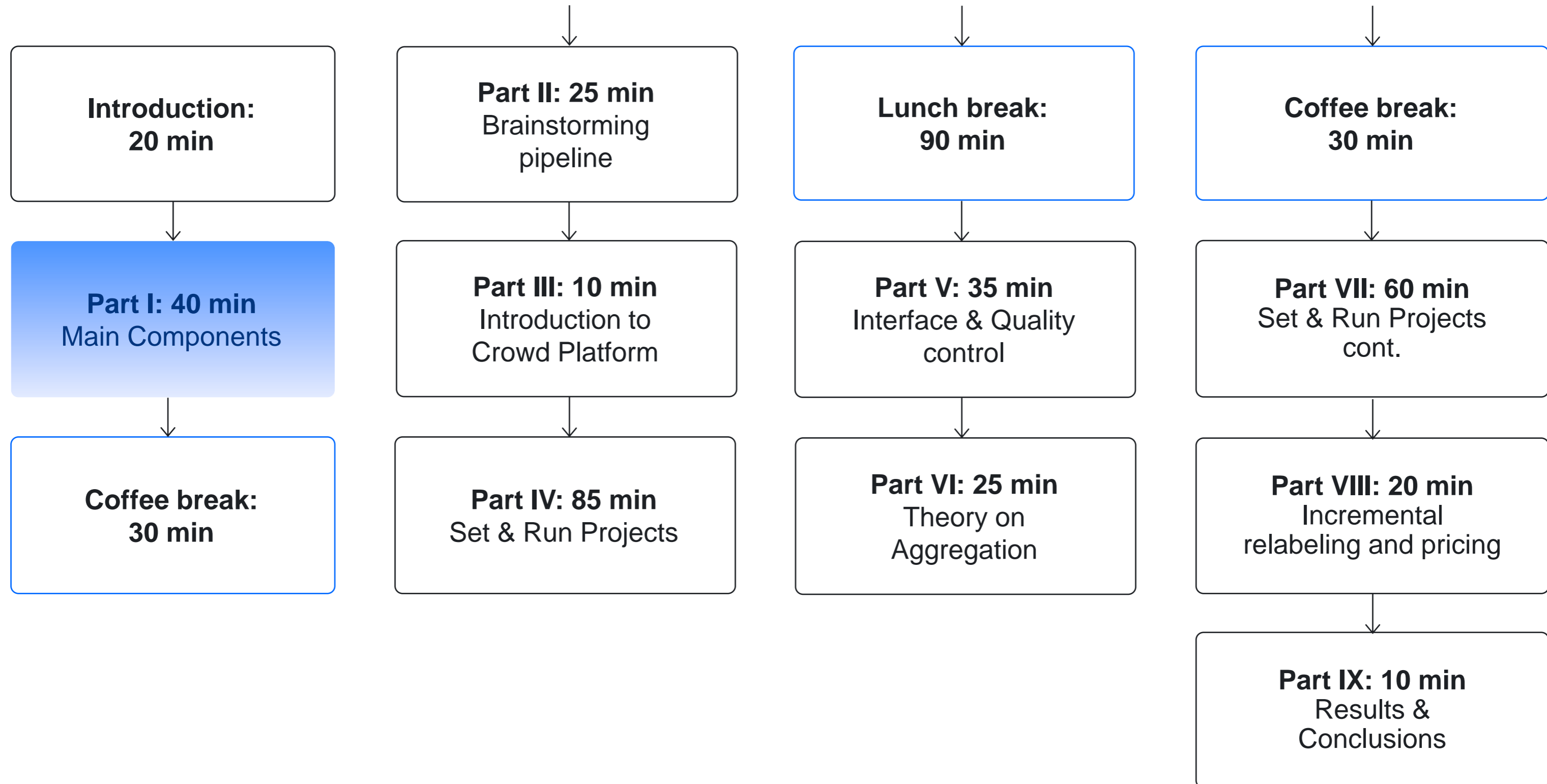


Part I

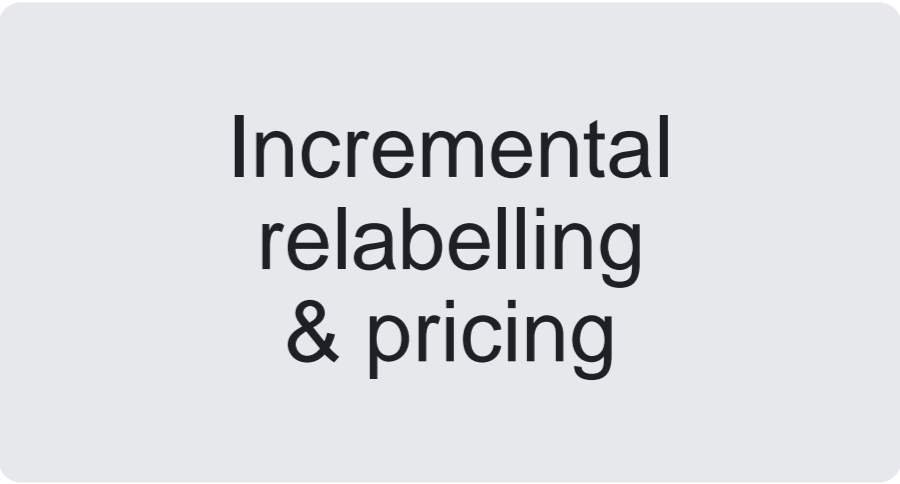
Main components of data collection via crowdsourcing

Olga Megorskaya,
CEO, Toloka

