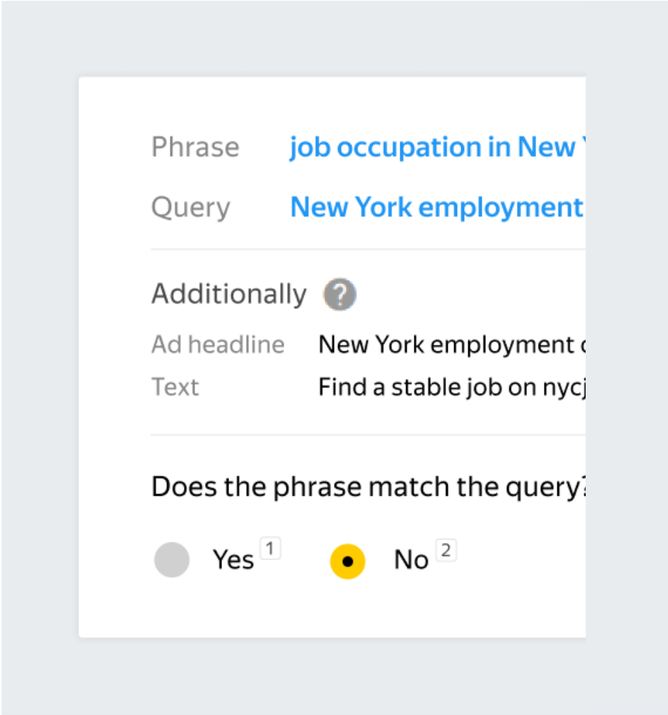
Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match



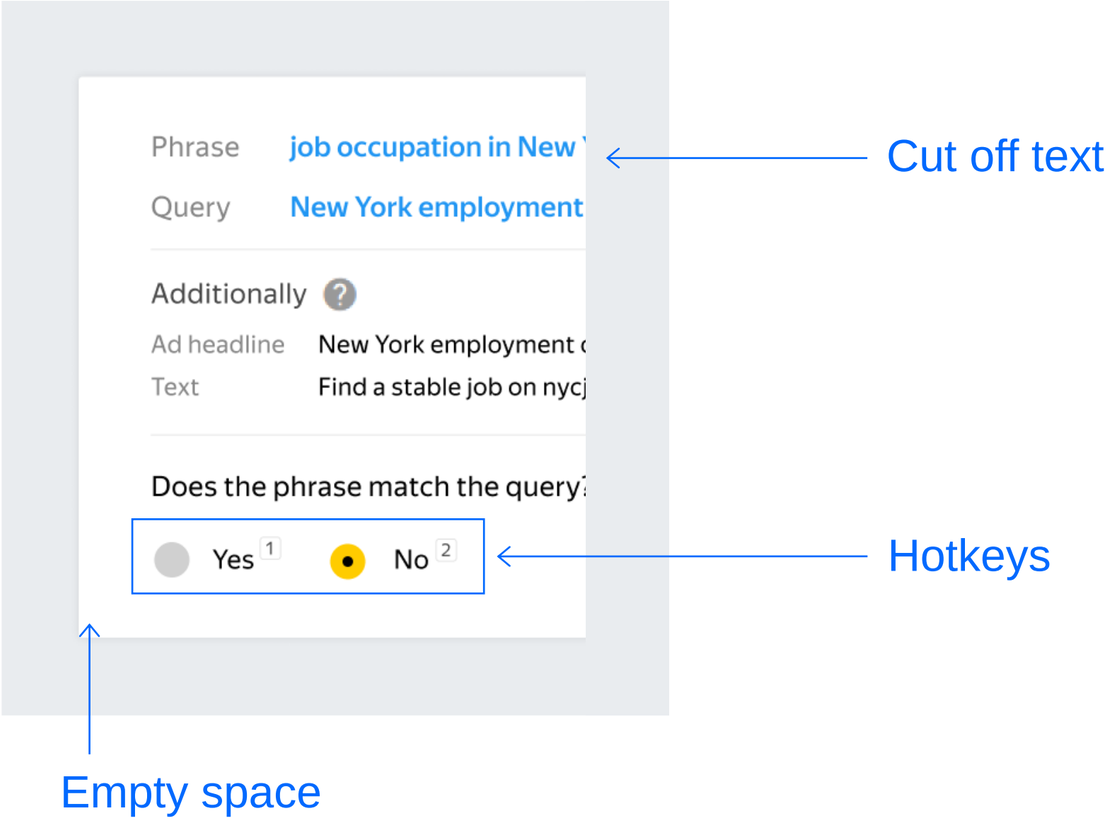
Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match



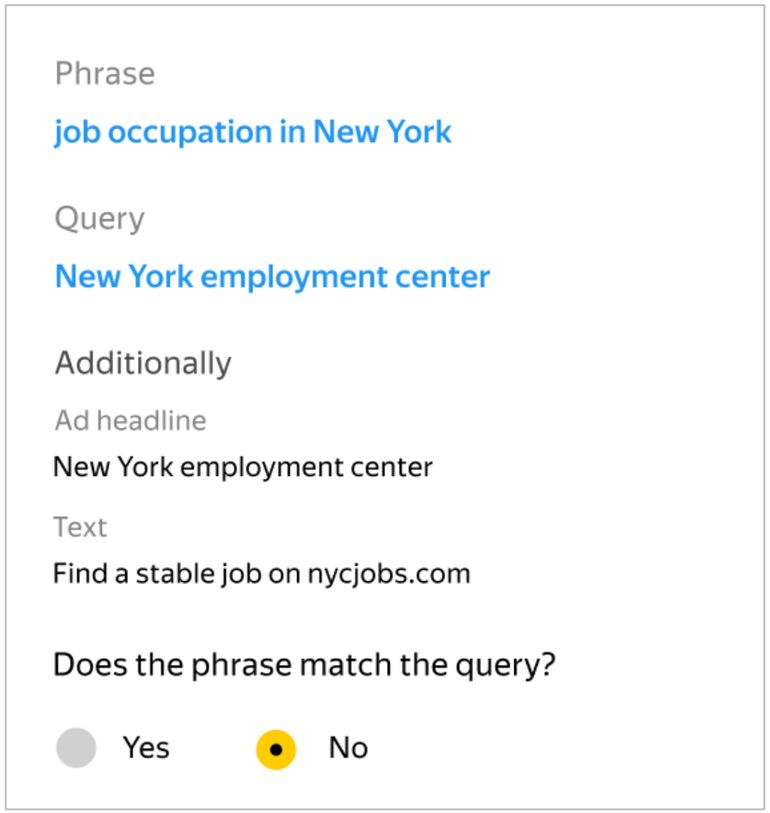
Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match



Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match



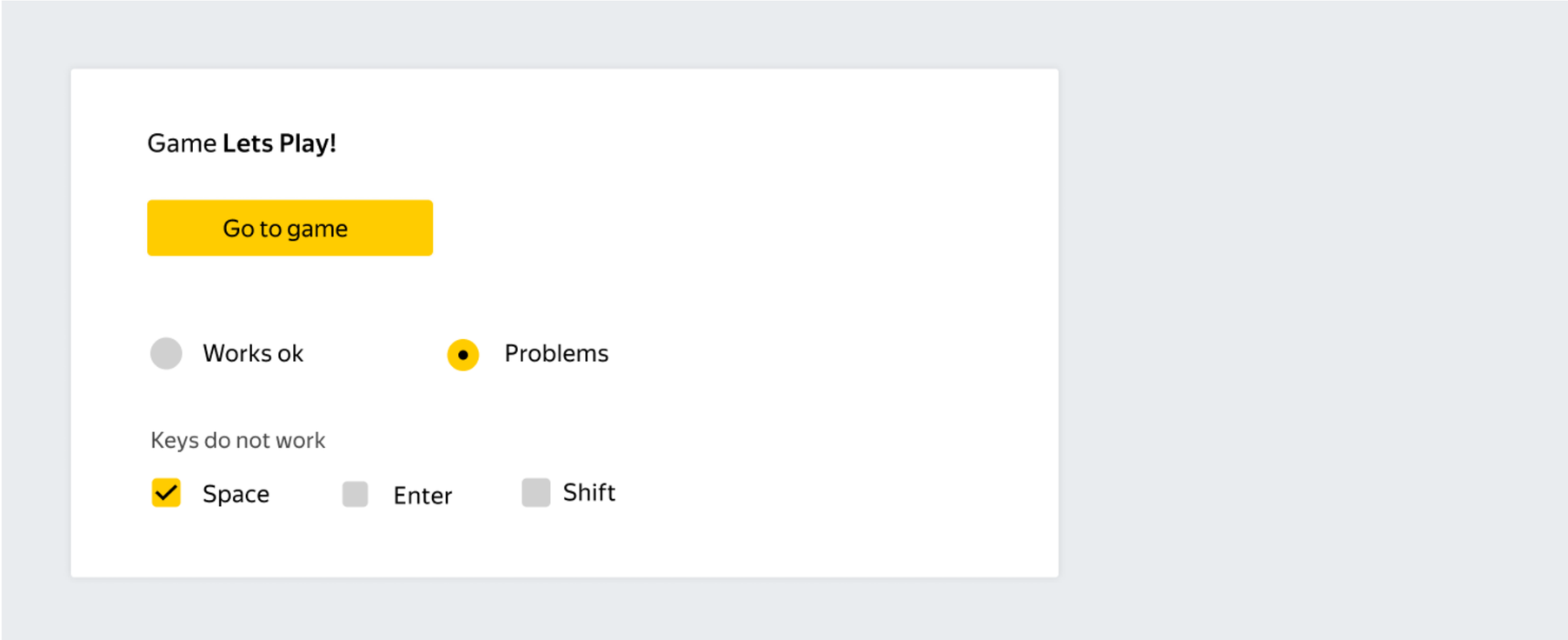
Rule #2. Hotkeys

- Used by about 28% of performers
- Affect task completion speed
- You can assign hotkeys to any action
- Hidden hotkeys should be documented

Ideal scenario: the task can be completed without using a mouse

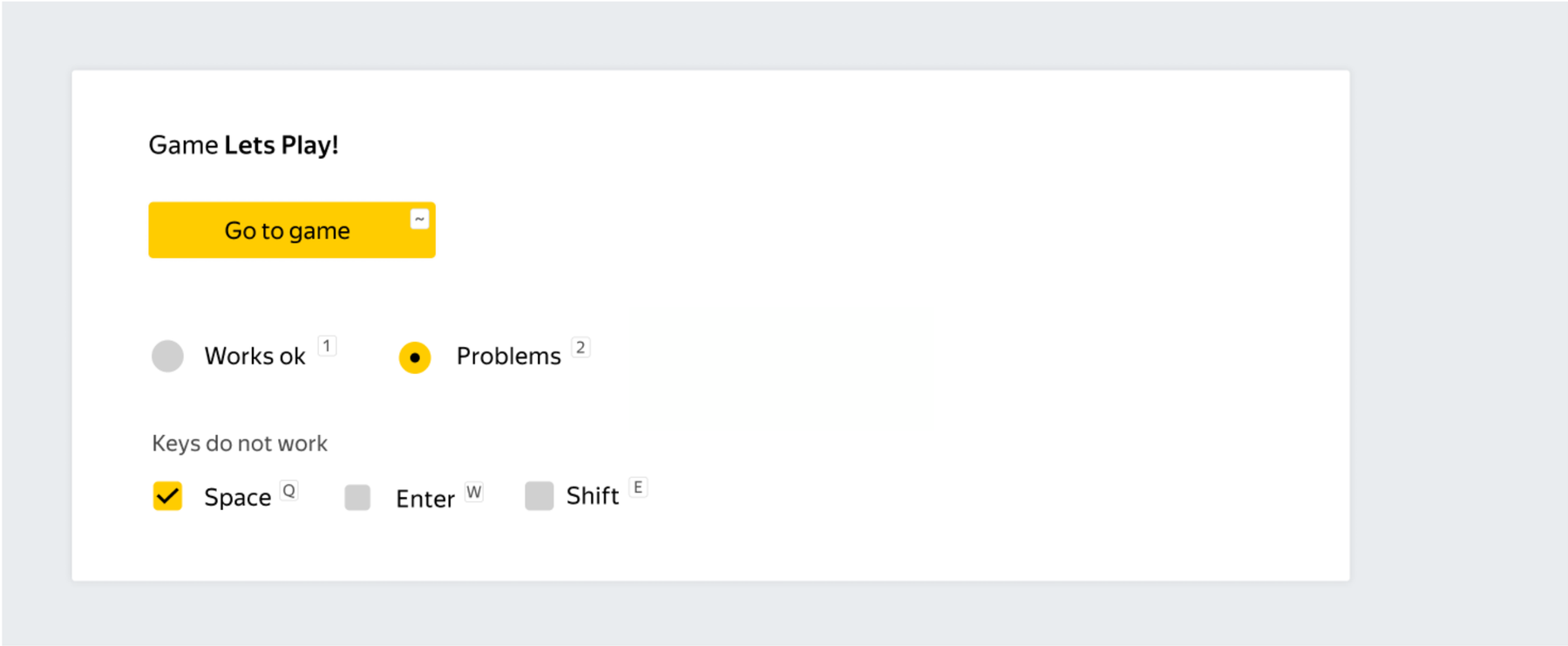
Rule #2. Hotkeys

Task: evaluate functionality of a game in a browser (works with a keyboard)



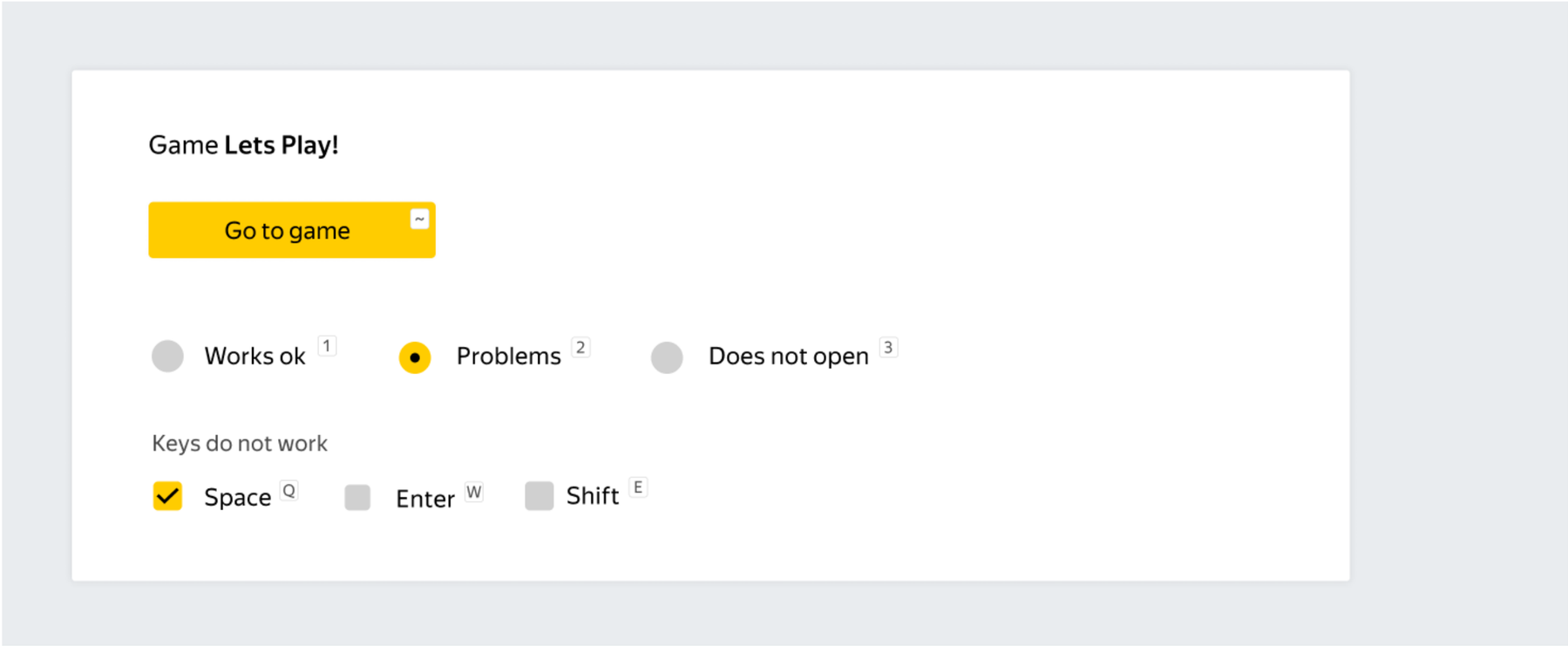
Rule #2. Hotkeys

Task: tell whether the game works in a web browser (works with a keyboard)



Rule #2. Hotkeys

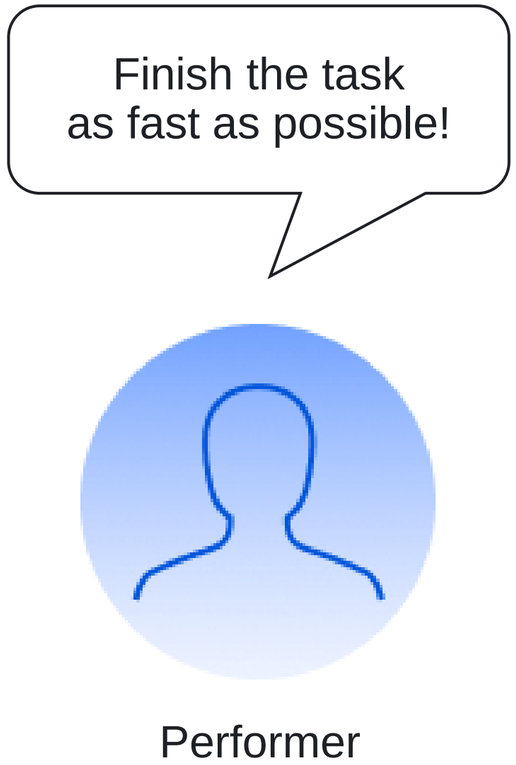
Task: tell whether the game works in a web browser (works with a keyboard)



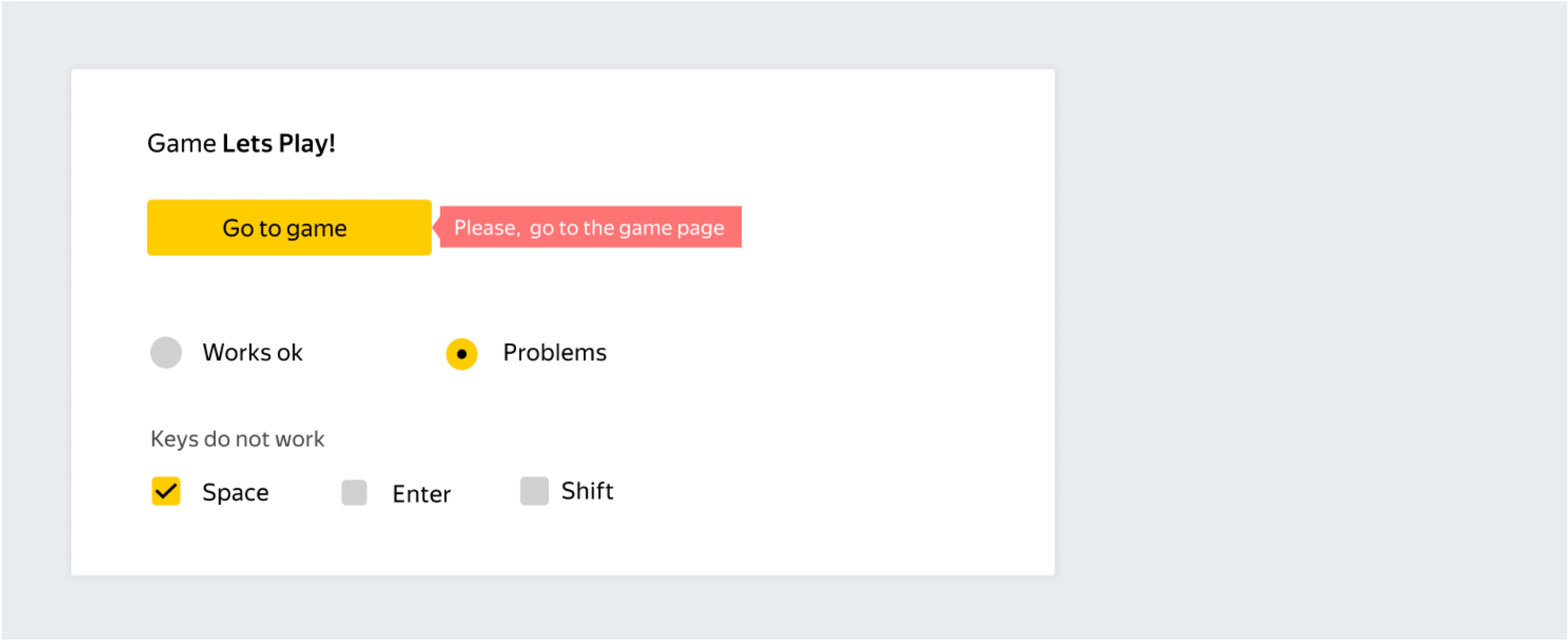
Rule #3. Action and data check

We can check if the performer:

- Watched the video or listened to the audio
- Went to external resources
- Provided correct input data
- Spent enough time on each task



Rule #3. Action and data check



Rule #4. Test the task

Always test the task before publishing it

- Preview option
- Test task pool in Toloka sandbox

Rule #5. Minimize external resources usage

Spoiler: not always applicable

- Impossible to control performer’s actions outside of the task interface
- External resources might not always work properly

Rule #5. Minimize external resources usage

- Show all information inside the task
- Copy data to your own storage
- Check performers’ actions and their input data

Idea: show screenshots instead of the links

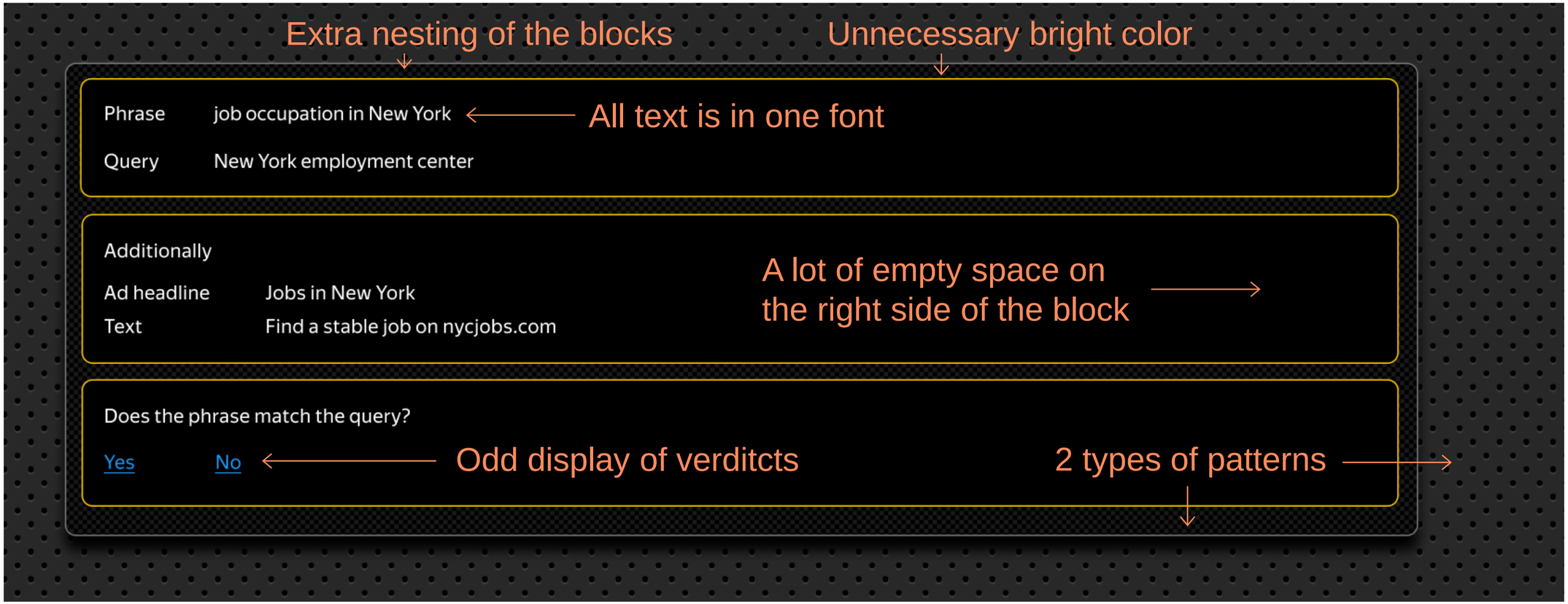
Rule #6. Avoid experimental design

Signs

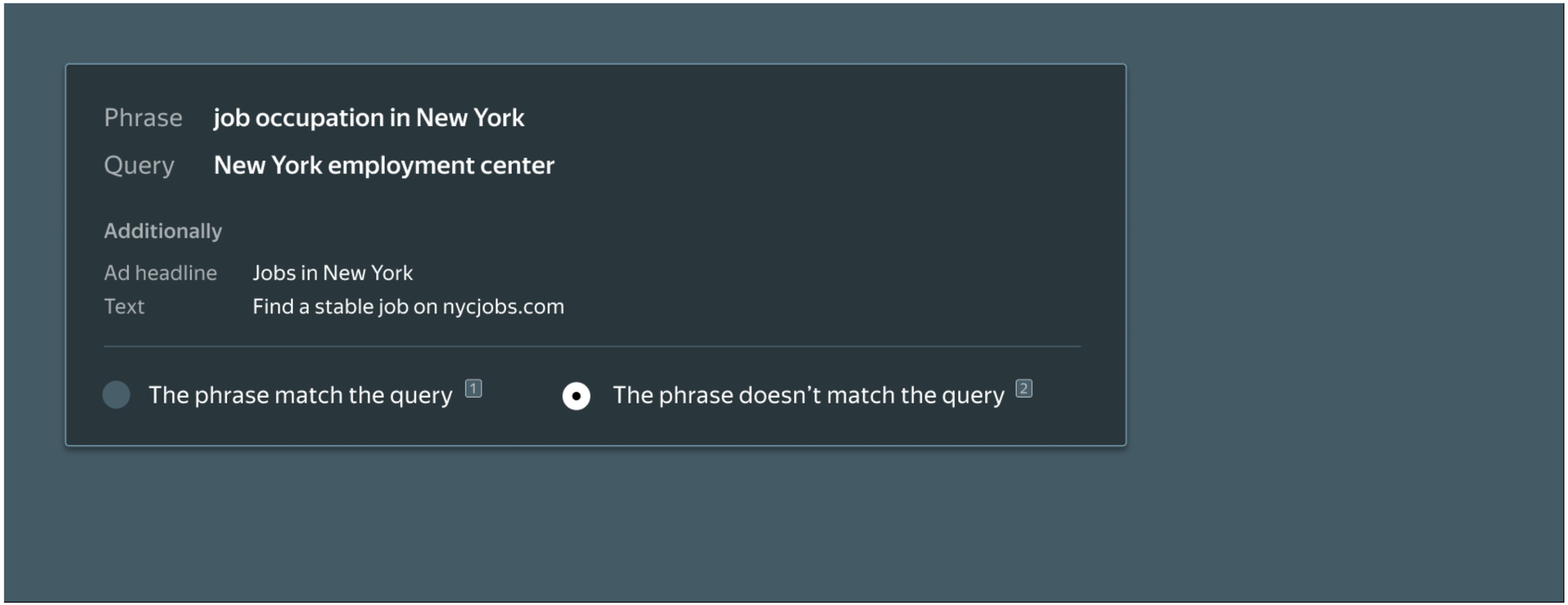
- *Odd layout of typical interface elements*
- Variety of bright and different colors
- The presence of conspicuous elements with an exclusively artistic function

Rule #6. Avoid experimental design

Rule #6. Avoid experimental design



Rule #6. Avoid experimental design

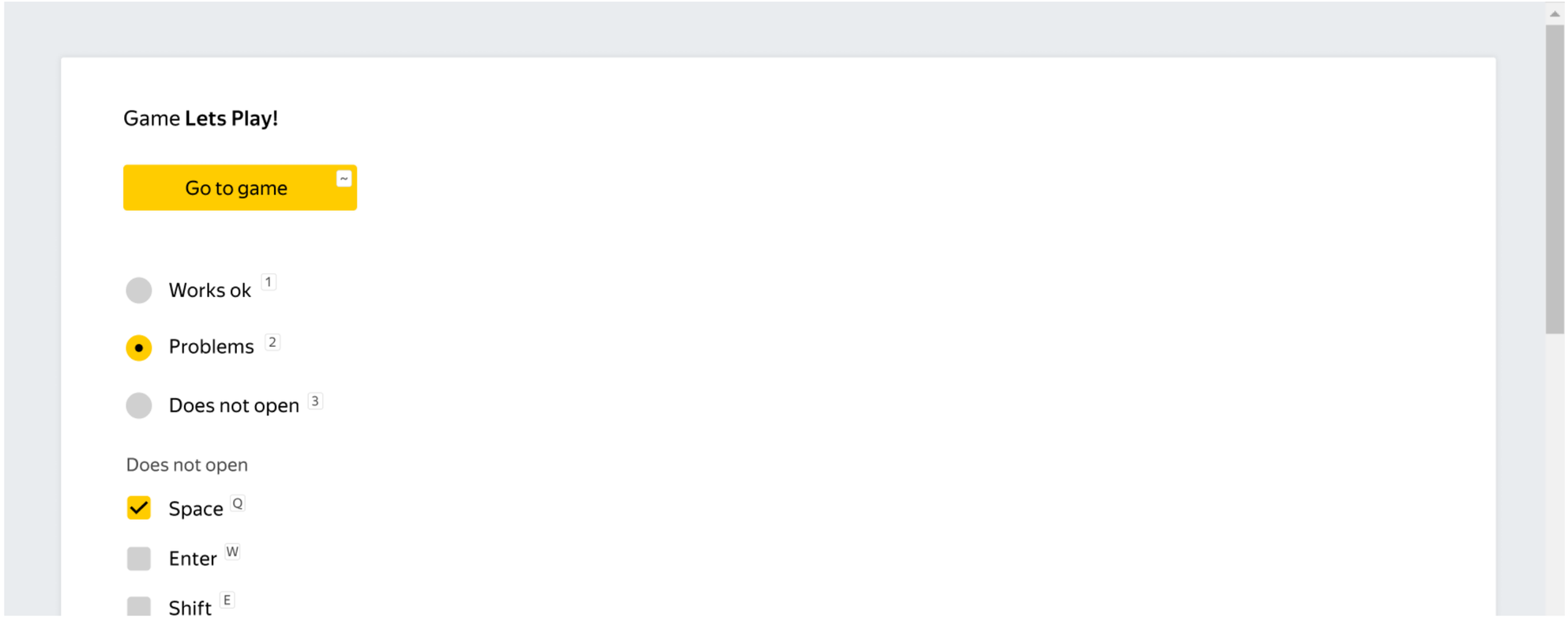


Rule #7. Efficient space usage

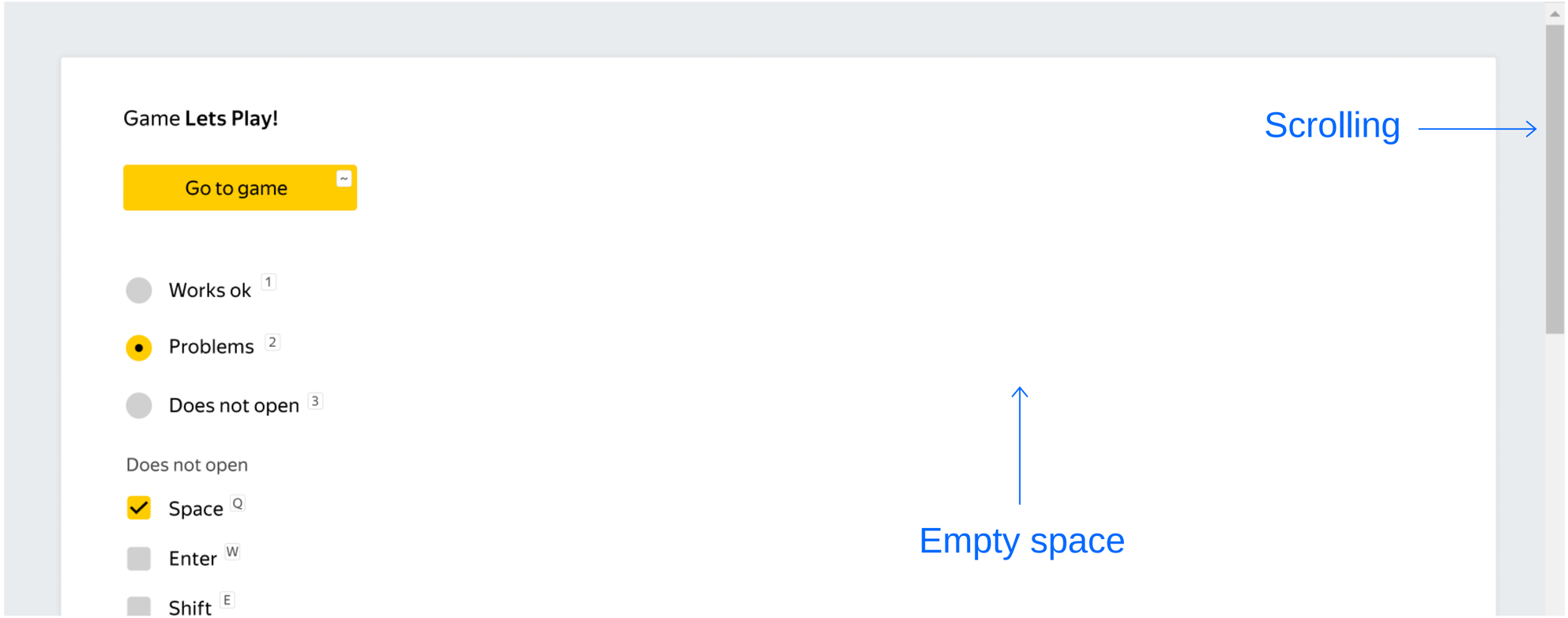
- Group the elements within your task block
- Absence of empty spaces
- Highlight most important information

Ideal scenario: one task perfectly fits the size of a monitor

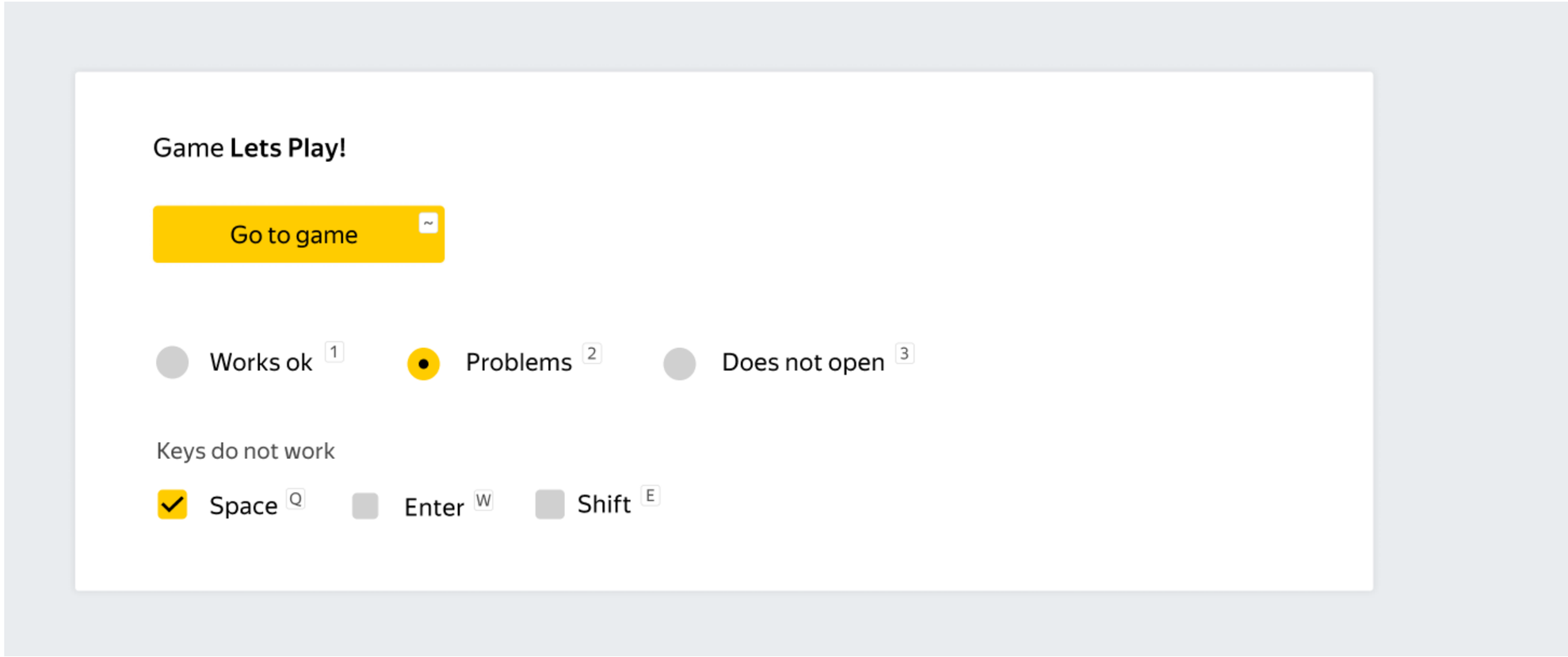
Rule #7. Efficient space usage



Rule #7. Efficient space usage



Rule #7. Efficient space usage



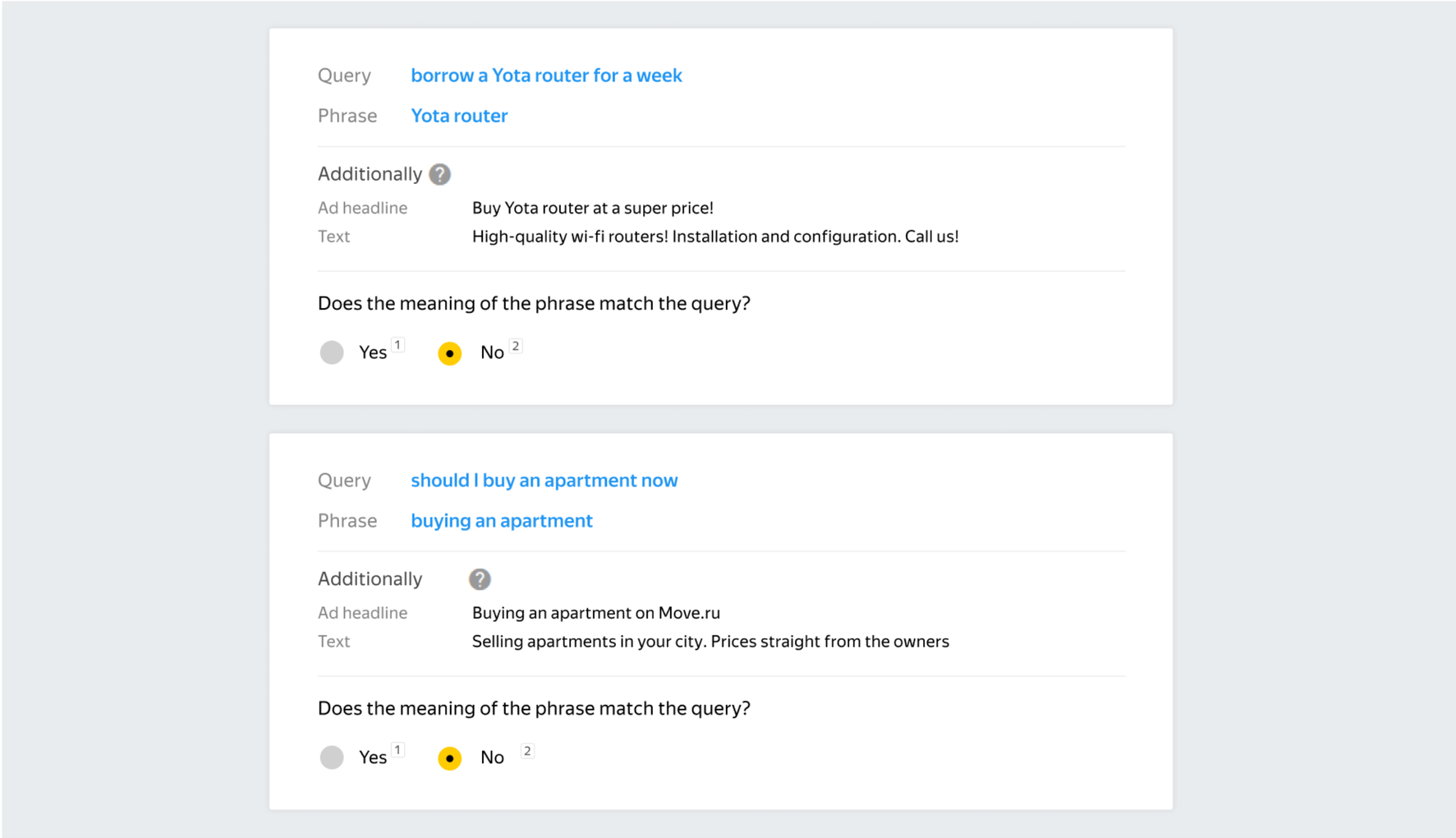
Rule #8. Constructing task suit

Page with many tasks

Check list:

- Absence of empty spaces
- Equal width of the task blocks
- No more than 2 (3) tasks in a row

Rule #8. Constructing task suit



Rule #9. Limit the number of elements in your interface

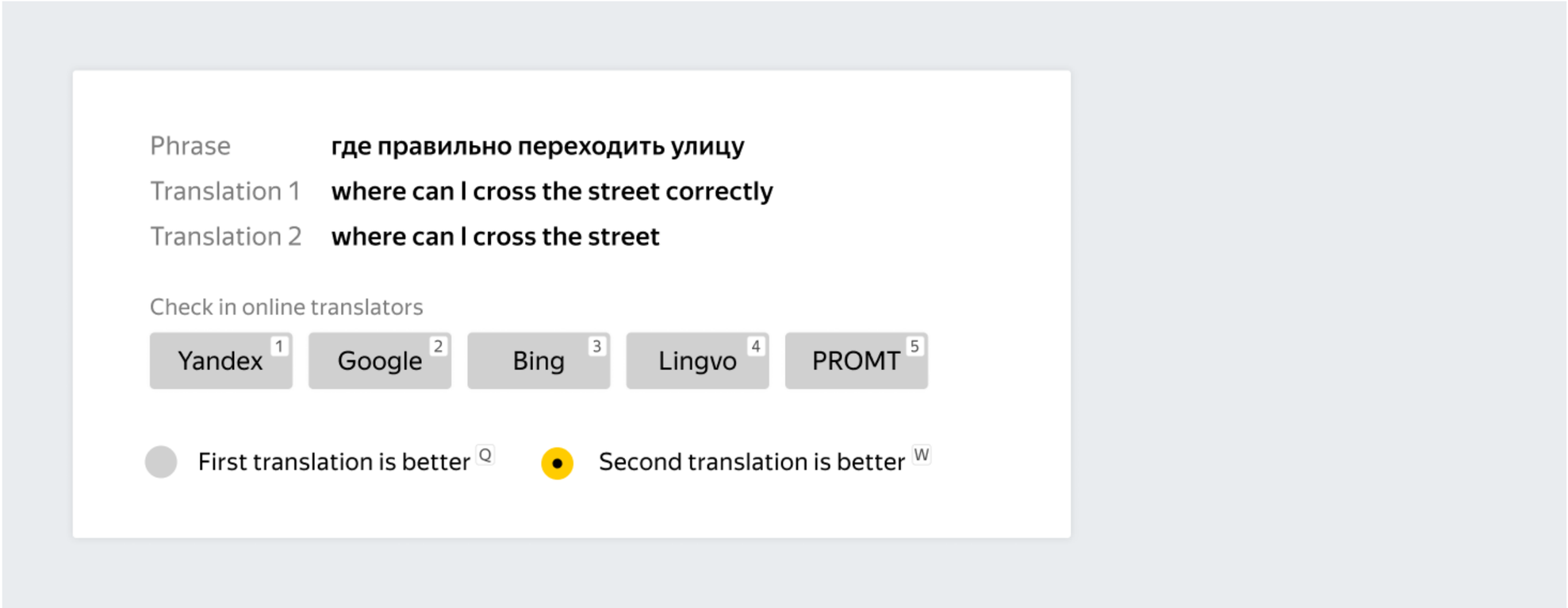
- Buttons
- Links
- Images
- Other elements, that with a particular function

The presence of any interface element must be justified

Every element of the interface should be useful for the performer

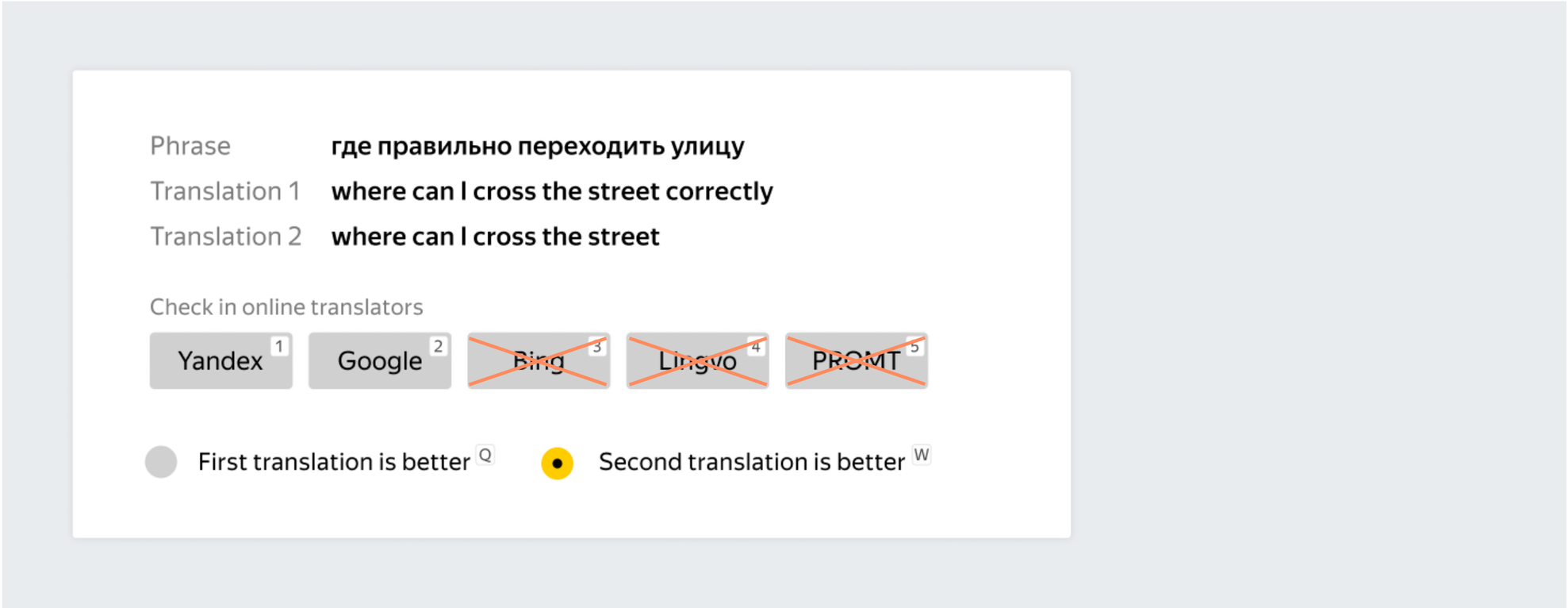
Rule #9. Limit the number of elements in your interface

Task: evaluate which translation from Russian to English is better



Rule #9. Limit the number of elements in your interface

Task: evaluate which translation from Russian to English is better



Bonus! Check list

1. Check the adaptability of the task template
2. Test task submission in the preview mode
3. Check the availability and functionality of hotkeys
4. Make sure that the required actions are checked
5. Check for the "not opening" option in tasks with external resources
6. Make sure that there are no experimental design solutions
7. Avoid page interface with a large number of tasks and different sizes of information in it
8. Make sure that there are no unnecessary interface elements in the task

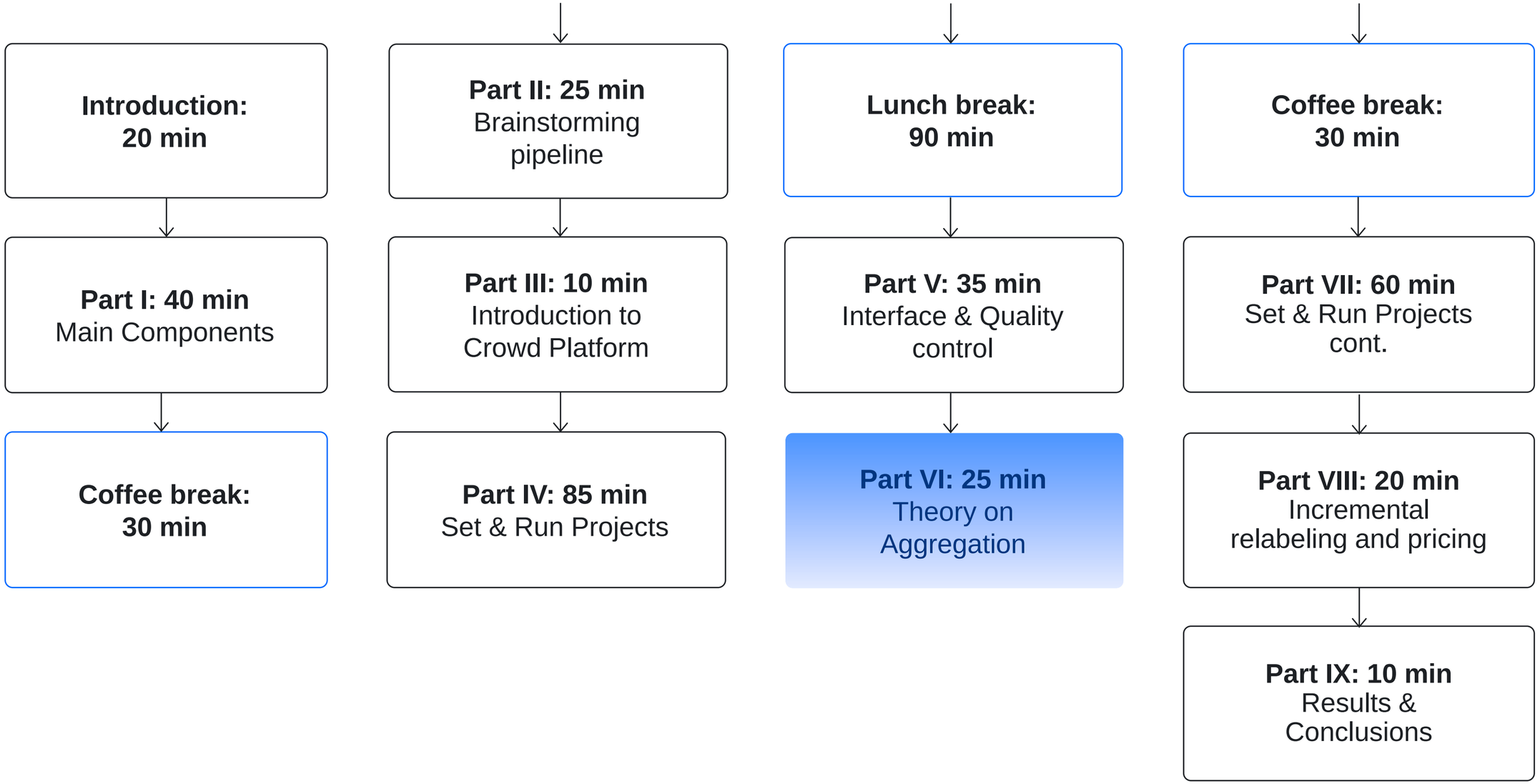
Part VI

Theory on Aggregation

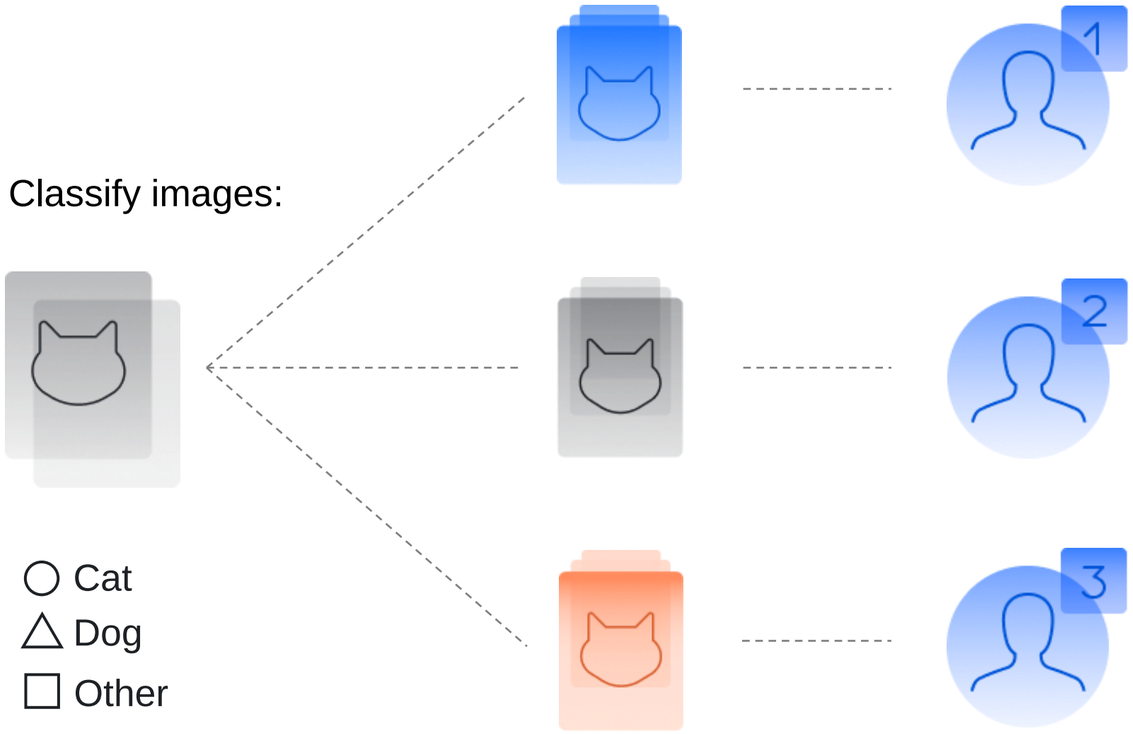
Valentina Fedorova,
Research analyst

Toloka

Tutorial schedule



Labeling data with crowdsourcing



- How to choose a reliable label?
- How many workers per object?
- How much to pay to workers?
- …

Evaluation of labeling approaches

vs

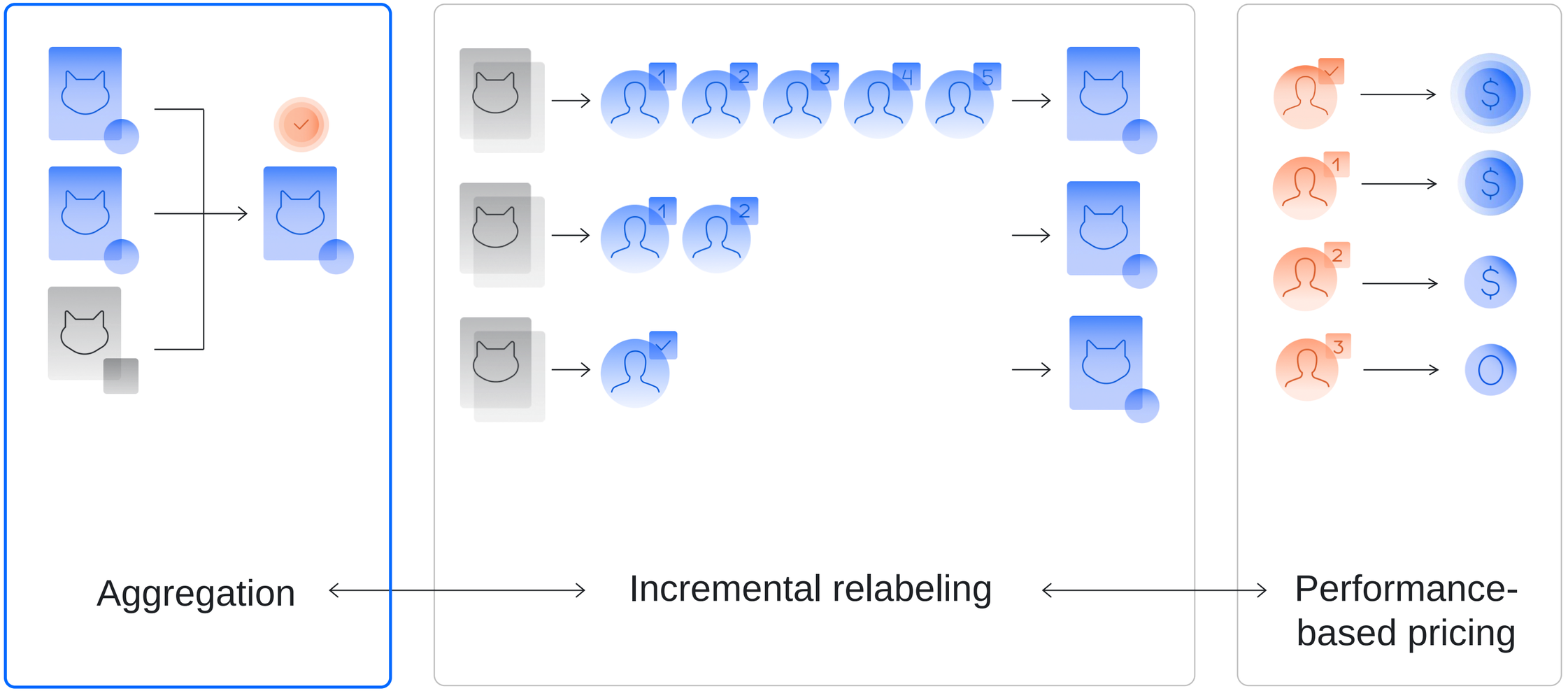
Accuracy

Cost

- Labels with a maximal level of accuracy for a **given budget**

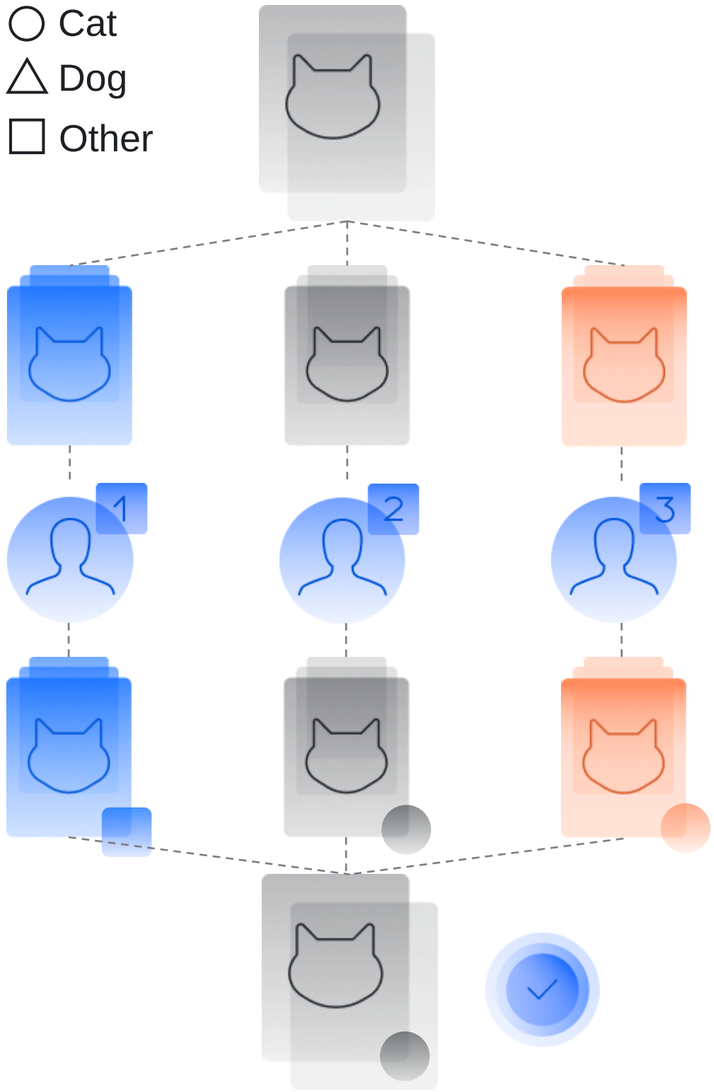
  or
- Labels of a chosen accuracy level for a **minimal budget**

Key components of labeling with crowds



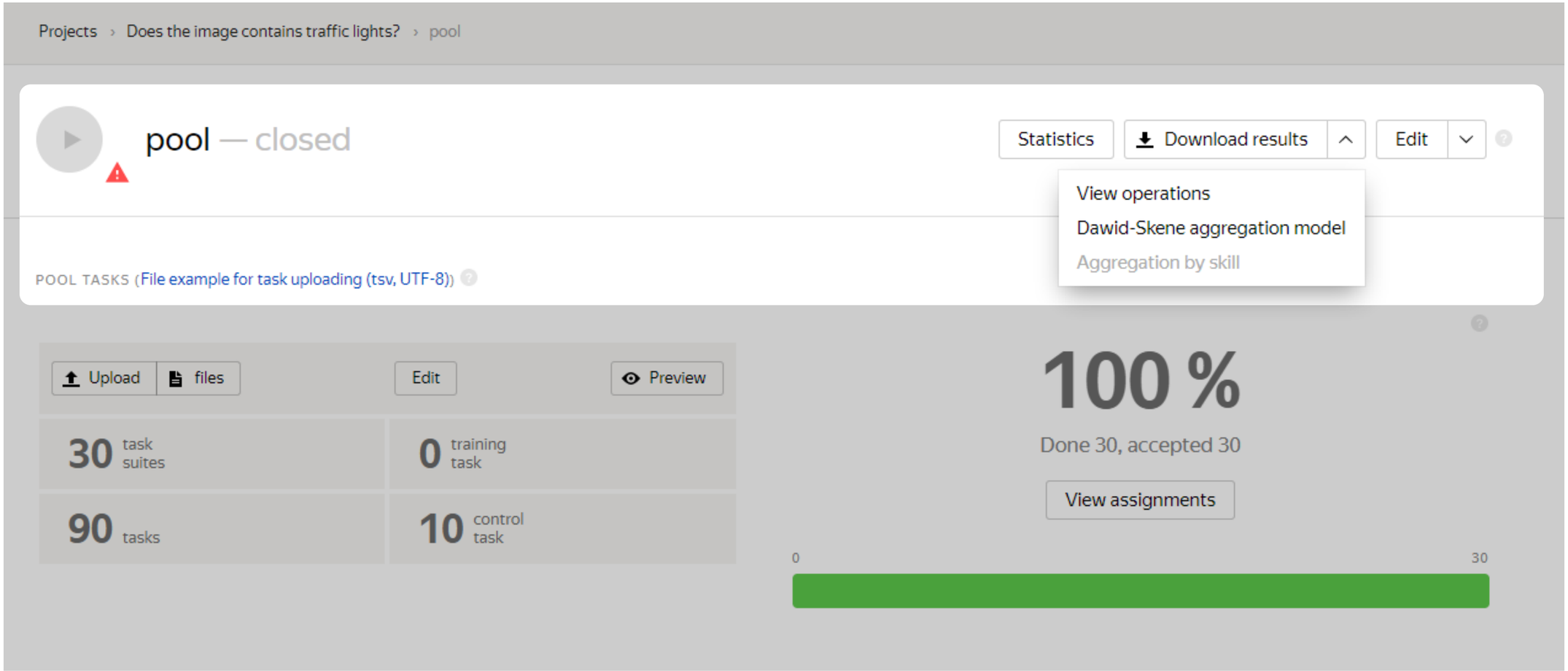
Aggregation

Labeling data with crowds



- Classify images
- Upload multiple copies of each object to label
- Workers assign noisy labels to objects
- Aggregate multiple labels for each object into a more reliable one

Process results



Multiclass labels

Project 1: Filter images

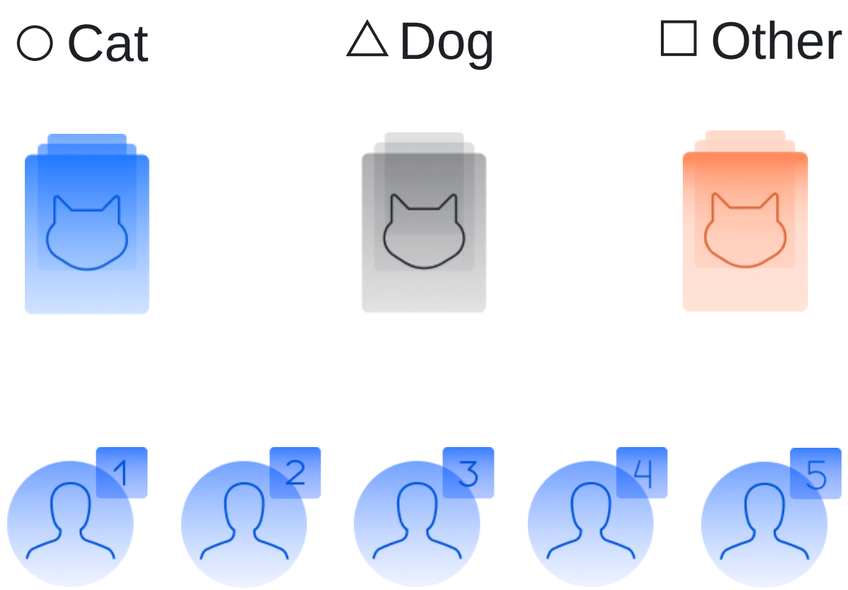
Are there shoes
in the picture?

Yes

No

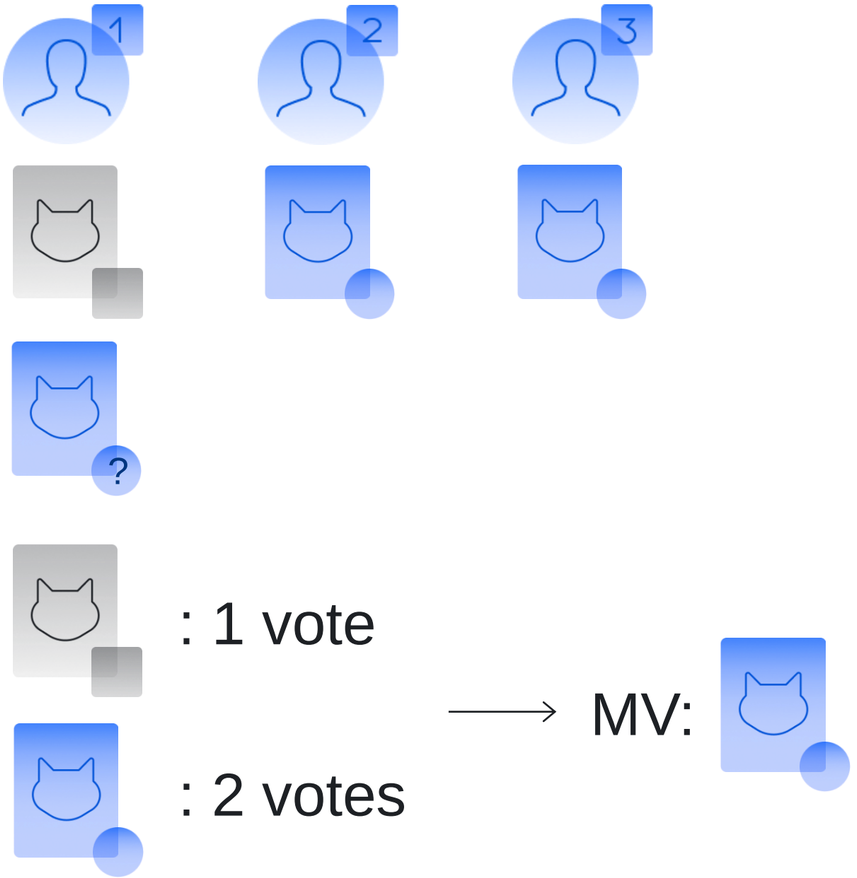
Notation

- Categories $k \in \{1,...,K\}$. E.g.:
- Objects $j \in \{1,...,J\}$. E.g.:
- Workers: $w \in \{1,...,W\}$. E.g.:
  - $W_j \subseteq \{1,...,W\}$ — workers labeled object j



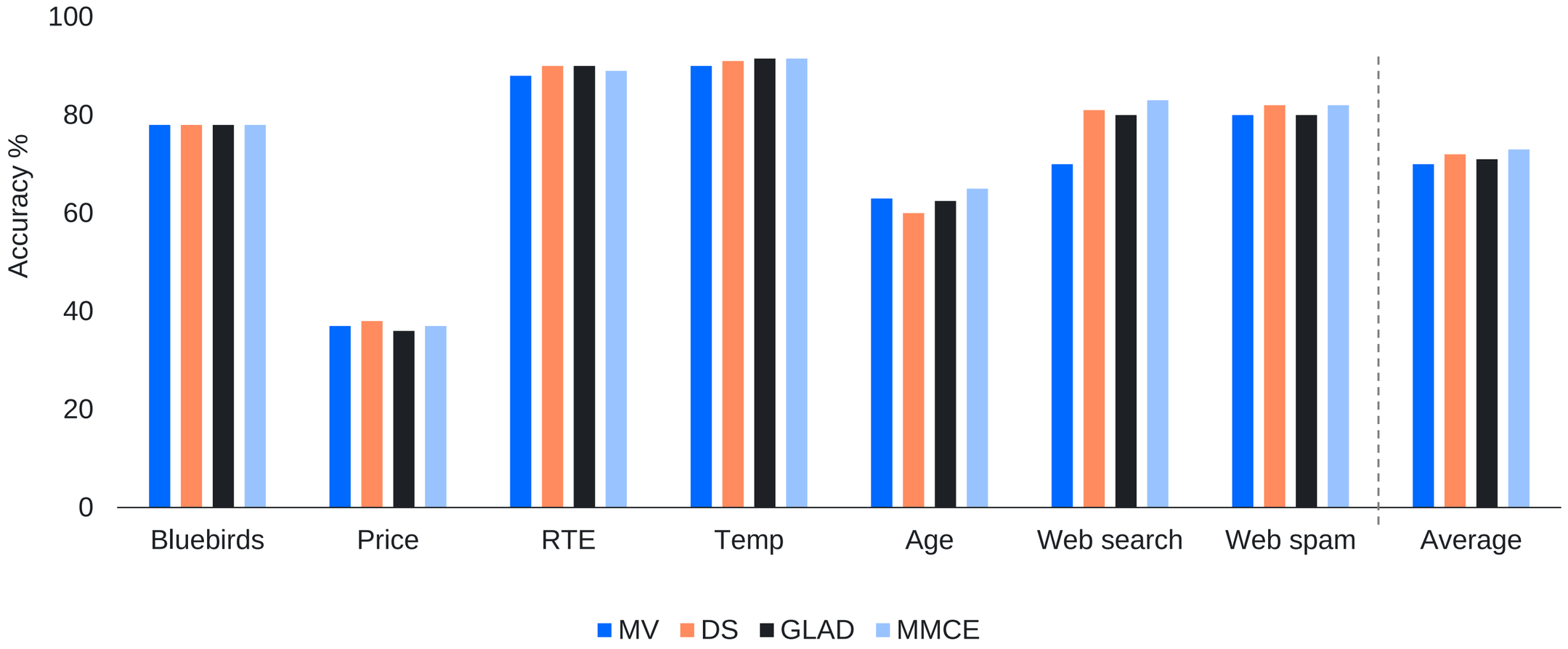
The simplest aggregation: Majority Vote (MV)

- The problem of aggregation:
  - Observe noisy labels
    $y = \{y_j^w | \, j = 1, \ldots, J \text{ and } w = 1, \ldots, W\}$
  - Recover true labels $z = \{z_j | \, j = 1, \ldots, J\}$
- A straightforward solution:



$\hat{z}_j^{MV} = \arg\max_{y=1,\ldots,K} \sum_{w \in W_j} \delta(y = y_j^w)$, where $\delta(A) = 1$ if A is true and 0 otherwise

Performance of MV vs other methods

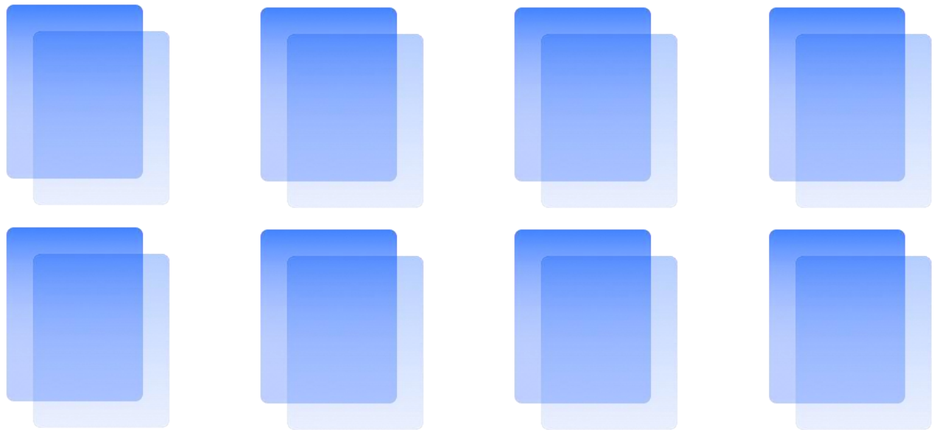


Zhou D. et al. Regularized minimax conditional entropy for crowdsourcing. 2015

Properties of MV

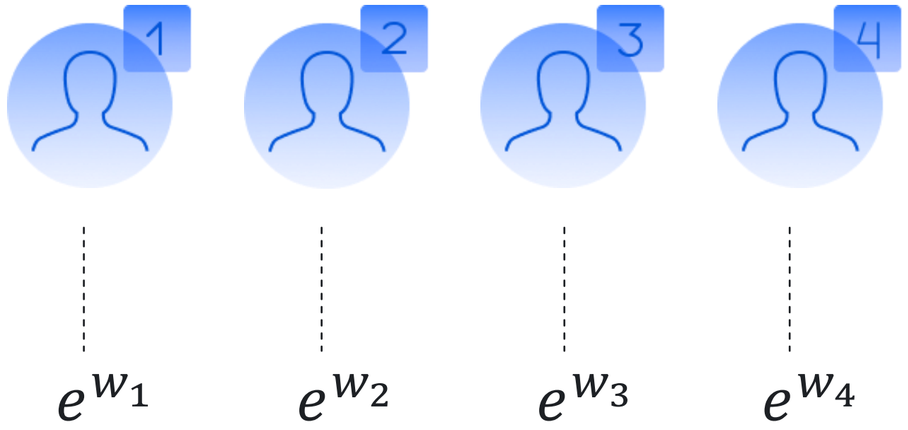
All workers are treated similarly

All objects are treated similarly

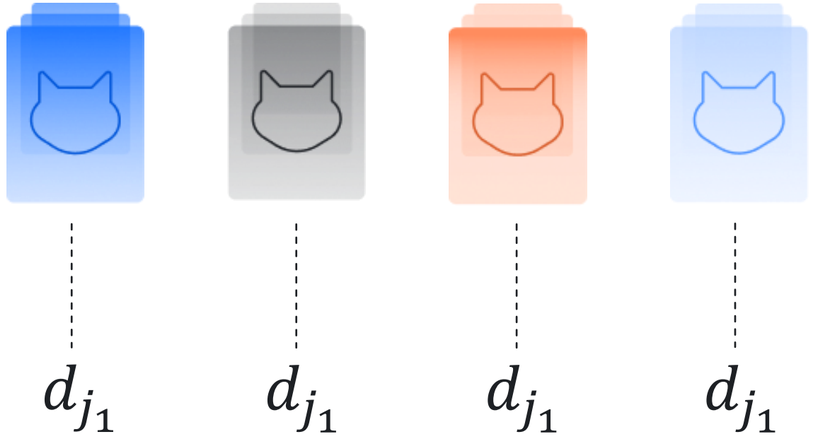


Advanced aggregation: workers and objects

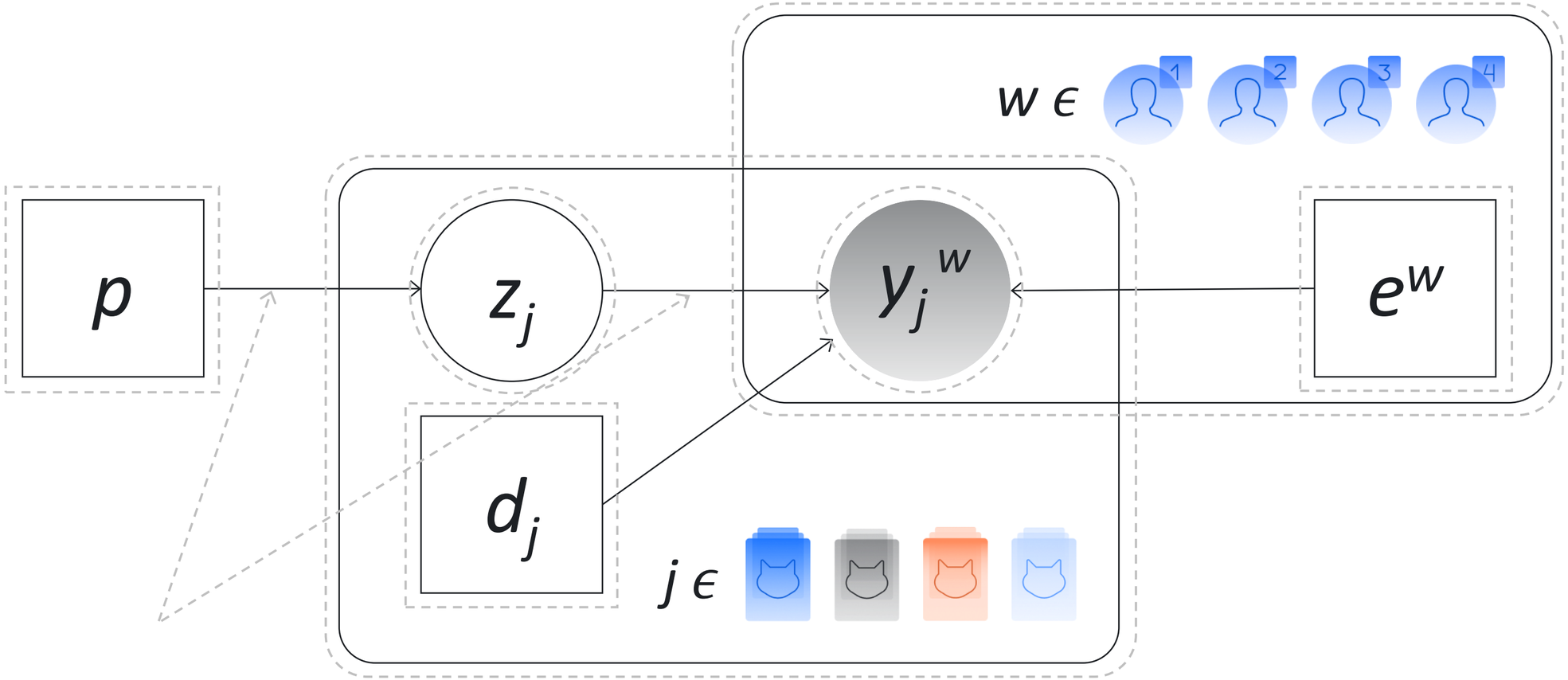
Parameterize expertise of workers by e^w



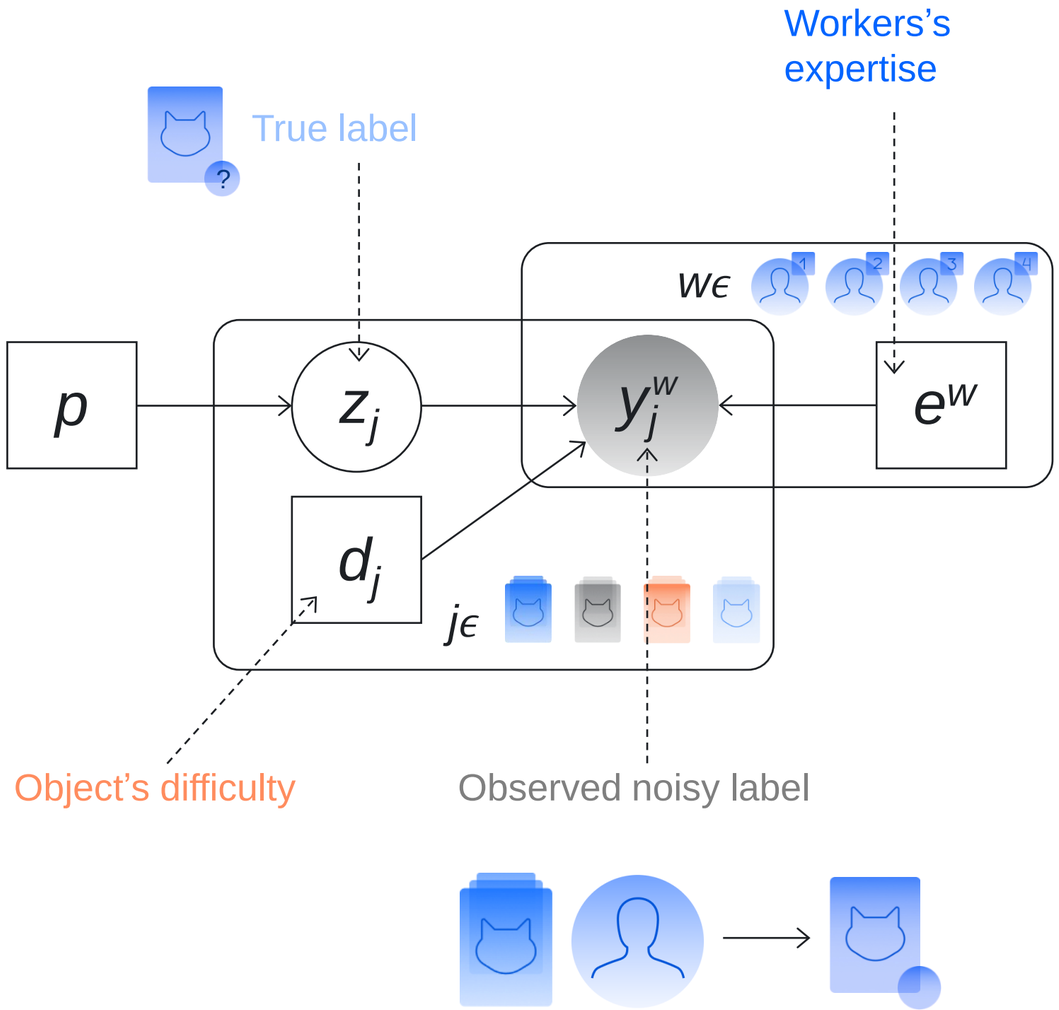
Parameterize difficulty of objects by d_j



Advanced aggregation: latent label models



Latent label models: noisy label model



A noisy label model $\mathrm{M}_\mathrm{j}^\mathrm{w} = \mathrm{M}\left(\mathrm{e}^\mathrm{w}, \mathrm{d}_\mathrm{j}\right)$ is a matrix of size $\mathrm{K} \times \mathrm{K}$ with elements

$$\mathrm{M}_\mathrm{j}^\mathrm{w}[\mathrm{c}, \mathrm{k}] = \Pr\left(\mathrm{Y}_\mathrm{j}^\mathrm{w} = \mathrm{k} \,|\, \mathrm{Z}_\mathrm{j} = \mathrm{c}\right)$$

j:

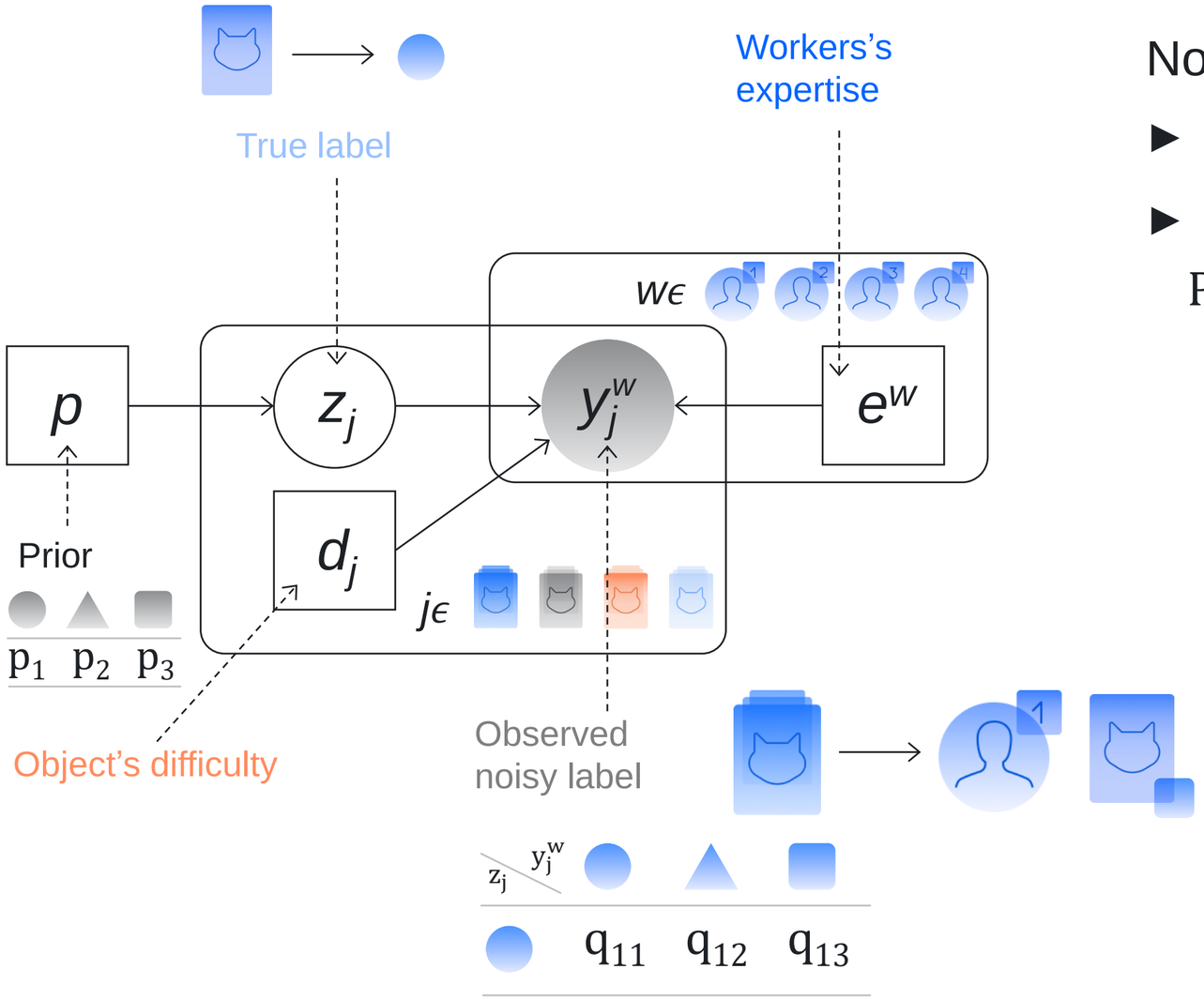
w:

$\mathrm{M}_\mathrm{j}^\mathrm{w}$:

Noisy / True	●	▲	■
●	q_{11}	q_{12}	q_{13}
▲	q_{21}	q_{22}	q_{23}
■	q_{31}	q_{32}	q_{33}

$\mathrm{q}_{c1} + \mathrm{q}_{c2} + \mathrm{q}_{c3} = 1$ for each c

Latent label models: generative process



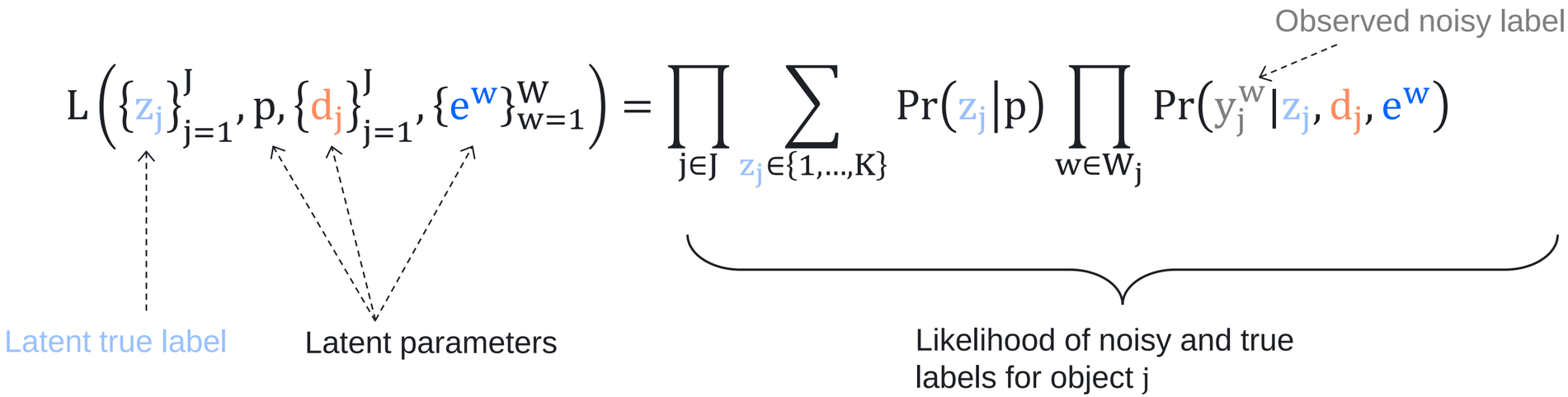
Noisy labels generation:

- Sample z_j from a distribution $P_Z(p)$
- Sample y_j^w from a distribution $P_Y(M_j^w[z_j,\cdot])$

In multiclassification, a standard choice for $P_Z(\cdot)$ and $P_Y(\cdot)$ is a Multinomial distribution Mult(·)

Latent label models: parameters optimization

- Assumption: y_j^w is cond. independent of everything else given z_j, d_j, e^w
- The likelihood of y and z under the latent label model:



- Estimate parameters and true labels by maximizing $L(...)$

Latent label models: EM algorithm

- Maximization of the expectation of log-likelihood (LL)*

$$\mathbb{E}_z \log \Pr(y, z) = \sum_{j \in J} \sum_{z_j \in \{1,\ldots,K\}} \Pr(z_j | p) \log \prod_{w \in W_j} \Pr(z_j | p) \Pr(y_j^w | z_j, d_j, e^w)$$

- **E-step:** Use Bayes' theorem for posterior distribution of $\hat{z}$ given p, d, e:

$$\hat{z}_j[c] = \Pr(Z_j = c | y, p, d, e) \propto \Pr(Z_j = c | p) \prod_{w \in W_j} \Pr(y_j^w | Z_j = c, d_j, e^w)$$

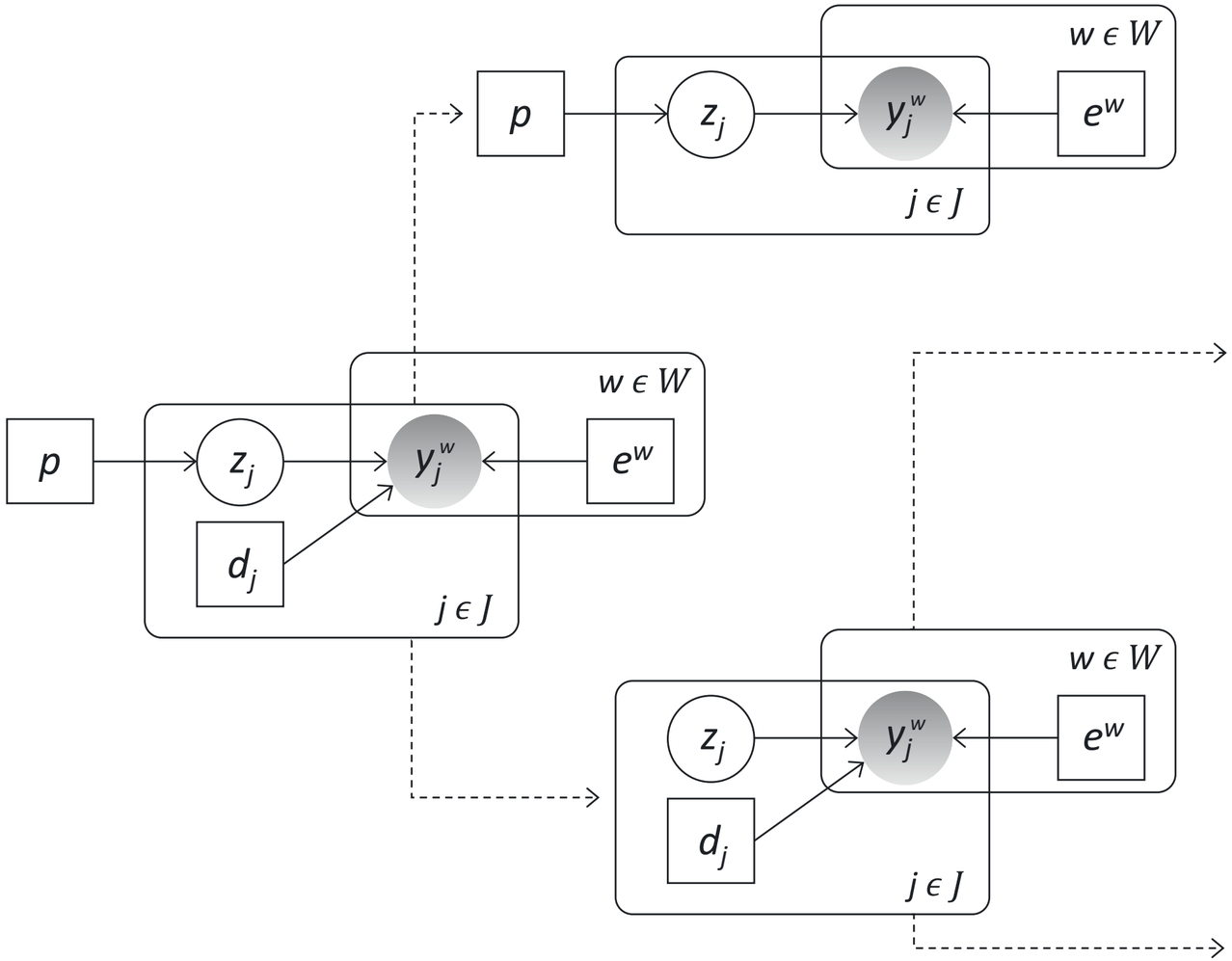
- **M-step:** Maximize the expectation of LL with respect to the posterior distribution of $\hat{z}$:

$$(p, d, e) = \operatorname{argmax} \mathbb{E}_{\hat{z}} \log \Pr(z_j | p) \prod_{w \in W_j} \Pr(y_j^w | z_j, d_j, e^w)$$

  - Analytical solutions
  - Gradient descent

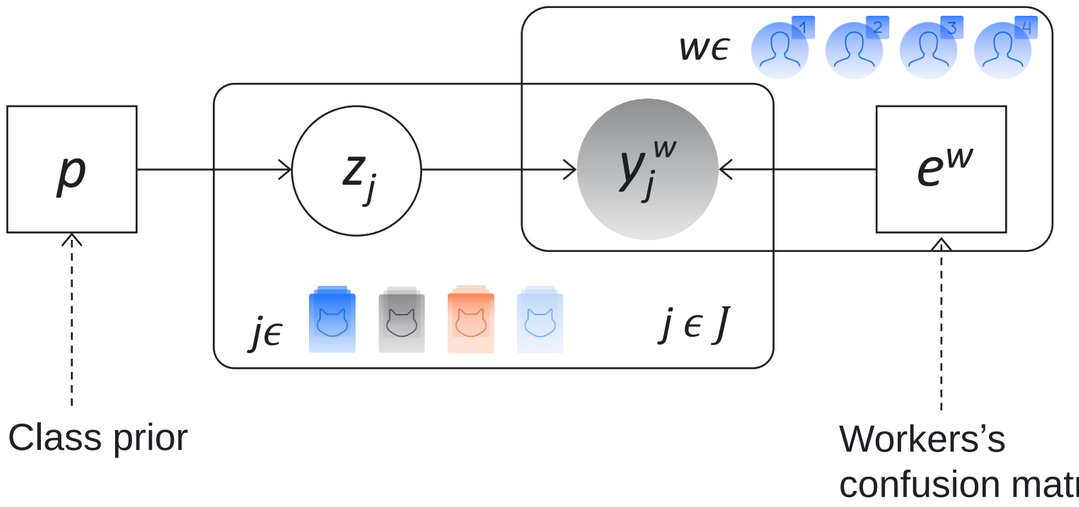
* it is a lower bound on LL of y and z

Latent label model (LLM): special cases



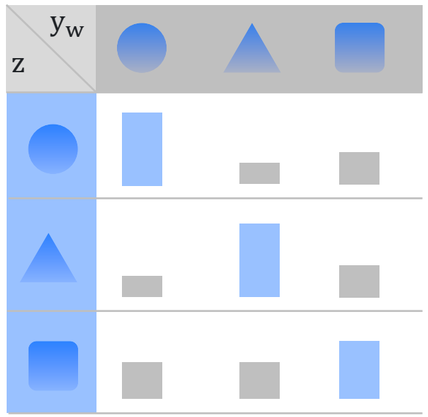
- Dawid and Skene model (DS):
  - Categories are different
  - Objects are **similar**
  - Workers are different
- Generative model of labels, abilities, and difficulties (GLAD):
  - Categories are **similar**
  - Objects are different
  - Workers are different
- Minimax conditional entropy model (MMCE):
  - Categories are different
  - Objects are different
  - Workers are different

Dawid and Skene model (DS)



LLM with parameters:

- p — vector of length K: $\mathrm{p[i]} = \Pr(\mathrm{Z} = \mathrm{c})$
- $\mathrm{e^w}$ — matrix of size $\mathrm{K} \times \mathrm{K}$: $\mathrm{e^w[c,k]} = \Pr(\mathrm{Y^w} = \mathrm{k} | \mathrm{Z} = \mathrm{c})$



DS: parameters optimization

- **E-step:**

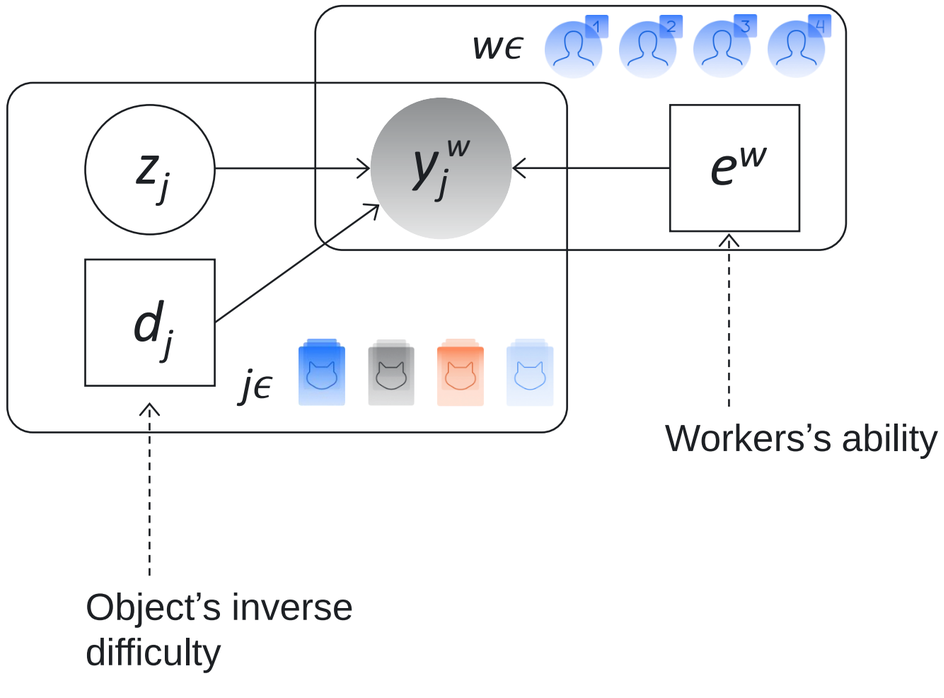
$$\hat{z}_j[c] = \frac{p[c] \prod_{w \in W_j} e^w\left[c, y_j^w\right]}{\sum_k p[k] \prod_{w \in W_j} e^w\left[k, y_j^w\right]}, \qquad c = 1, \ldots, K$$

- **M-step:** Analytical solution

$$e^w[c, k] = \frac{\sum_{j \in J} \hat{z}_j[c] \delta\left(y_j^w = k\right)}{\sum_{q=1}^{K} \sum_{j \in J} \hat{z}_j[c] \delta\left(y_j^w = q\right)}, \qquad k, c = 1, \ldots, K$$

$$p[c] = \frac{\sum_{j \in J} \hat{z}_j[c]}{J}, \qquad c = 1, \ldots, K$$

Generative model of Labels, Abilities, and Difficulties (GLAD)



LLM with parameters:

- Scalar $d_j \in (0, \infty)$
- Scalar $e^w \in (-\infty, \infty)$
- Model:

$$\Pr(Y_j^w = k | Z_j = c) = \begin{cases} a(w,j), & c = k \\ \dfrac{1 - a(w,j)}{K - 1}, & c \neq k \end{cases}$$

$$\text{where } a(w,j) = \frac{1}{1 + \exp(-e^w d_j)}$$

Whitehill et al., Whose vote should count more: Optimal integration of labels from labelers of unknown expertise, 2009

GLAD: parameters optimization

- Let $a(w,j) = \frac{1}{1+\exp(-e^w d_j)}$ and $P(z_j)$ be a predefined prior (e.g., $P(z_j) = {}^1/_K$)

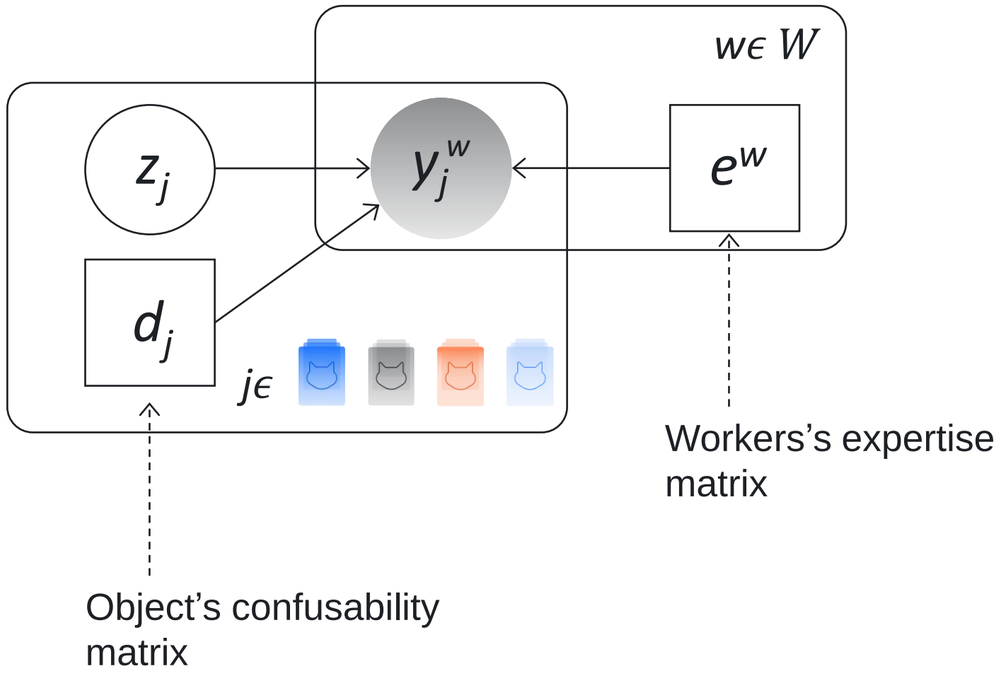
- **E-step:**

$$\hat{z_j}\,[c] \propto P(Z_j = c) \prod\nolimits_{w \in W_j} a(w,j)^{\delta(y_j^w = c)} \left(\frac{1-a(w,j)}{K-1}\right)^{\delta(y_j^w \neq c)}, \quad c = 1, \ldots, K$$

- **M-step:** estimate (d, e) for given $\hat{z}$ using gradient descent

$$(d^t, e^t) = \operatorname{argmax} \sum_{j \in J} \left[\mathbb{E}_{\hat{z_j}} \log P(z_j) + \sum_{w \in W_j} \mathbb{E}_{\hat{z_j}} \log \Pr(y_j^w | z_j) \right]$$

MiniMax Conditional Entropy model (MMCE)



- Find parameters that minimize the maximum conditional entropy of observed labels:

$$\min_Q \max_P - \sum_{\substack{j \in J \\ c \in \{1,\ldots,K\}}} Q(Z_j = c) \sum_{\substack{w \in W \\ k \in \{1,\ldots,K\}}} P(Y_j^w = k | Z_j = c) \log P(Y_j^w = k | Z_j = c)$$

- LLM with parameters:
  - d_j — matrix of size $K \times K$
  - e^w — matrix of size $K \times K$
  - Noisy label model:

$$\Pr(Y_j^w = k | Z_j = c) = \exp(d_j[c,k] + e^w[c,k])$$

Zhou et al., Learning from the Wisdom of Crowds by Minimax Entropy, 2012

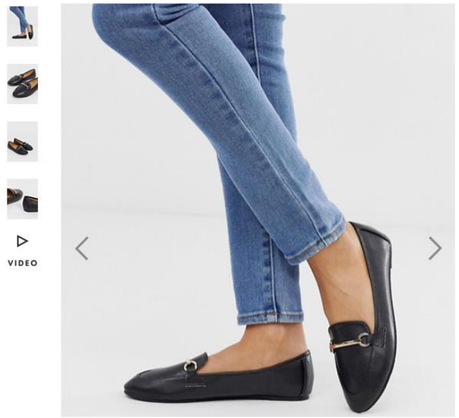
Summary of aggregation methods

	MV	DS	GLAD	MMCE
Categories (K)				
Objects (J)				
Workers (W)				
Number of parameters	0	$WK^2 + K$	$W + J$	$(W + J)K^2$

Pairwise comparisons

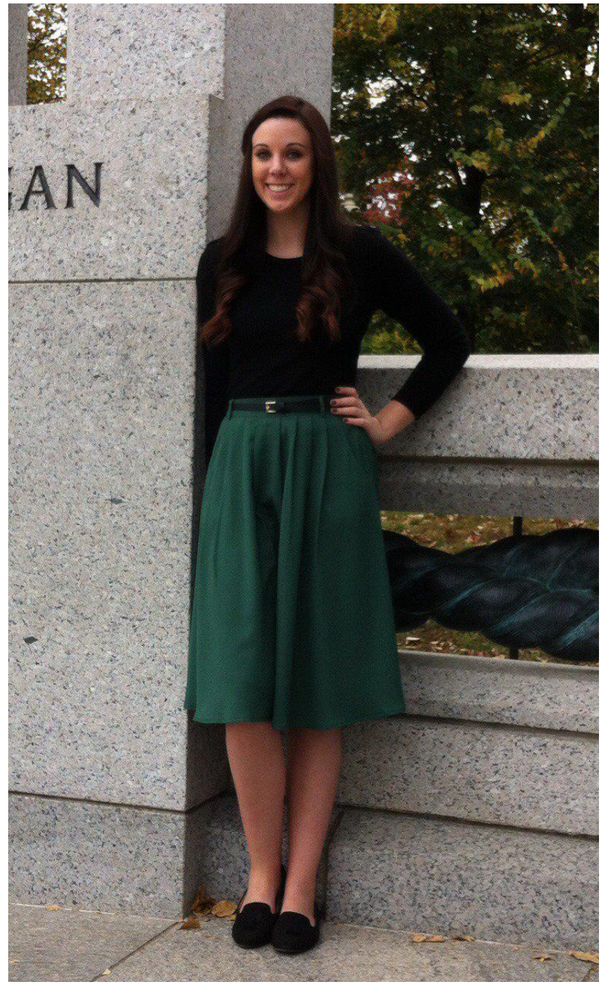
Project 4: Compare items

Which shoes look more similar to the one in the picture?



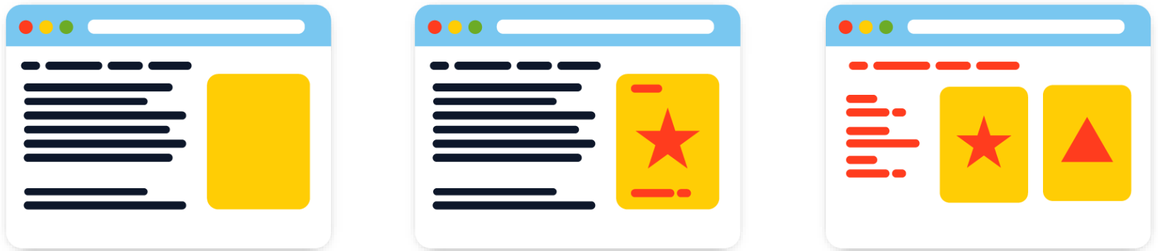
Left

Right

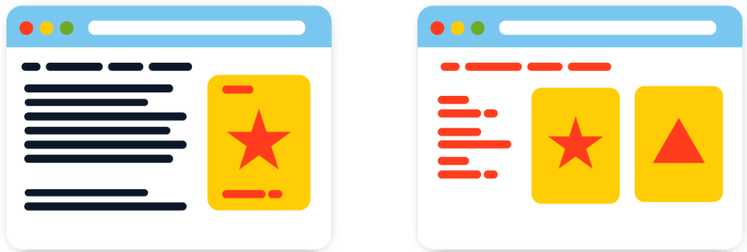


Notation

- Answers: Left or Right
- Items $d_j \in \{1, \ldots, N\}$ E.g.:



- Tasks:



Choose a better item:
Left
Right

- Workers $w \in \{1, \ldots, W\}$ E.g.:

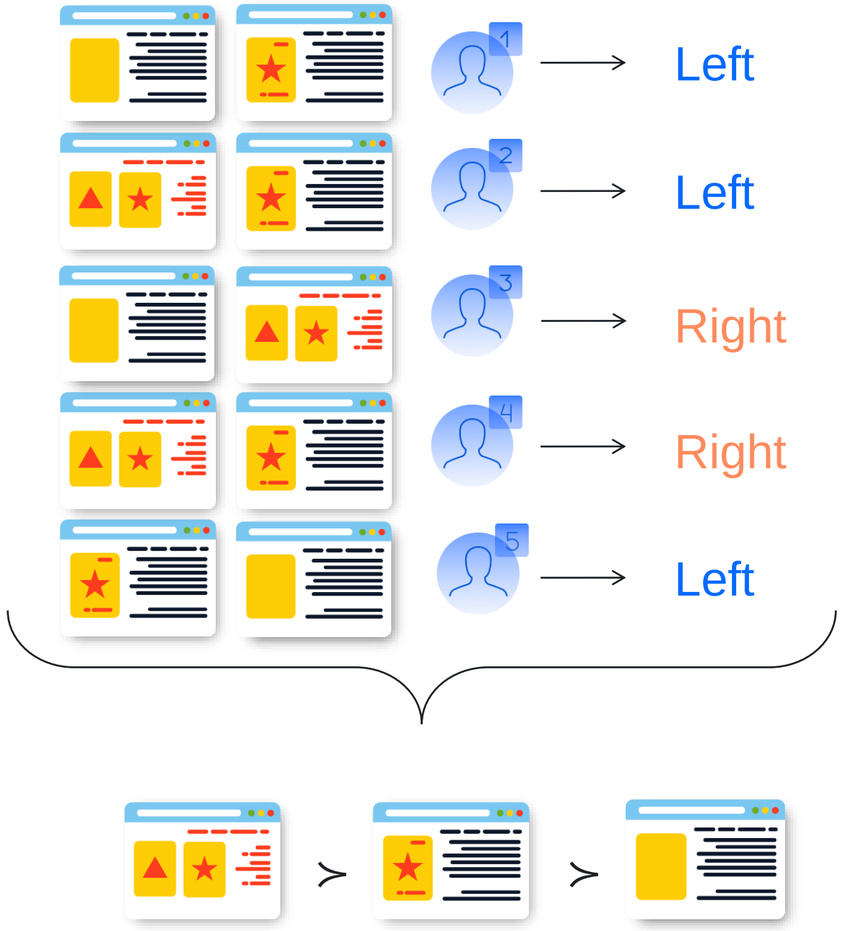
Formalization

Ranking from pairwise comparisons:

- Given pairwise comparisons for items in D:

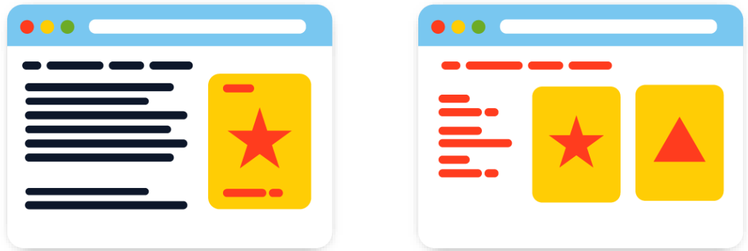
$$P = \{(w_k, d_i, d_j) : i \succ_k j\}$$

- Obtain **a ranking** π over items $D \rightarrow \{1, \ldots, N\}$ based on answers in P



Difference from multiclassification

- The latent label assumption is not satisfied when comparing complex items



- Different tasks may contain common items

Bradley and Terry model (BT)

- Assume that each item $d_i \in D$ has a latent “quality” score $s_i \in \mathbb{R}$

d_i  $\longleftrightarrow$ s_i

- The probability that $d_i \in D$ will be preferred in a comparison over $d_j \in D$

$$\Pr(i \succ j) = f(s_i - s_j), \text{ where } f(x) = {}^{1}/_{1+e^{-x}}.$$

- The model assumes that all workers are equally good and truthful

Bradley, R. A. and Terry, M. E. "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons". 1952

NoisyBT model: parameterization of workers

w_k  ⟷ "reliability" γ_k and "bias" q_k

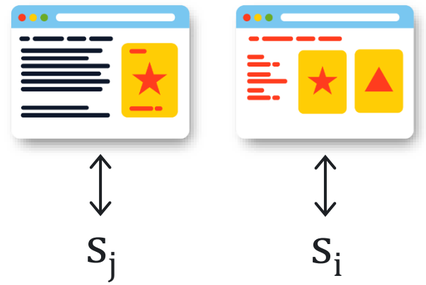
- The probability that w reads task is

$$\Pr(w_k \text{ reads a task}) = \mathrm{f}(y_k) \leftarrow \text{Logistic function}$$

- If w_k reads a task, she answers according to scores:

$$\underbrace{(\mathrm{f}(\mathrm{s_i} - \mathrm{s_j})}, \mathrm{f}(\mathrm{s_j} - \mathrm{s_i}))$$

Probability to choose Left if compares items



- If w_k does not read a task, she answers according to her bias

$$\underbrace{\mathrm{f}(q_k)}, \mathrm{f}(-q_k)$$

Probability to choose Left if answers randomly

NoisyBT: likelihood of workers' answers

The likelihood of $i \succ_k j$ is

$$\Pr(i \succ_k j) = \underbrace{f(\gamma_k) f(s_i - s_j)}_{\text{Truthful answer}} + \underbrace{\big(1 - f(\gamma_k)\big) f\big((-1)^{(1-\mathbb{I}(d_i \text{ was left}))} q_k\big)}_{\text{Random answer}},$$

where $\mathbb{I}(d_i \text{ was left})$ is the indicator for the order of d_i and d_j

$\mathbb{I}(d_i \text{ was left}) = 1$ $\quad$ $\mathbb{I}(d_i \text{ was left}) = 0$

NoisyBT: parameters optimization

Likelihood of observed comparisons:

$$T(s, q, \gamma) = \sum_{(w_k, d_i, d_j) \in P} \log \Pr(i \succ_k j) =$$

$$\sum_{(w_k, d_i, d_j) \in P} \log\left[f(\gamma_k) f(s_i - s_j) + \left(1 - f(\gamma_k)\right) f\left((-1)^{(1 - \mathbb{I}(d_i \text{ was left}))} q_k\right)\right]$$

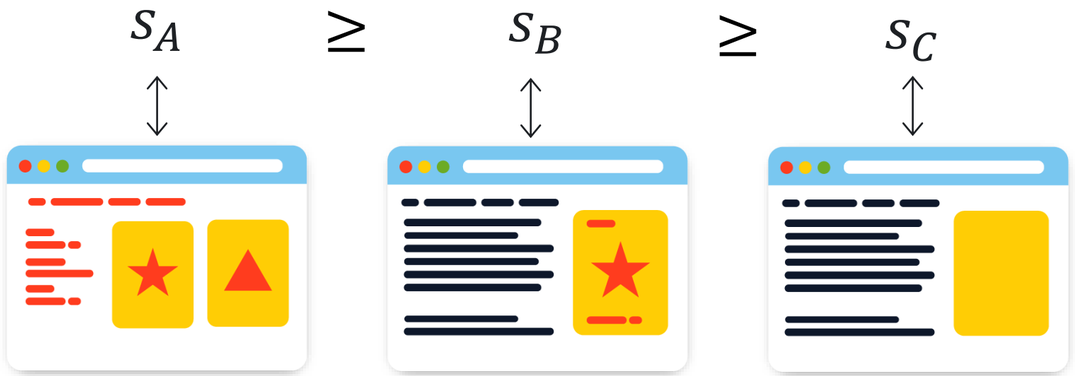
- $\{s_i\}_{i=1,\ldots,N}$ and $\{\gamma_k, q_k\}_{k=1,\ldots,W}$ are inferred by maximizing the log-likelihood:

$$T(s, q, \gamma) \to \max_{\{s_i, \gamma_k, q_k\}}$$

- To obtain a **ranking** π over items, **sort** items according to their **scores**

Summary about pairwise comparisons

- Latent scores models for ranking from pairwise comparisons:



- To reduce bias from unreliable answers parameterize workers

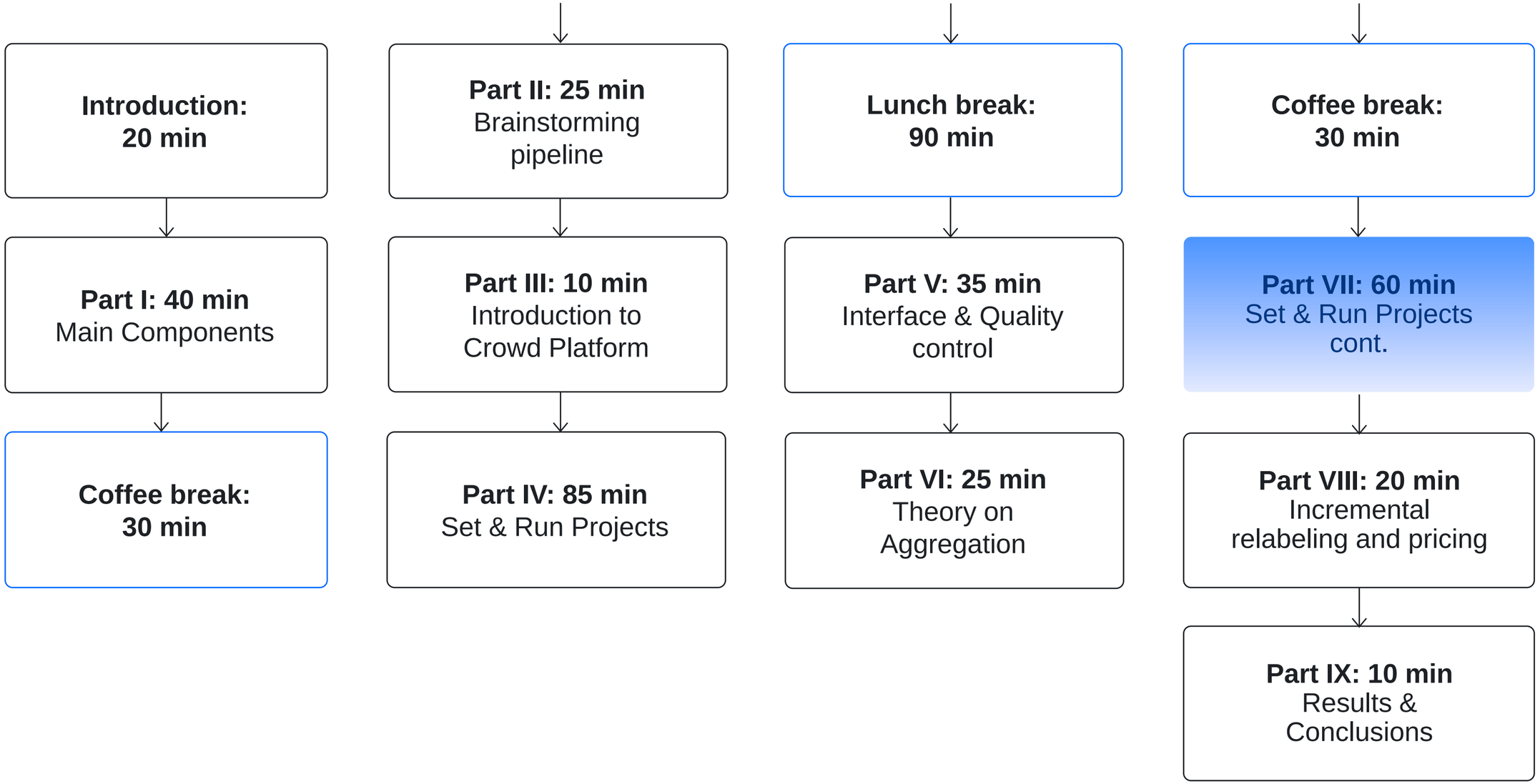
Part VII

Setting up and running label collection projects cont.

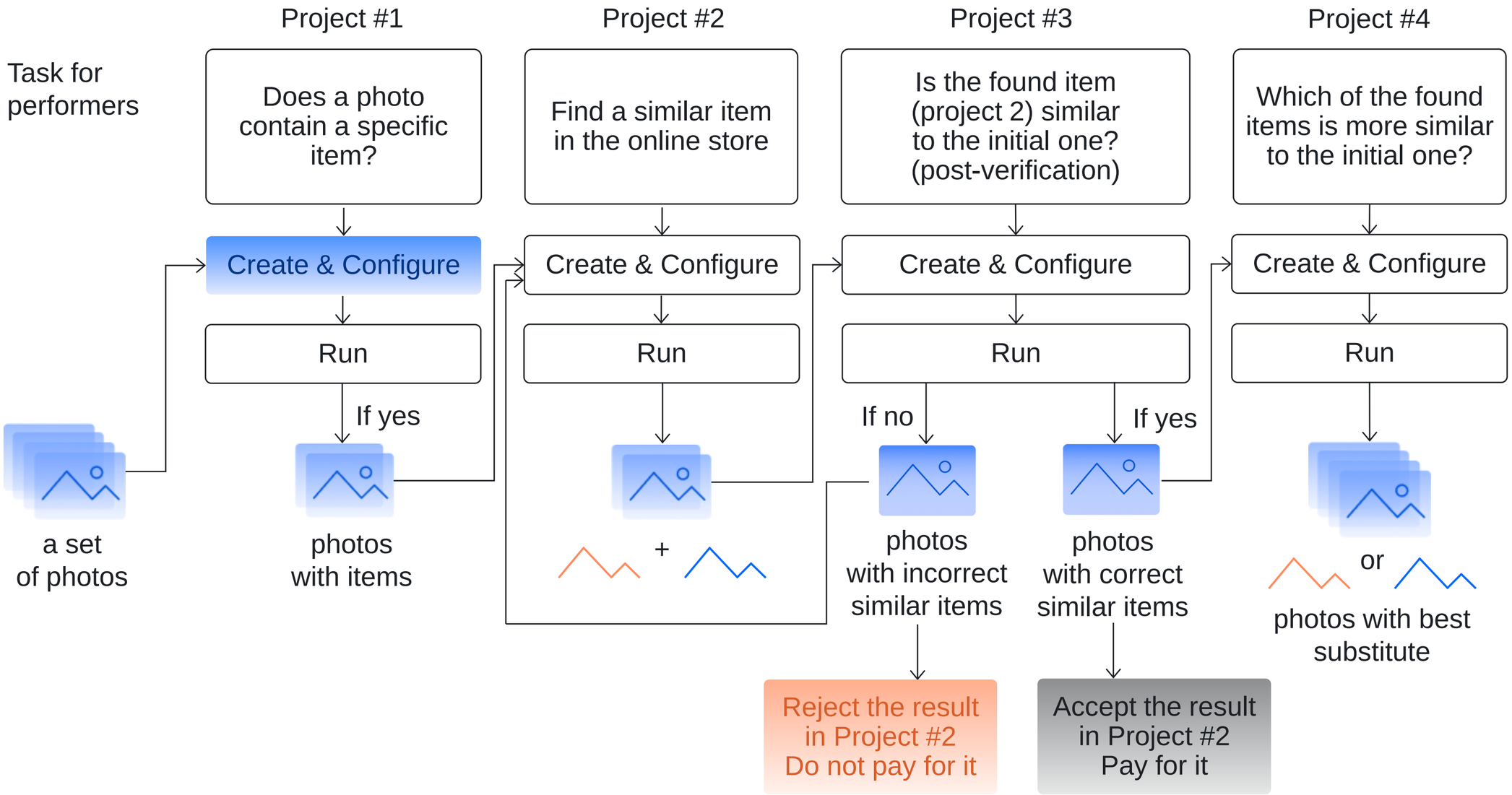
Daria Baidakova,
Project Manager

Toloka

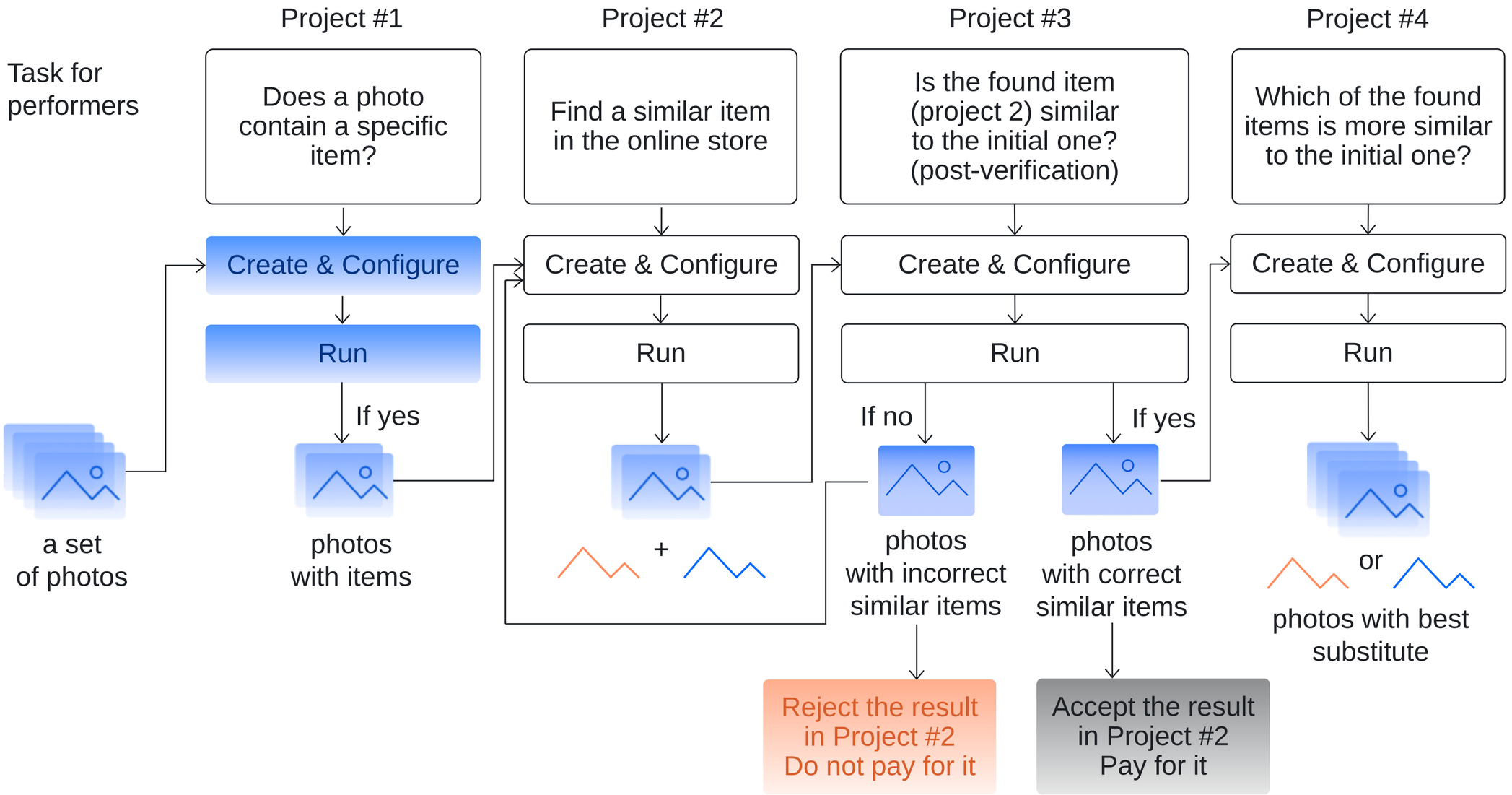
Tutorial outline



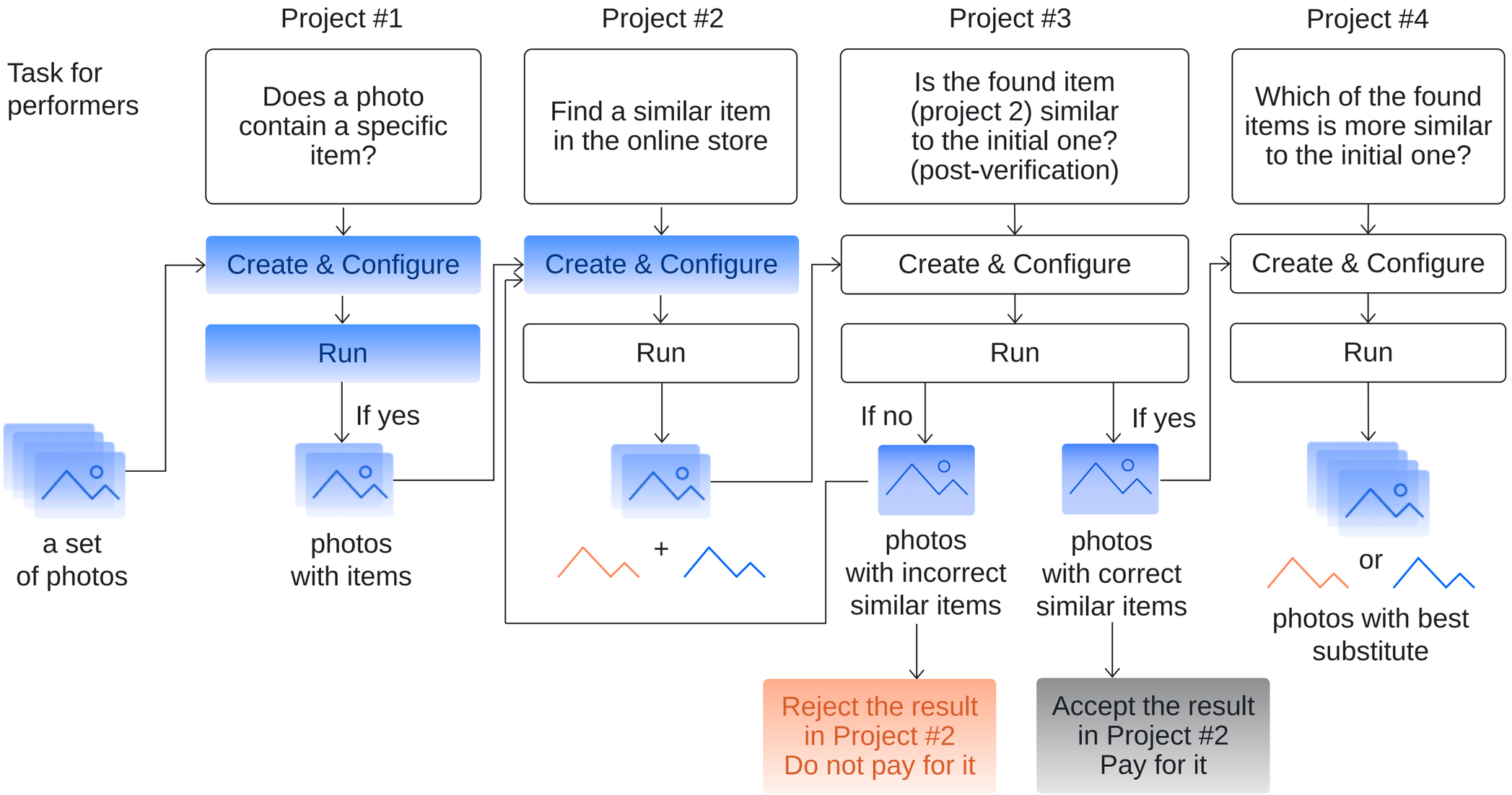
Most of us are at this step



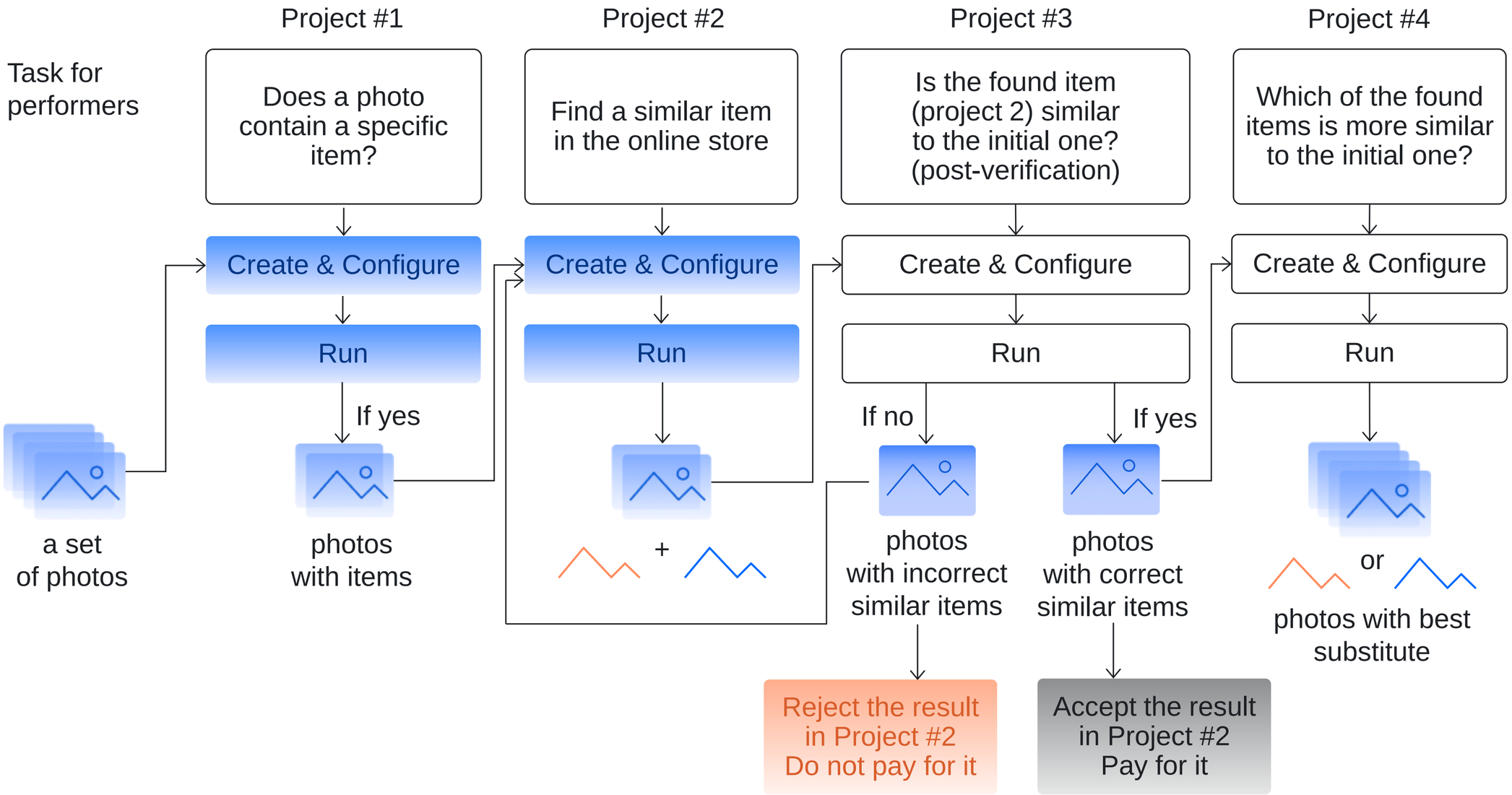
Most of us are at this step



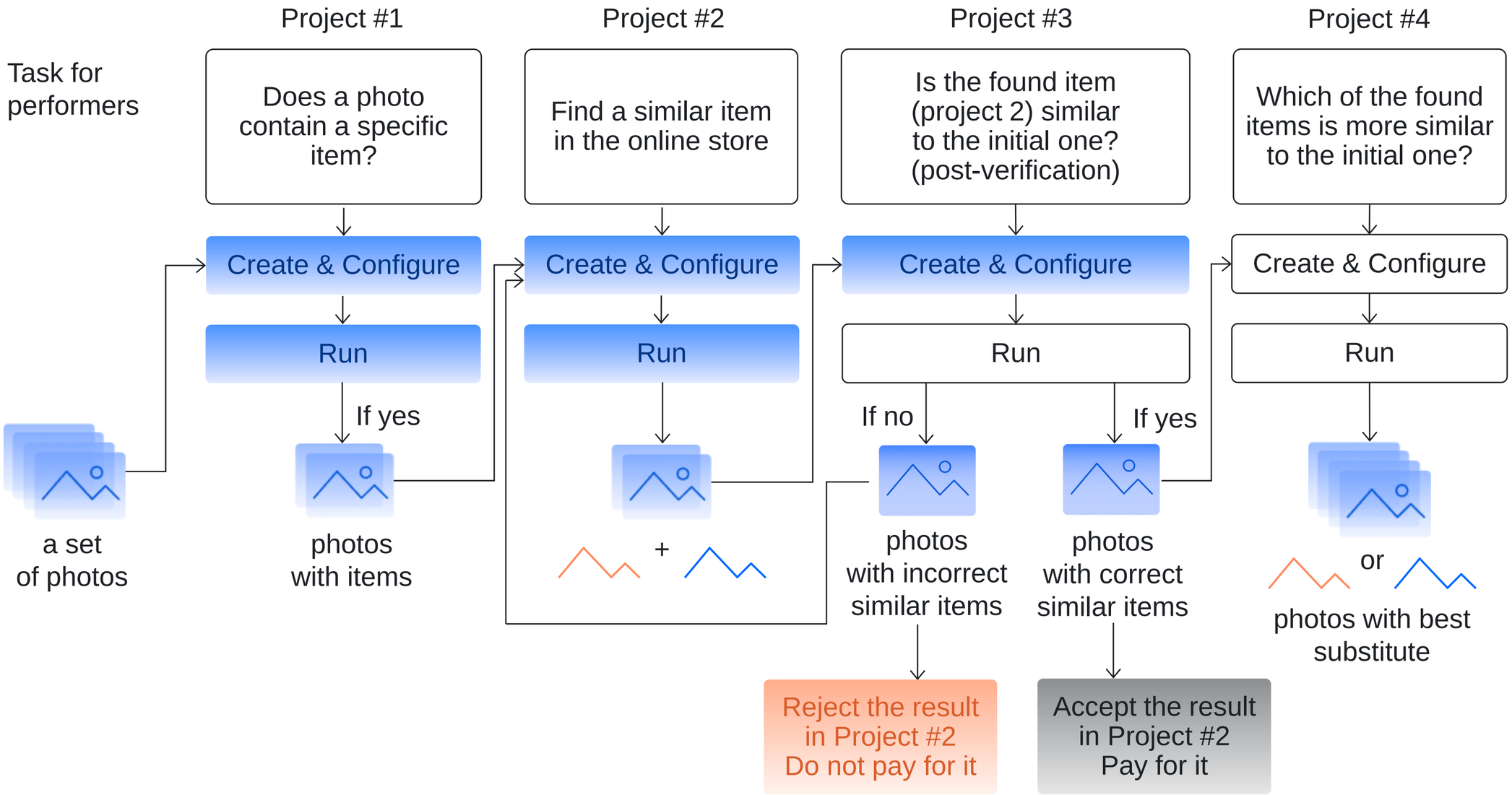
Most of us are at this step



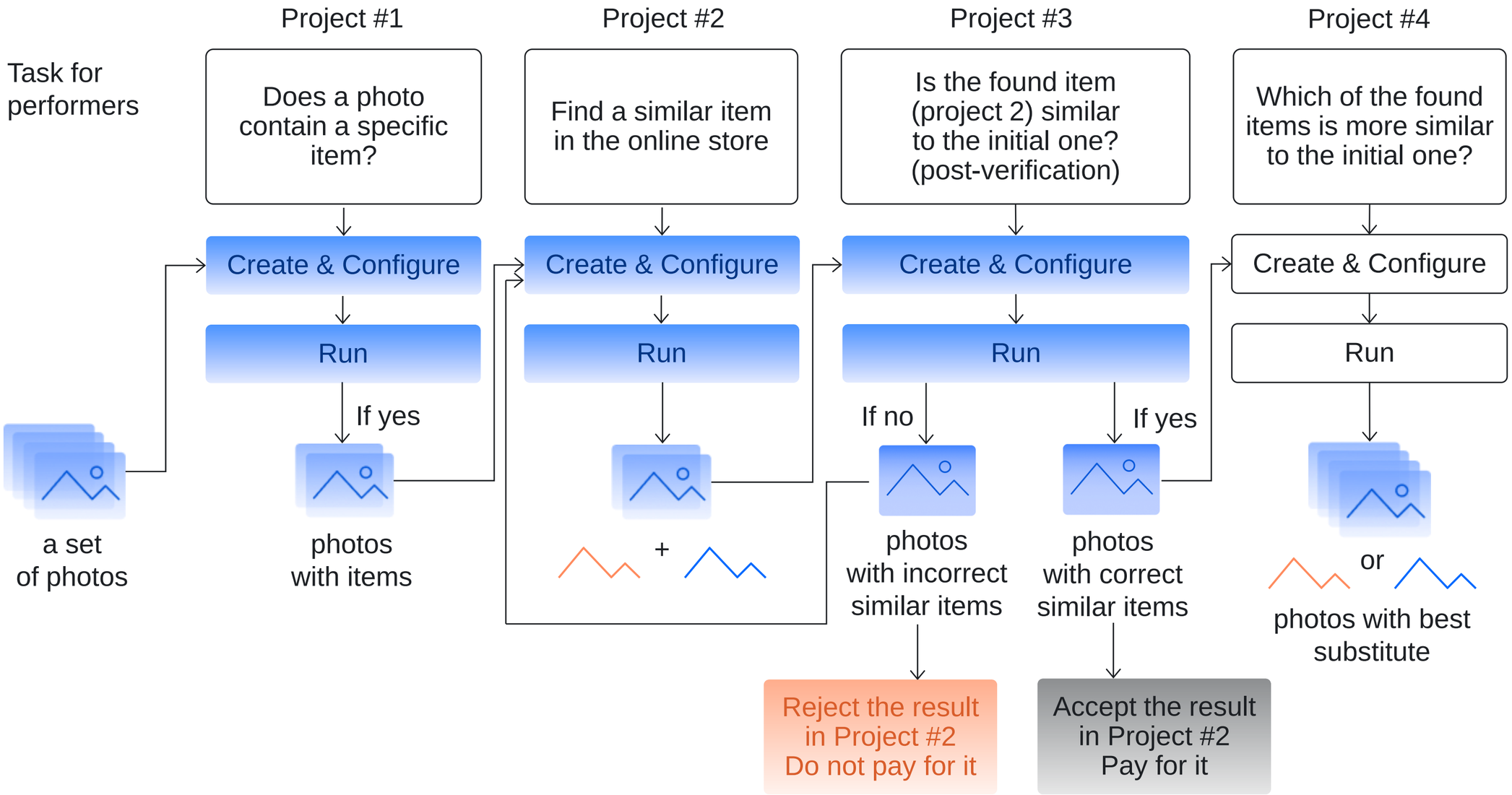
Most of us are at this step



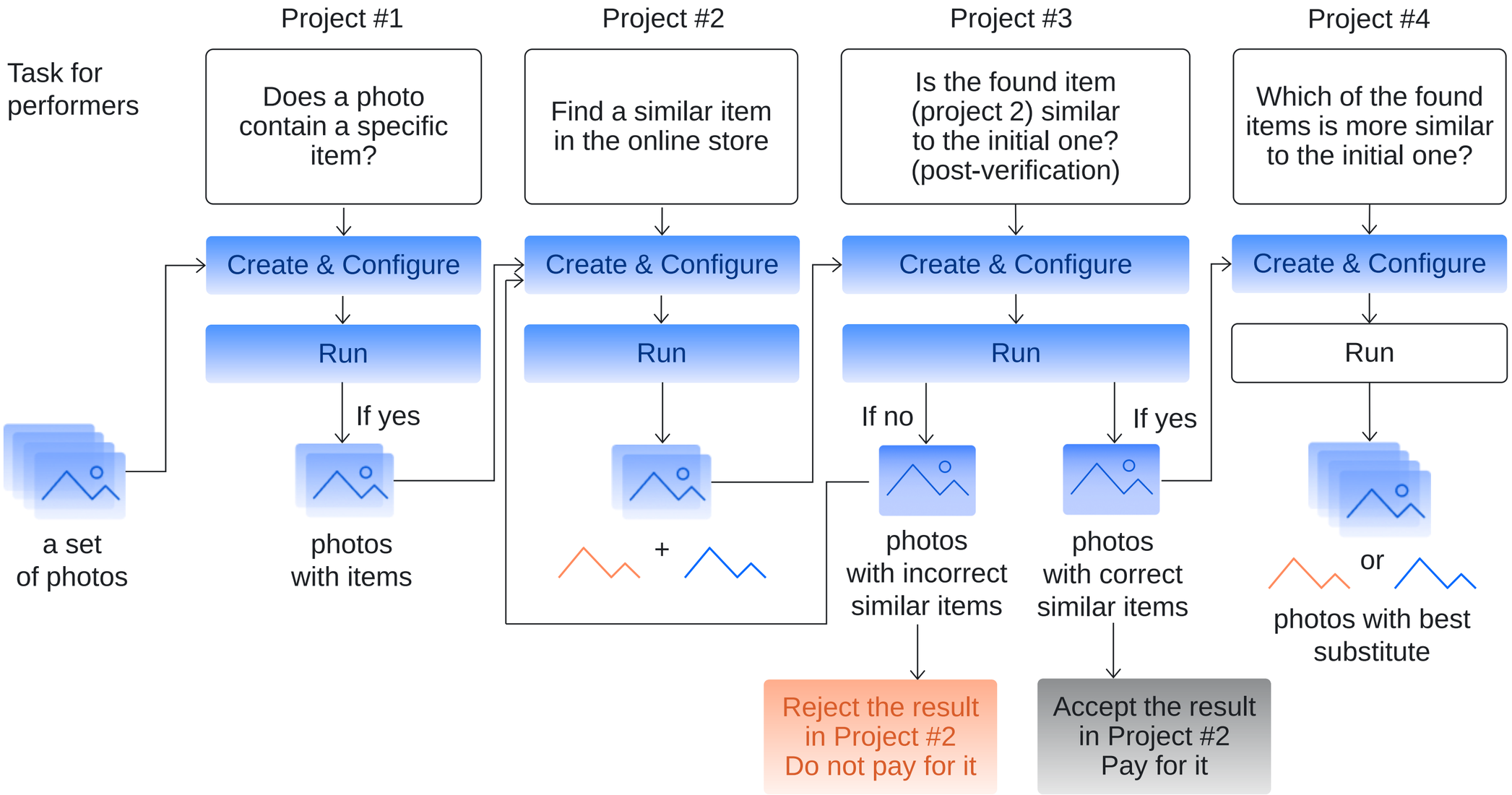
Most of us are at this step



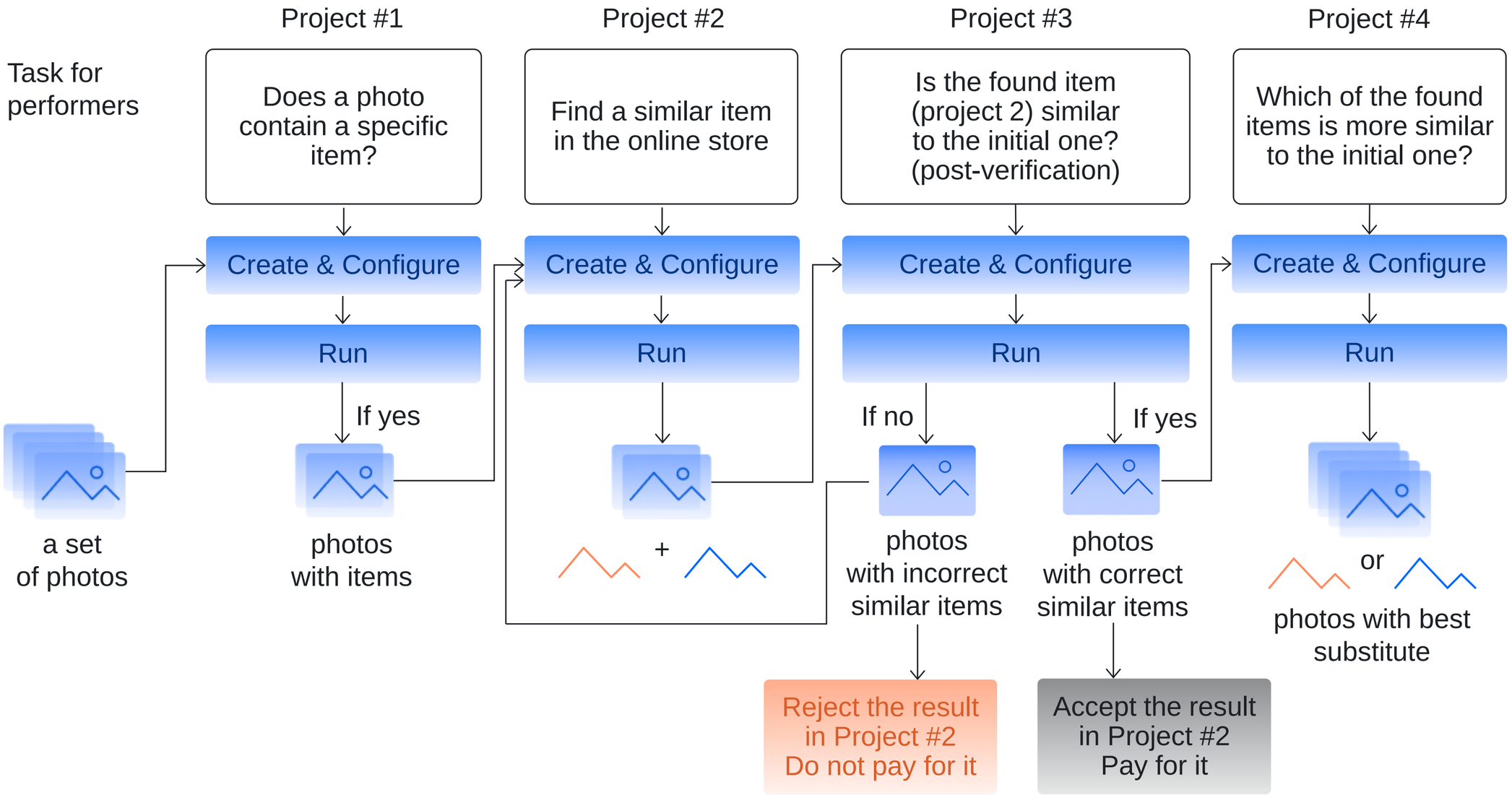
Most of us are at this step



Most of us are at this step



Most of us are at this step



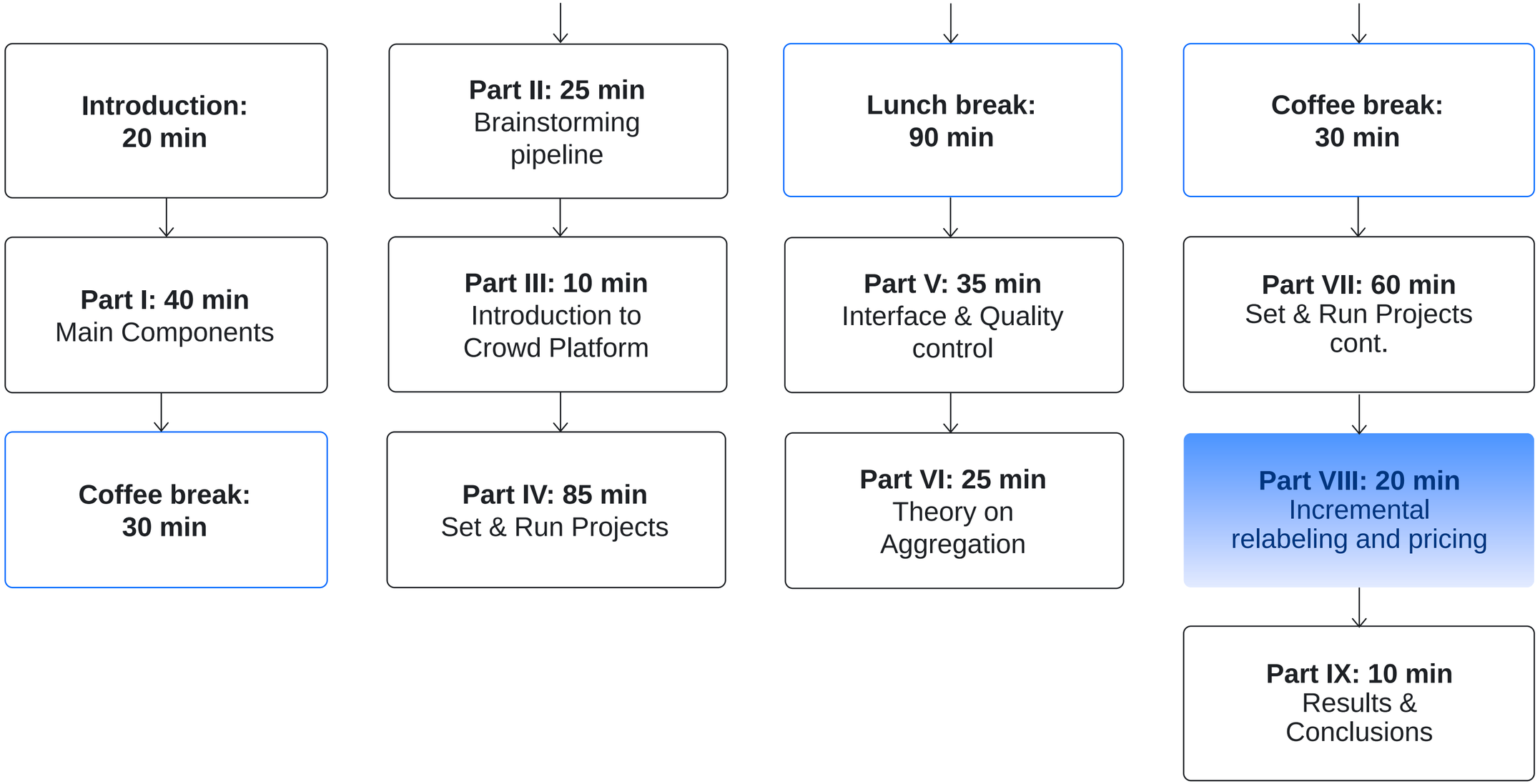
Part VIII

Theory on incremental relabelling and pricing

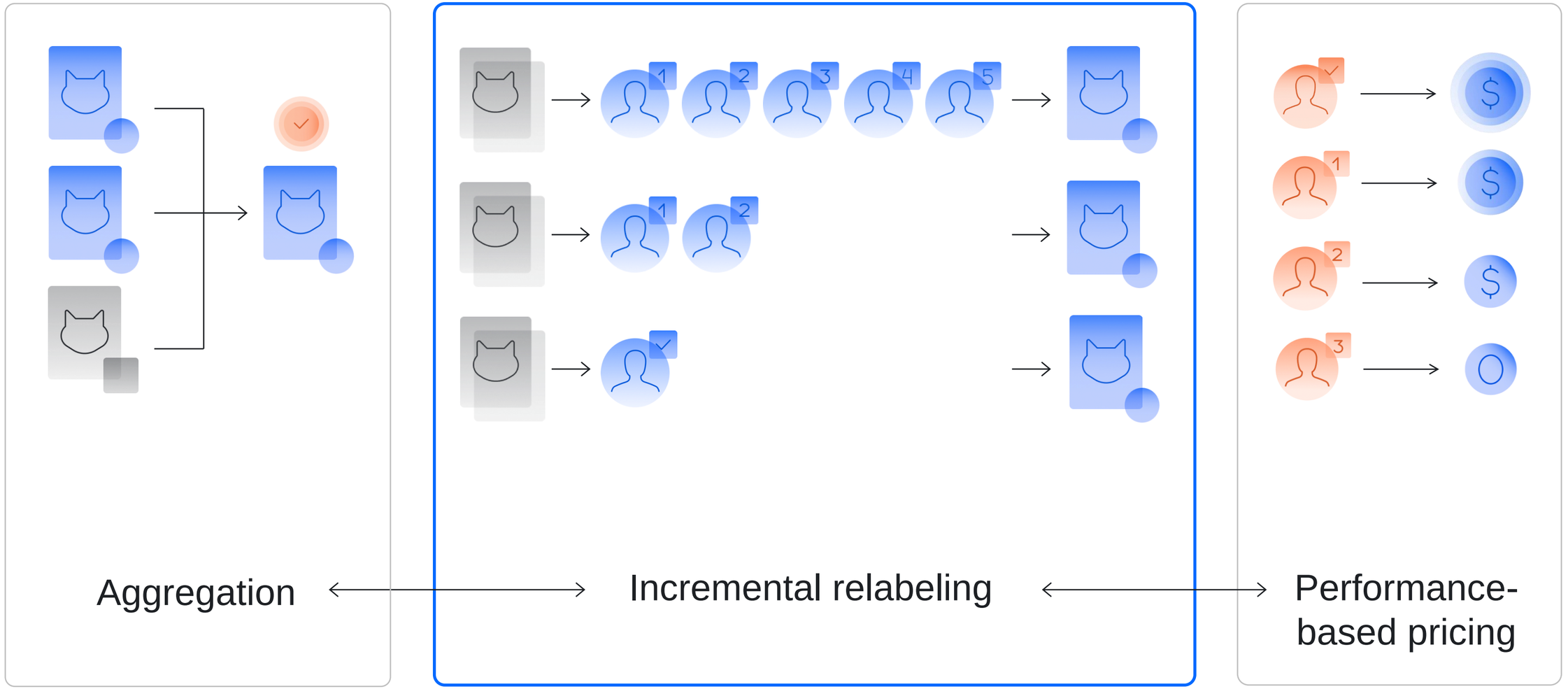
Valentina Fedorova,
Research analyst

Toloka

Tutorial schedule

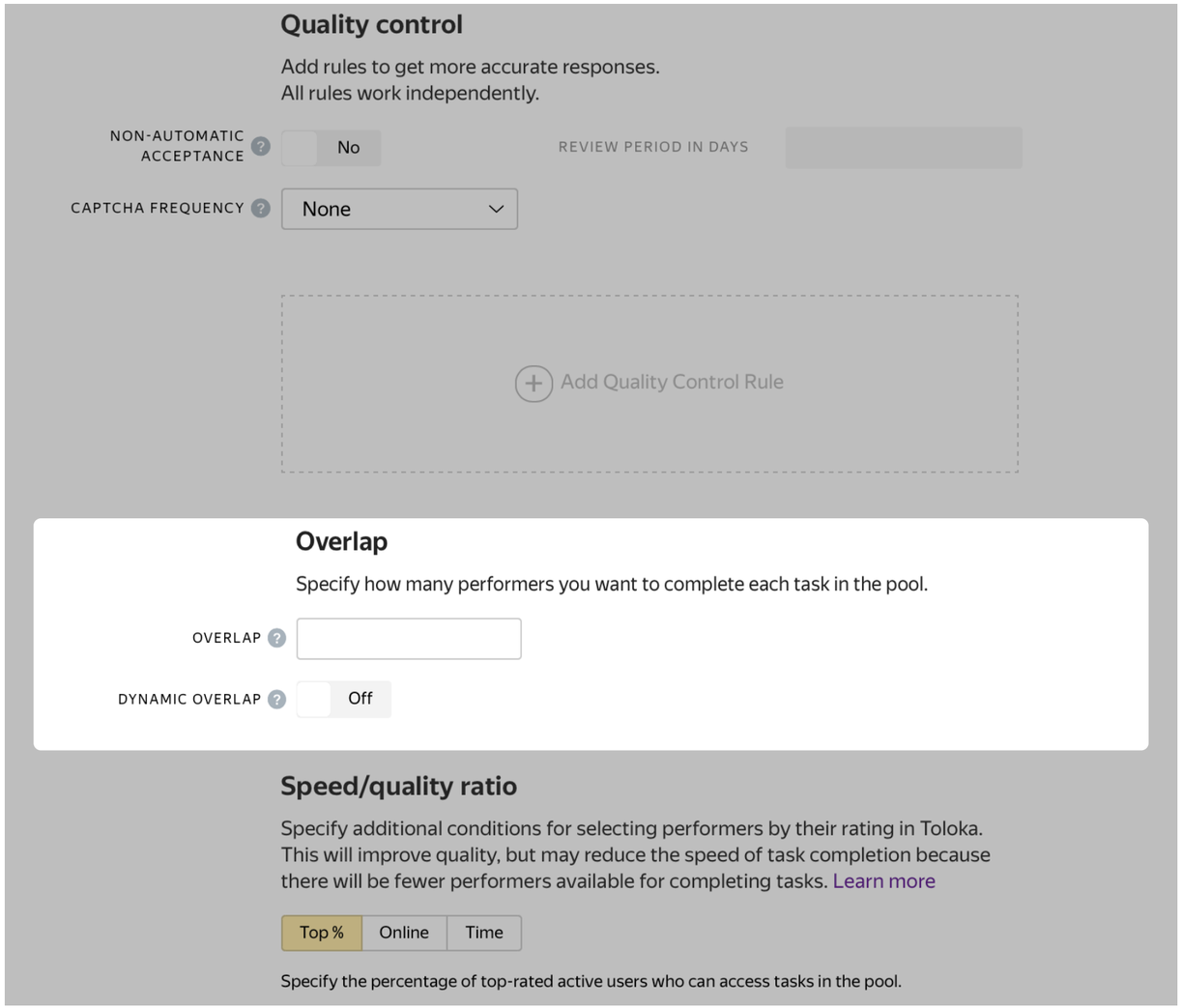


Key components of labeling with crowds



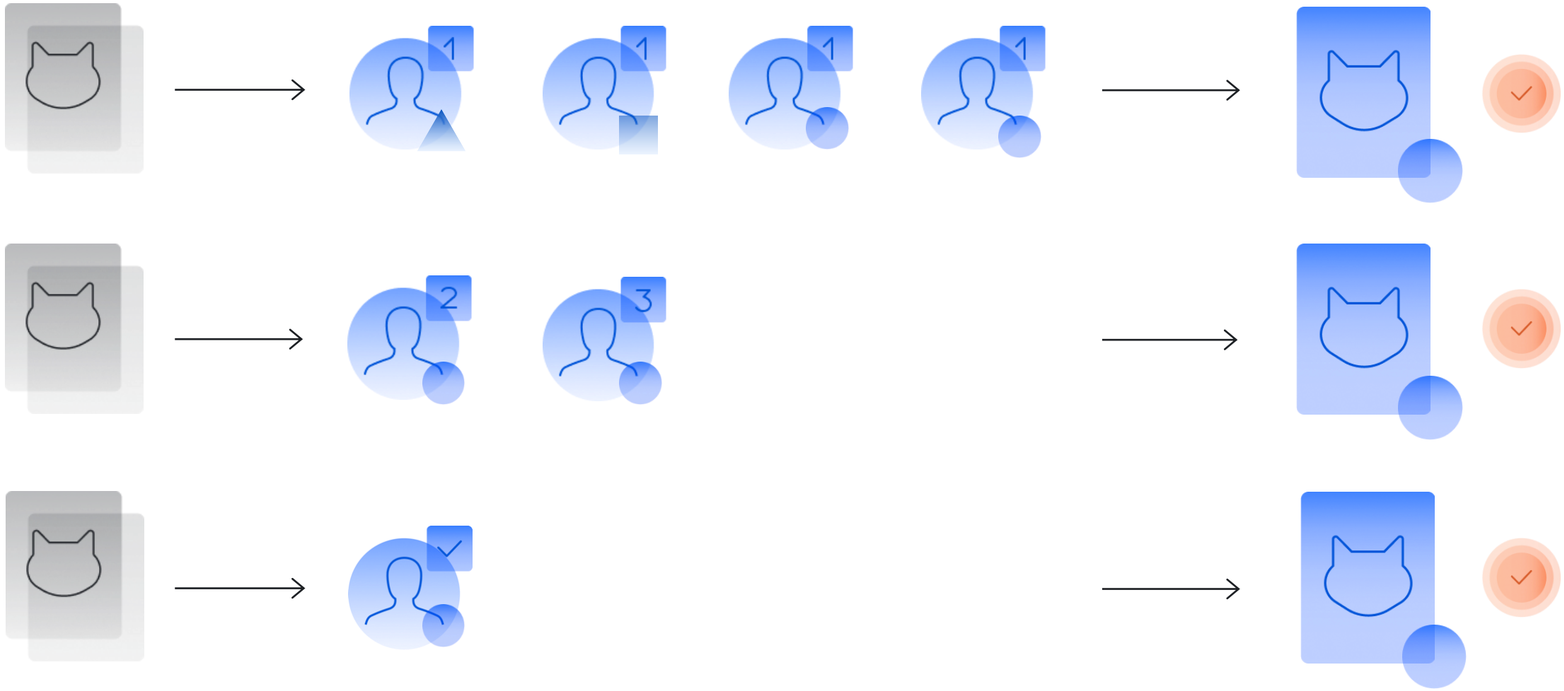
Incremental relabeling
aka dynamic overlap

Pool settings: dynamic overlap



Incremental relabeling problem

Obtain aggregated labels of a desired quality level using a fewer number of noisy labels



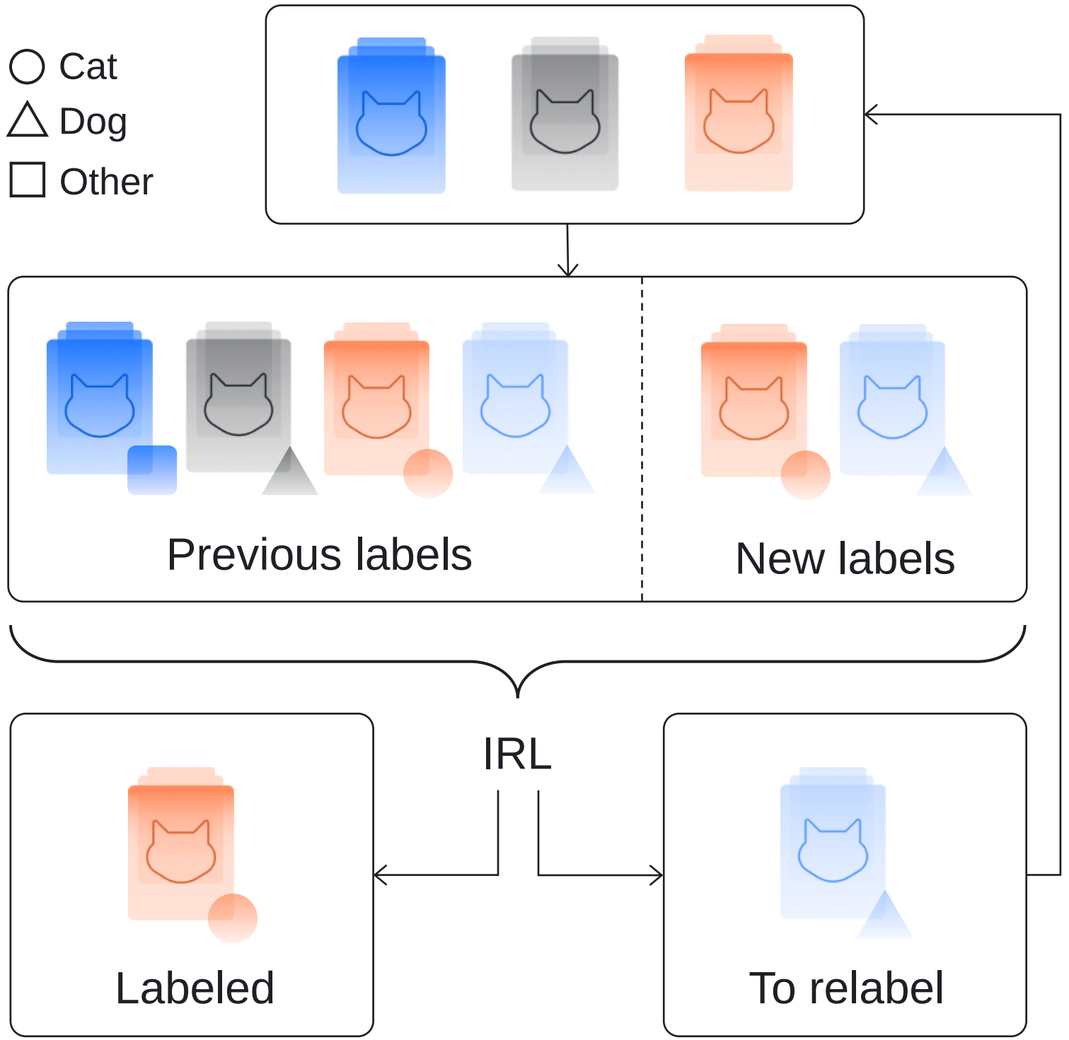
Incremental relabeling scheme (IRL)

Request 1 label for each object

In real time IRL algorithm receives:
(1) previously accumulated labels
(2) new labels

Decides:
(1) which objects are labeled
(2) which objects to relabel

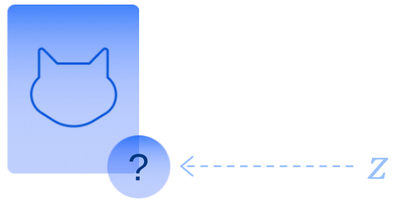
Repeat until all tasks are labeled



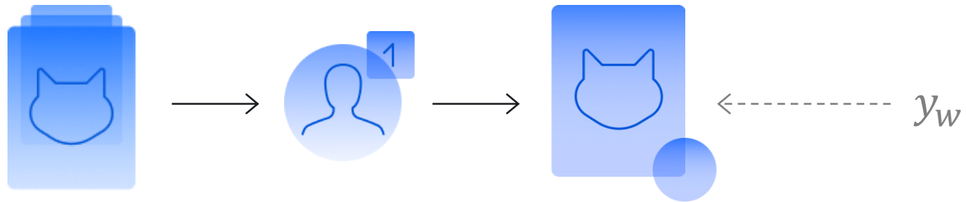
Notations

- Consider one object

- $z \in \{1, \ldots, K\}$ — latent true label



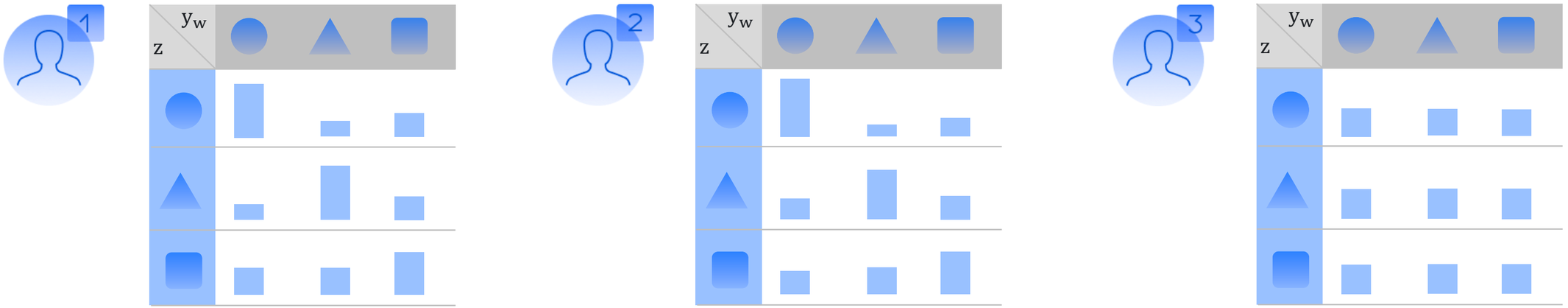
- $y_w \in \{1, \ldots, K\}$ — observed noisy label from worker w:



Notations

- Noisy label model for worker w:

$M_w \in [0,1]^{K \times K}$: $\Pr(Y_w = k | Z = c) = M_w[c, k]$



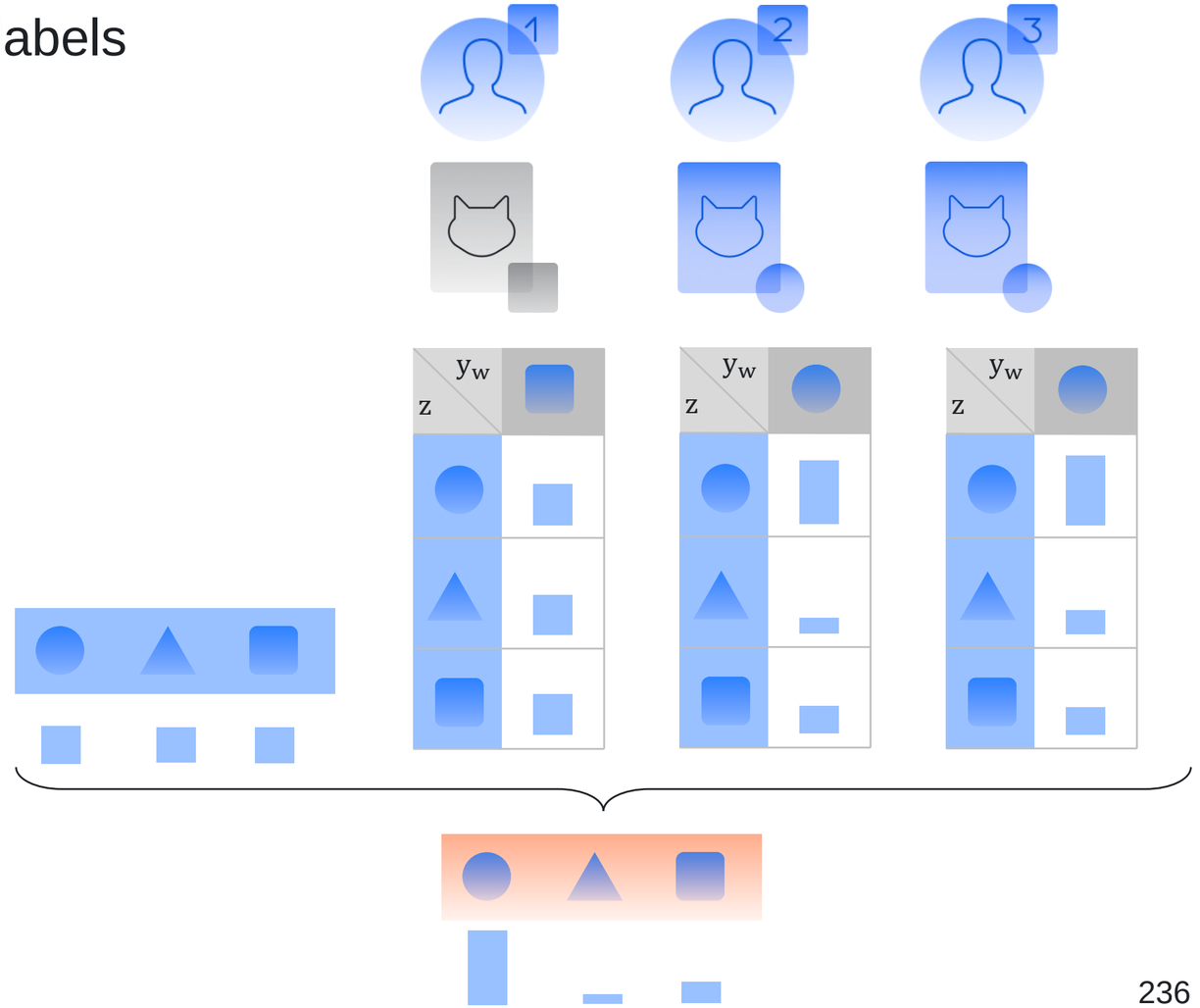
- Prior distribution: $\Pr(Z = k) = p_k$

Posterior distribution

- $\{y_{w_1}, \dots, y_{w_n}\}$ — accumulated noisy labels for the object

- Using Bayes rule:

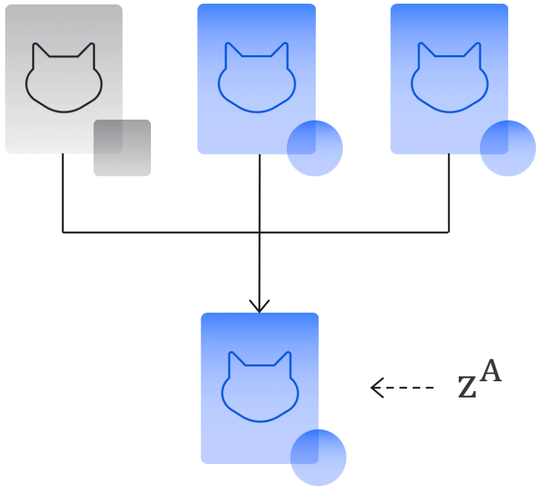
$$\Pr(Z = k | \{y_{w_1}, \dots, y_{w_n}\})$$
$$= \frac{\Pr(Z = k)\Pr(\{y_{w_1}, \dots, y_{w_n}\} | Z = k)}{\Pr(\{y_{w_1}, \dots, y_{w_n}\})}$$
$$= \frac{p_k \prod_{i=1}^{n} M_{w_i}[k, y_{w_i}]}{\sum_{t=1}^{K} p_t \prod_{i=1}^{n} M_{w_i}[t, y_{w_i}]}$$



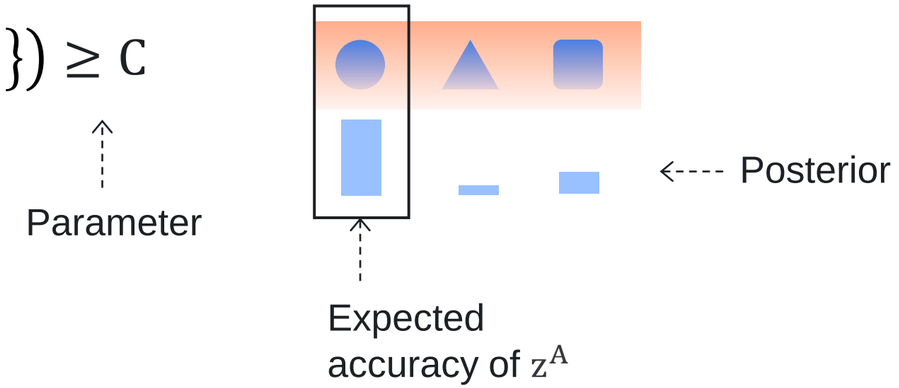
Expected accuracy of aggregated labels

- Let A be an aggregation model, e.g. MV, DS, GLAD,…
- Denote aggregated label $z^A = A(\{y_{w_1}, \cdots, y_{w_n}\})$
- Expected accuracy of aggregated labels given noisy labels is

$$E(\delta(z = z^A) | \{y_{w_1}, \cdots, y_{w_n}\}) = Pr(z = z^A | \{y_{w_1}, \cdots, y_{w_n}\})$$



- Stop labeling if $E(\delta(z = z^A) | \{y_{w_1}, \cdots, y_{w_n}\}) \geq C$



Sheng VS, Provost F, Ipeirotis PG. Get another label? improving data quality and data mining using multiple, noisy labelers. KDD 2008

Incremental relabeling algorithm

Input: $U_{t=1}^{T-1}\ Y^t$ — previous labels till step T

Y^T — new labels

Output: R — objects to relabel

For each object j with a label in Y^T: ← Object with a new label

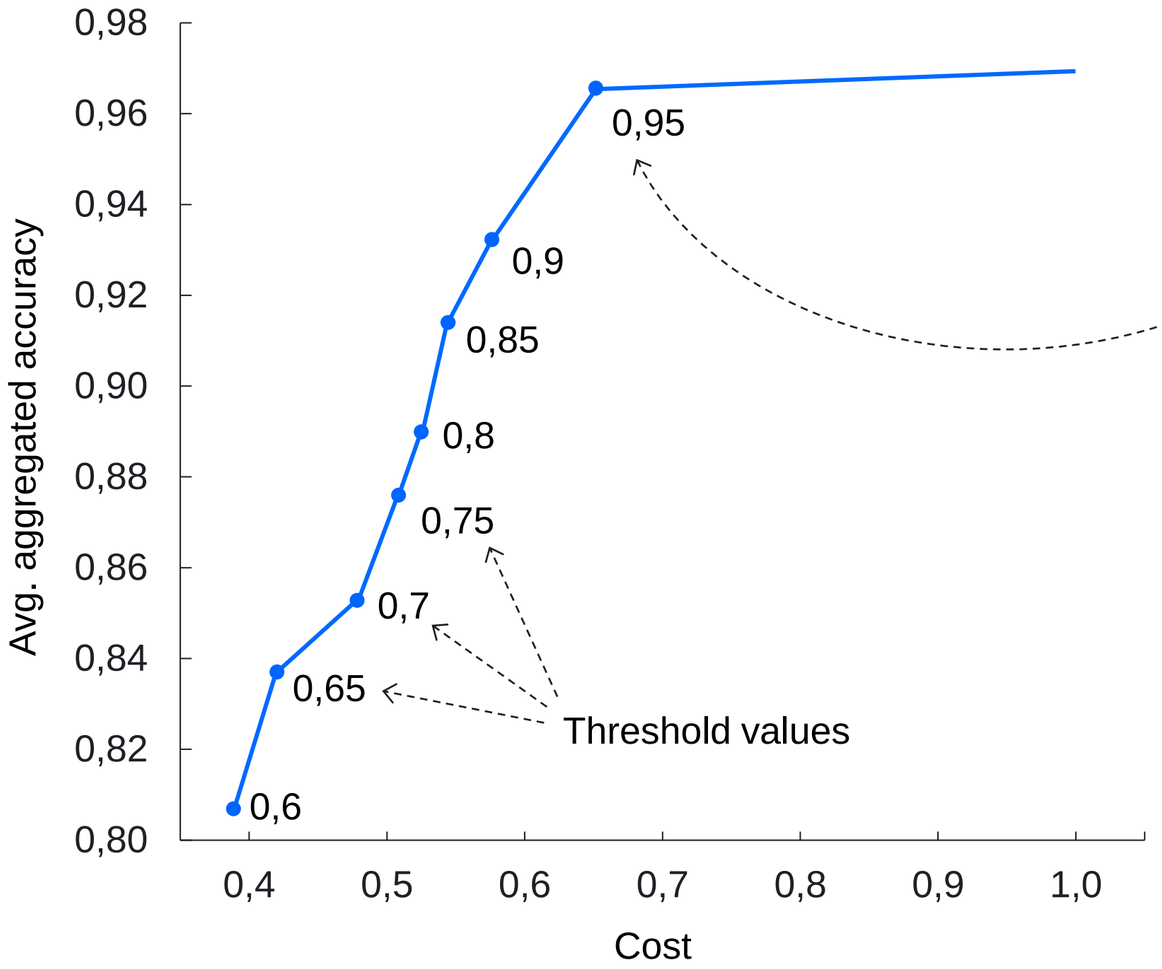
$z_j^M = M(U_{t=1}^{T}\ Y^t)$ ← Current aggregated label

$c_j = E(z_j = z_j^M | U_{t=1}^{T}\ Y^t)$ ← Expected accuracy for the current aggregated label

If $c_j < c$, then $R = R \cup j$

↑ Parameter: c — threshold for expected accuracy

Threshold in IRL: cost – accuracy trade-off



- Optimal threshold $c = 0.95$
- A higher c does not increase accuracy
- Saving ≈ 35% of noisy labels

How to obtain a cost-accuracy plot

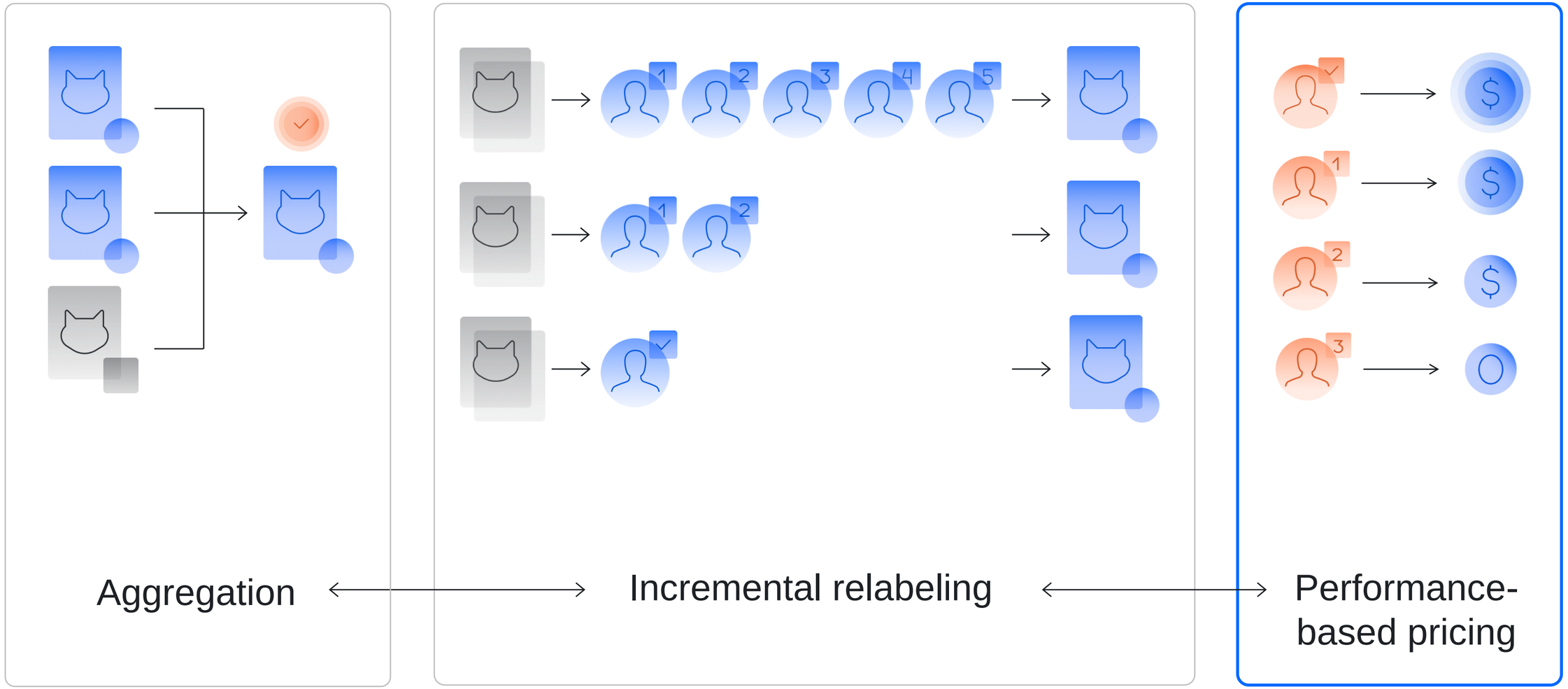
Data for the plot:

- Label a pool of objects with a redundant overlap (e.g., 10)
- Obtain ground truth labels for the objects (e.g., expert labels or MV labels)

Simulate IRL with different thresholds using the data:

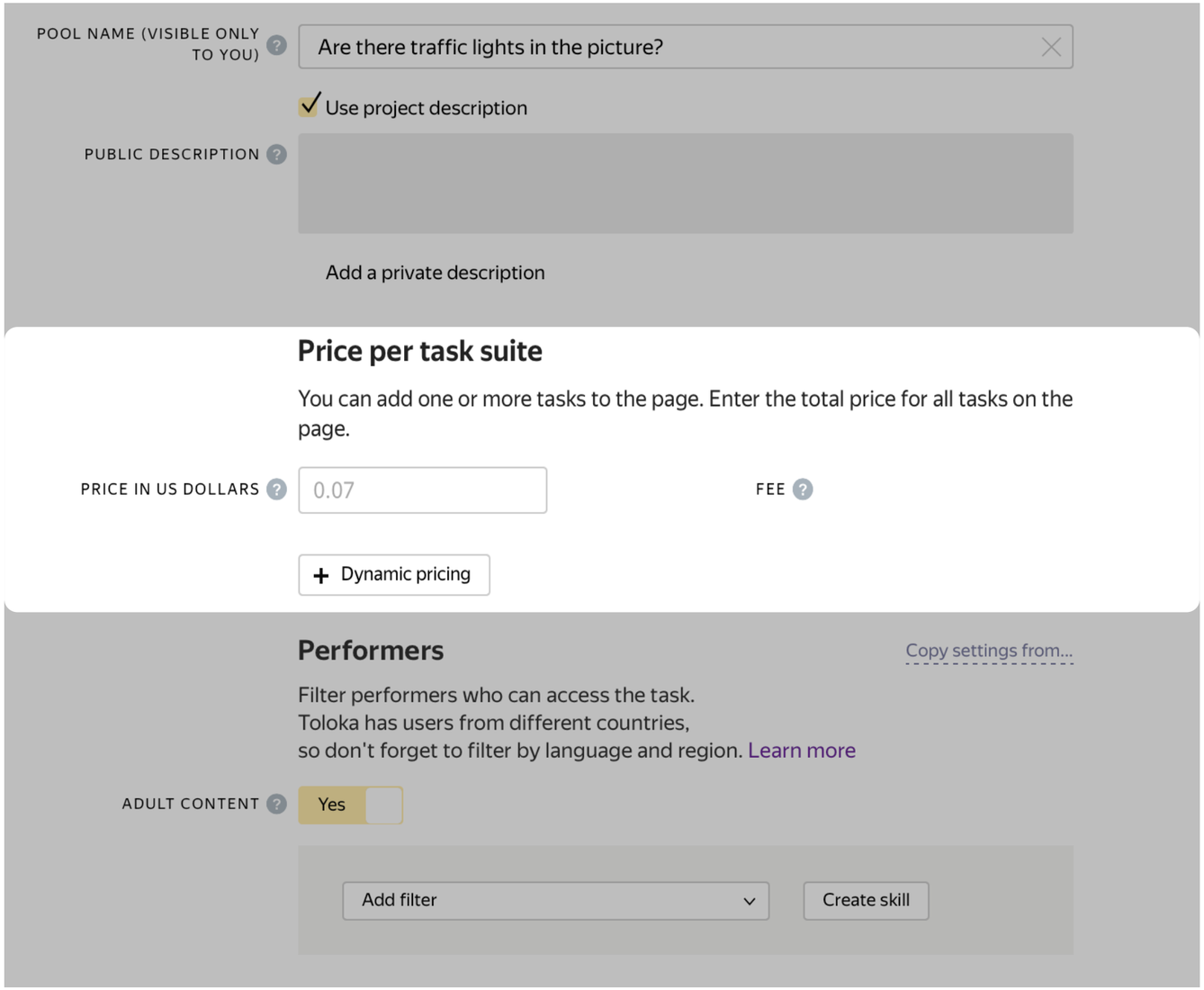
- For each threshold c from a grid $0 < c_0 < ... < c_m \leq 1$
- Repeat N times:
  1. Shuffle noisy labels and fix the order of labels
  2. Draw labels sequentially and test the IRL condition after each label
  3. Once the IRL condition for an object is met, discard unused labels for the object
  4. When all objects are labelled calculate
     - accuracy of aggregated labels
     - cost as the fraction of used noisy labels
- Average N values of aggregated accuracy and N values of cost for each value of threshold c

Key components of labeling with crowds

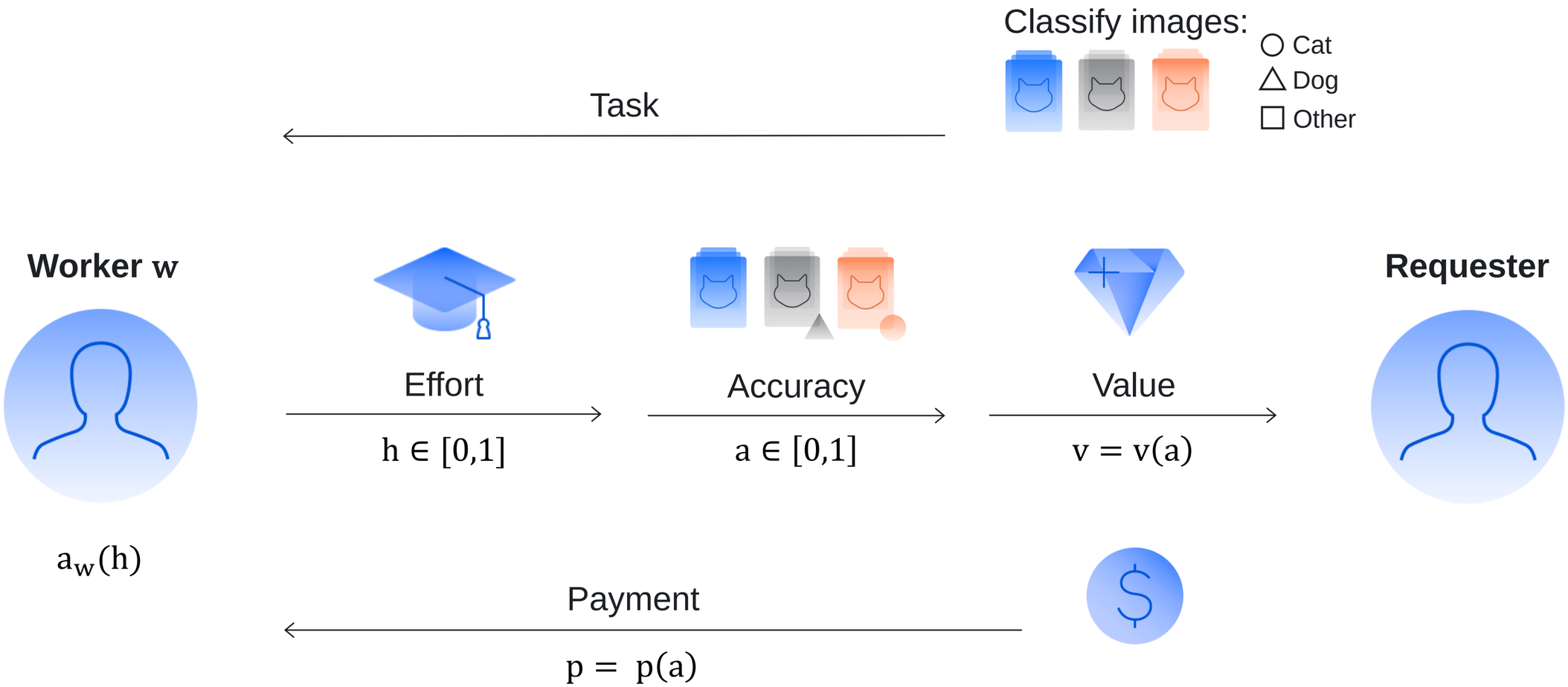


Performance-based pricing
aka dynamic pricing

Pool settings: dynamic pricing

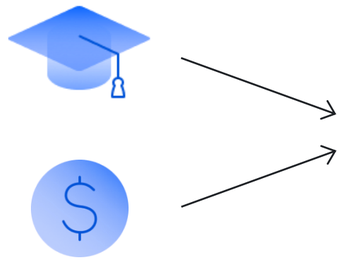


Labeling as a game: notation



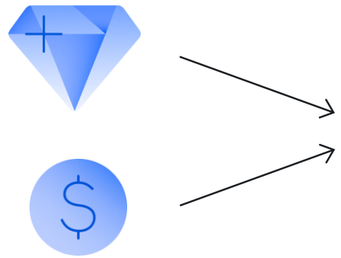
Labeling as a game: formalization

- Each worker w chooses a level of effort h for labeling object to maximize earnings per unit of spent effort:



$$\frac{p(a_w(h))}{h} \to \max_{h \geq 0}$$

- The requester chooses a pricing p(a) to minimize payments per unit of obtained value



$$\frac{v(a)}{p(a)} \to \max_{a \in [0,1]}$$

Labeling as a game: incentive compatible pricing

- Assume $a_w(h)$ is a linear function of h:

$$a_w(h) = c_1 h + c_0$$

Accuracy

Theorem: The requester and workers maximize their utility simultaneously if the pricing $p(a)$ for each label is proportional to its accuracy a

Performance-based pricing in practice: settings

- Price p for the level of accuracy a_0 : $\Pr(\hat{z} = z) \geq a_0$ E.g.:

$p = 0.3\$$ $a_0 = 0.99$

- $\hat{q}_w = \Pr(y^w = z)$ — estimated quality level of worker w, e.g. the fraction of correct labels for golden set (GS):

5 correct GS among 10
$\hat{q}_w = 0.5$

16 correct GS among 20
$\hat{q}_w = 0.8$

100 correct GS among 100
$\hat{q}_w = 1$

Performance-based pricing in practice: settings

- Aggregation $\hat{z}_j^{wMV} = \arg\max_{y=1,\dots,K} \sum_{w \in W_j} \hat{q}_w \delta(y = y_j^w)$

- IRL algorithm is based on the expected accuracy of $\hat{z}_j^{wMV}$

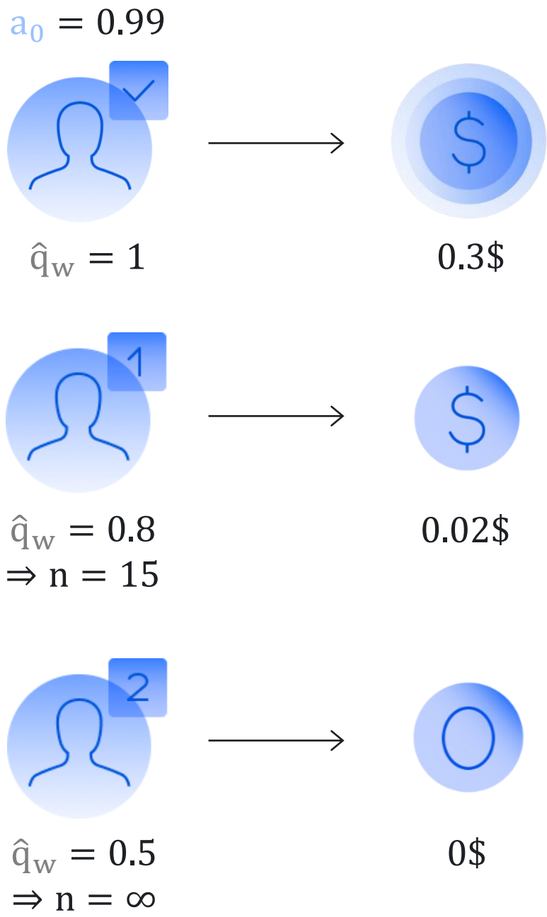
Performance-based pricing in practice

- Pricing rules

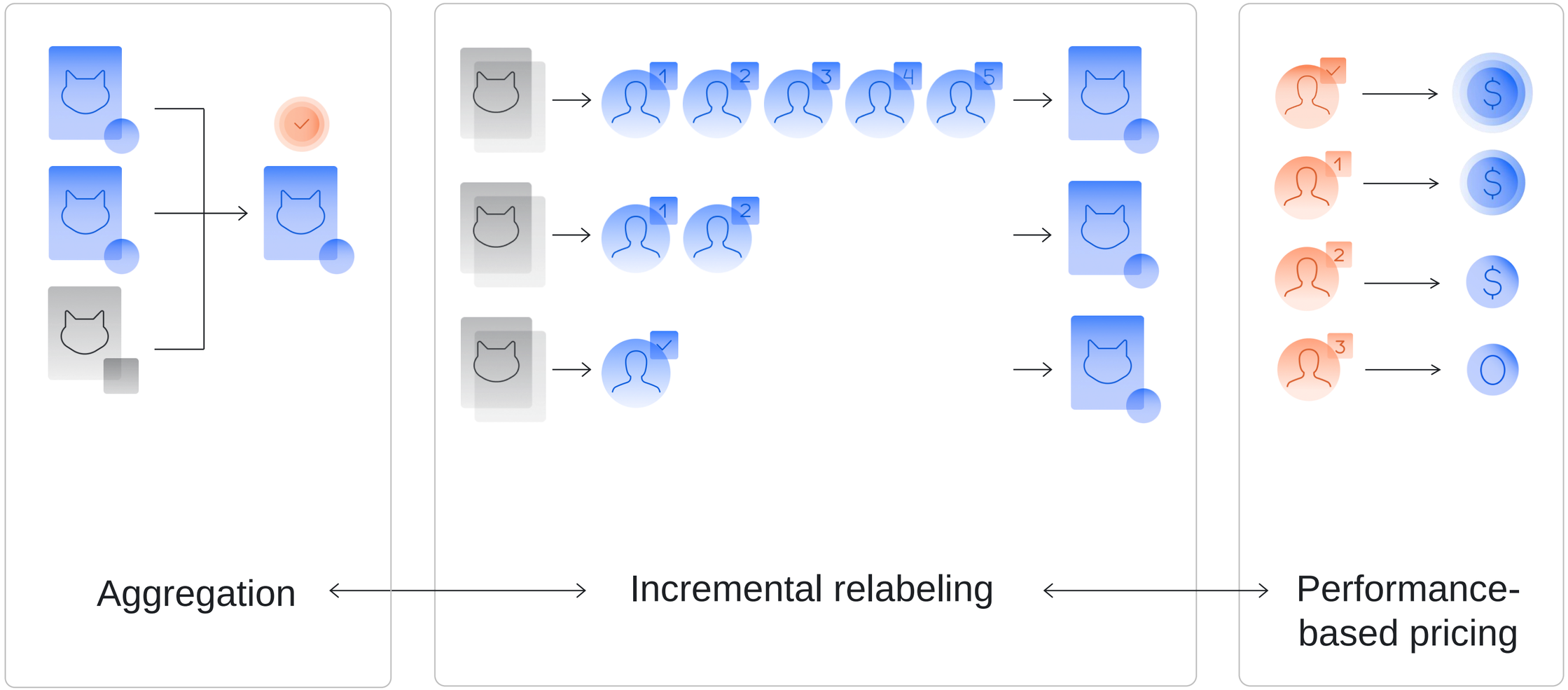
1. If $\hat{q}_w \geq a_0$, then the price is p
2. Else find n:

$$\underbrace{\sum_{k=0}^{n/2} \binom{n}{k} \hat{q}_w^{n-k} (1-\hat{q}_w)^k}_{\text{Expected accuracy for MV}} \geq a_0$$

The price is p/n



Key components of labeling with crowds

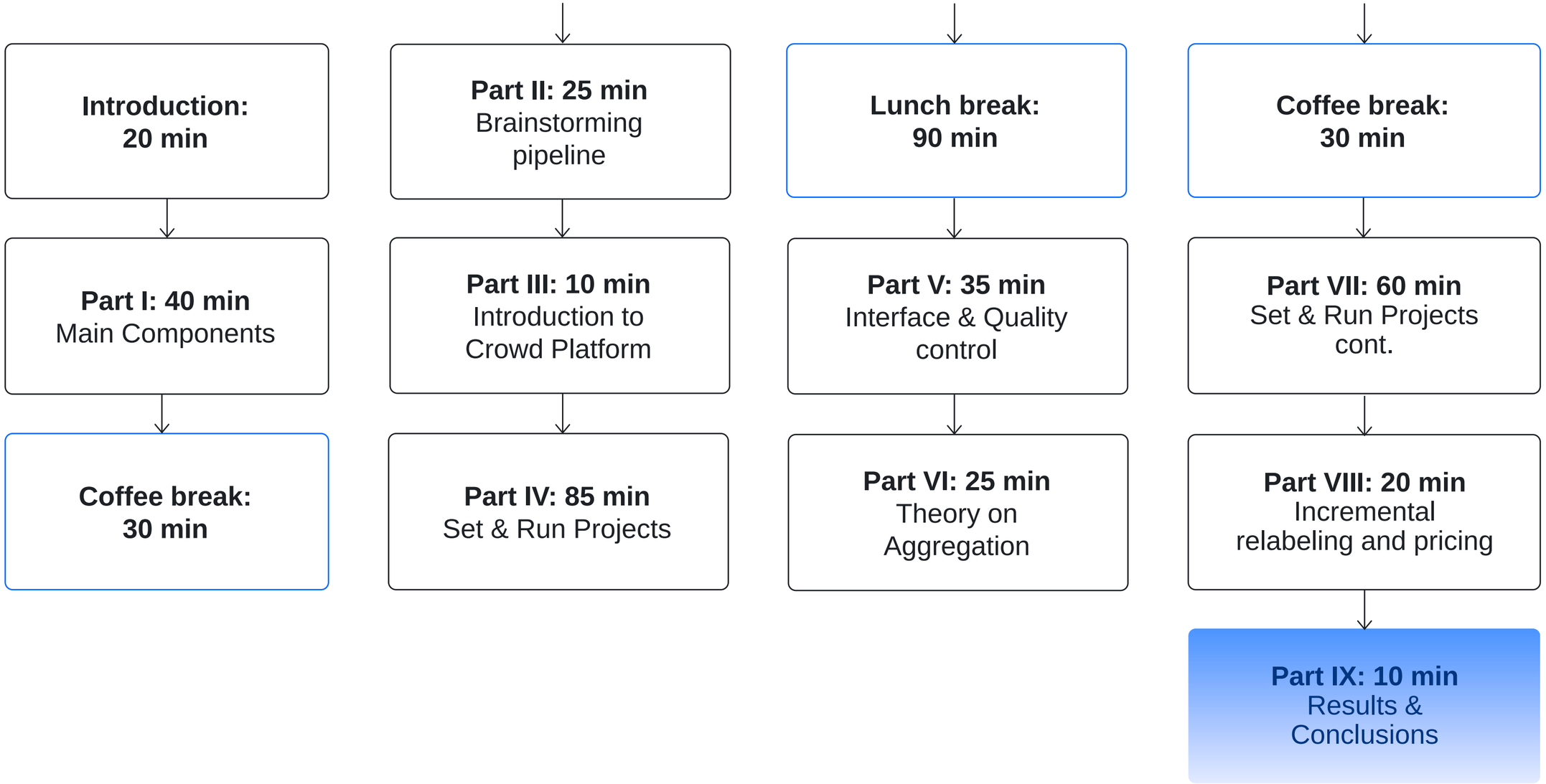


Part IX

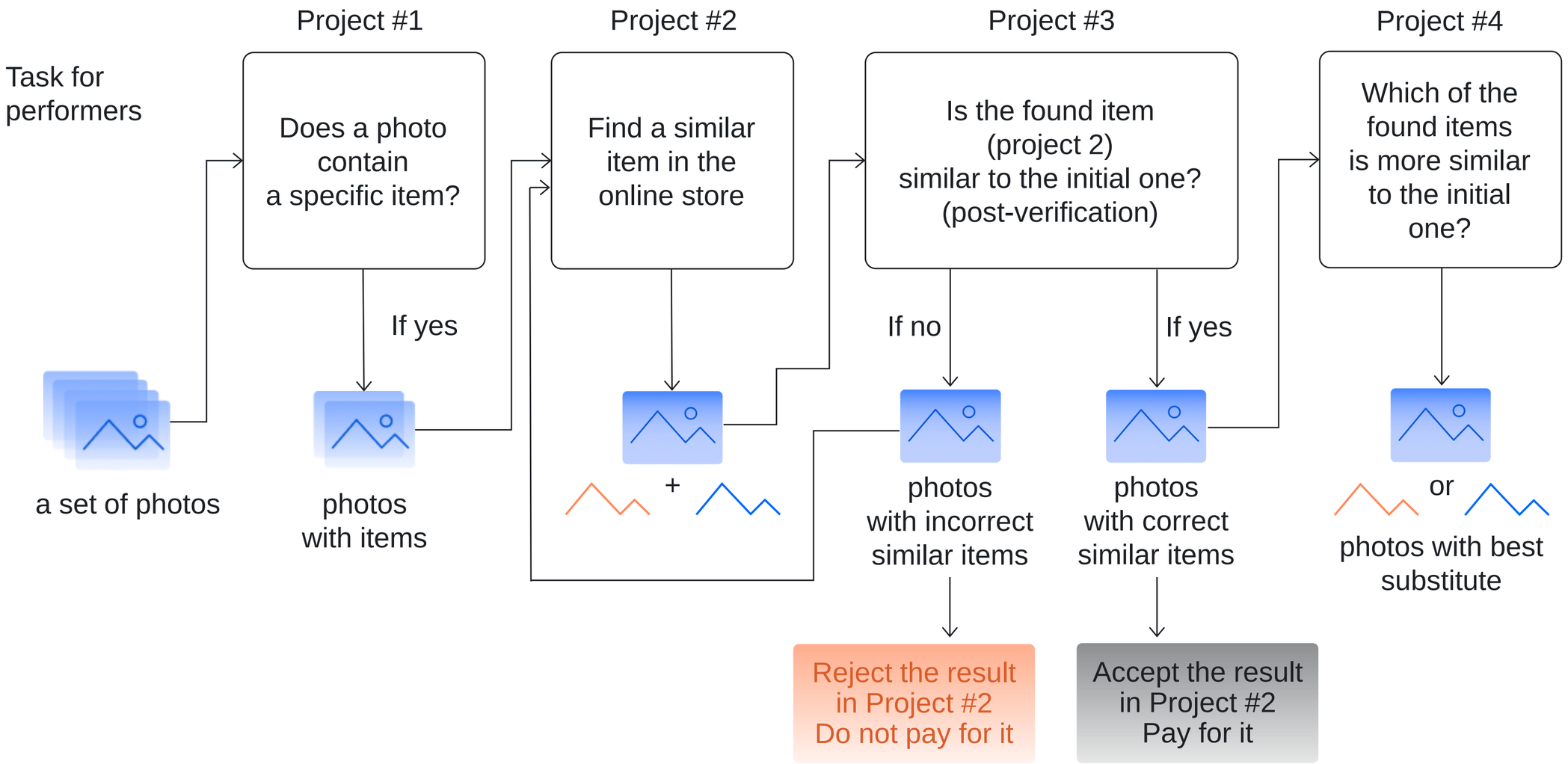
Discussion of the projects’ results. Conclusions

Alexey Drutsa,
Head of Efficiency and Growth Division, Toloka

Tutorial schedule



Reminder: we implemented this pipeline



Project #1: Filter out photos without objects

Task

- Does a photo contain an item of desired type?

Are there **shoes** in the picture?

Yes No Picture not found

Our results

- Amount: 30 photos
- Overlap: 3
- Time: 5 min
- Cost: $0.09 + Toloka fee

Project #2: Searching for similar items on the online store

Task

- Find a similar item on the internet

Our results

- Amount: 25 photos
- Overlap: 3
- Time: 25 min
- Cost: $1.74 + Toloka fee

Find the same **shoes** on ASOS

ASOS

Shoes must be the same color and the same style.

Paste the link here

Upload the image here. The image should show the shoes you found.

Project #3: Accept correctness of items found

Task

- Is the found item (project 2) similar to the initial one?

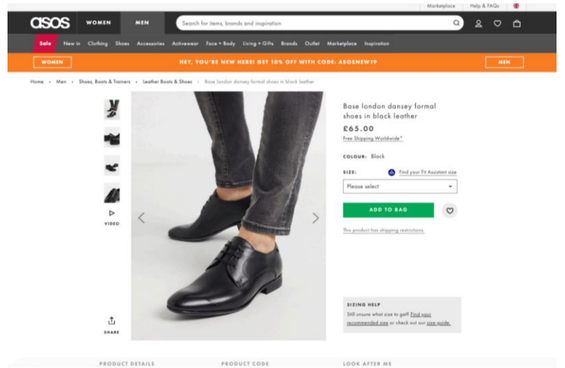
Our results

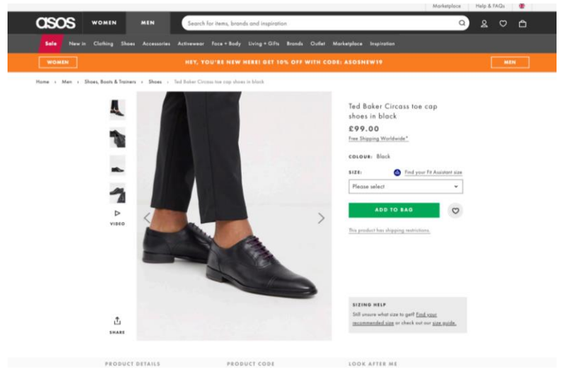
- Amount: 75 photos
- Overlap: 3
- Time: 3 min
- Cost: $0.20 + Toloka fee

Project #4: Decide which substitute works best

Task

- Which of the items is similar to the initial one?





Our results

- Amount: 62 photos
- Overlap: 3
- Time: 10 min
- Cost: $0.10 + Toloka fee

Statistics over the whole pipeline

- 30 photos processed to find the substitute items and evaluate their similarity
- Within 45 min on real performers
- Total cost: $2.15 + Toloka fee

Afterparty: upgrade your pipeline

To obtain more comprehensive data

- Use more item types at the same time

To reduce costs

- Use incremental relabeling aka Dynamic overlap

To improve quality

- Use dynamic pricing
- Add more Golden Sets and hints
- Experiment with aggregation methods
- Add training for performers

API of Toloka

Allows you to automate all steps of our pipeline

- Discover at: https://yandex.com/dev/toloka/

Crowdsource all types of data

Search Relevance

Moderation

Generation of content

Computer vision

Speech Technologies

References: Aggregation

1. Dawid, A. P. and Skene, A. M, Maximum likelihood estimation of observer error-rates using the EM algorithm, Applied statistics 1979
2. Whitehill, J., Wu, T., Bergsma, J., Movellan, J. R, Ruvolo, P. L, Whose vote should count more: Optimal integration of labels from labelers of unknown expertise}, NIPS 2009
3. Zhou, D., Liu, Q., Platt, J. C, Meek, C., Shah, N. B, Regularized minimax conditional entropy for crowdsourcing, arXiv preprint 2015
4. Raykar, V. C, Yu, S., Zhao, L. H, Valadez, G. H., Florin, C., Bogoni, L., Moy, L., Learning from crowds, JMLR 2010
5. Snow, R., O'Connor, B., Jurafsky, D., Ng, A. Y, Cheap and fast---but is it good?: evaluating non-expert annotations for natural language tasks, EMNLP 2008
6. Ruvolo, P., Whitehill, J.,Movellan, J. R, Exploiting Commonality and Interaction Effects in Crowdsourcing Tasks Using Latent Factor Models, NIPS ’13 Workshop on Crowdsourcing: Theory, Algorithms and Applications
7. Faridani, S. and Buscher, G., LabelBoost: An Ensemble Model for Ground Truth Inference Using Boosted Trees, HCOMP 2013
8. Welinder, P., Branson, S., Perona, P., Belongie, S. J , The multidimensional wisdom of crowds, NIPS 2010
9. Jin, Y., Carman, M., Kim, D., Xie, L., Leveraging Side Information to Improve Label Quality Control in Crowd-Sourcing, HCOMP 2017
10. Imamura, H., Sato, I., Sugiyama, M., Analysis of Minimax Error Rate for Crowdsourcing and Its Application to Worker Clustering Model, arXiv preprint 2018

References: Aggregation

11. Sheshadri, A. and Lease, M., Square: A benchmark for research on computing crowd consensus, HCOMP 2013
12. Kim, H. and Ghahramani, Z., Bayesian classifier combination, AISTATS 2012
13. Venanzi, M., Guiver, J., Kazai, G., Kohli, P., Shokouhi, M., Community-based bayesian aggregation models for crowdsourcing, WWW2014
14. Vuurens, J., de Vries, A. P, Eickhoff, C., How much spam can you take? an analysis of crowdsourcing results to increase accuracy, SIGIR Workshop CIR 2011
15. Chen, X. and Bennett, P. N and Collins-Thompson, K. and Horvitz, E., Pairwise ranking aggregation in a crowdsourced setting, WSDM 2013
16. Liu, C. and Wang, Y., Truelabel+ confusions: A spectrum of probabilistic models in analyzing multiple ratings, ICML 2012

References: Incremental relabeling & Pricing

17. Ipeirotis, P. G and Provost, F. and Sheng, V. S and Wang, J., Repeated labeling using multiple noisy labelers, KDD 2014
18. Abraham, I., Alonso, O., Kandylas, V., Patel, R., Shelford, S., Slivkins, A., How many workers to ask?: Adaptive exploration for collecting high quality labels, SIGIR 2016
19. Ertekin, S., Hirsh, H., Rudin, C., Learning to predict the wisdom of crowds, arXiv preprint 2012
20. Lin, C. H, Mausam, M., Weld, D. S, To Re(label), or Not To Re(label), HCOMP 2014
21. Zhao, L., Sukthankar, G., Sukthankar, R., Incremental relabeling for active learning with noisy crowdsourced annotations, PASSAT/SocialCom 2011
22. Wang, J., Ipeirotis, P. G, Provost, F., Quality-based pricing for crowdsourced workers, working paper, 2013
23. Cheng, J., Teevan, J., Bernstein, M. S, Measuring crowdsourcing effort with error-time curves, CHI 2015
24. Ho, C., Slivkins, A., Suri, S., Vaughan, J. W., Incentivizing high quality crowdwork, WWW 2015
25. Difallah, D. E., Catasta, M., Demartini, G., Cudr`e-Mauroux, P., Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement, HCOMP 2014
26. Yin, M., Chen, Y., Sun, Y., The effects of performance-contingent financial incentives in online labor markets, AI 2013
27. Shah, N., Zhou, D., Peres, Y., Approval voting and incentives in crowdsourcing, ICML 2015

[26] Shah, N. and Zhou, D., No oops, you won’t do it again: Mechanisms for self-correction in crowdsourcing, ICML 2016

References: Tutorials

27. Crowdsourcing: Beyond Label Generation, Vaughan, J. W. KDD 2017
28. Crowd-Powered Data Mining, Li, G., Wang, J., Fan, J., Zheng, Y., Chai, C., KDD 2018
29. Social Spam Campaigns Social Spam, Campaigns, Misinformation and Crowdturfing, Lee, K., Caverlee, J., Pu, C., WWW2014
30. From Complex Object Exploration to Complex Crowdsourcing, Amer-Yahia, S., Roy, S.B., WWW 2015
31. Crowdsourced Data Management: Overview and Challenges, Li, G., Zheng, Y., Fan, J., Wang, J., Cheng, R, SIGMOD 2017
32. Spatial Crowdsourcing: Challenges, Techniques, and Applications, Tong, Y., Chen, L., Shahab, C., VLDB 2016
33. Truth Discovery and Crowdsourcing Aggregation: A Unified Perspective, Gao, J., Li, Q., Zhao, B., Fan, W., Han, J., VLDB 2015
34. Data-Driven Crowdsourcing: Management, Mining, and Applications, Chen, L., Lee, D., Milo, T., ICDE 15
35. Practice of Efficient Data Collection via Crowdsourcing at Large-Scale, Drutsa A., Fedorova V., Megorskaya O., Zerminova E., KDD 2019