Tutorial schedule

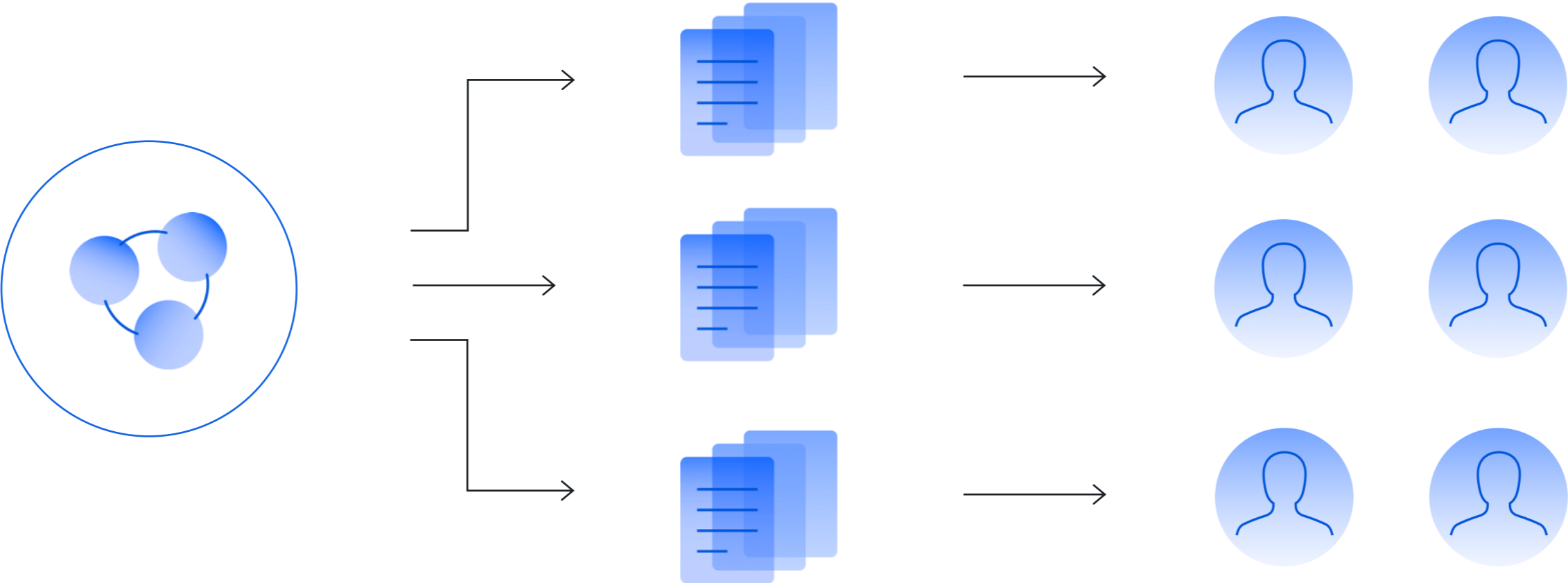


Main components for effective crowdsourcing



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:

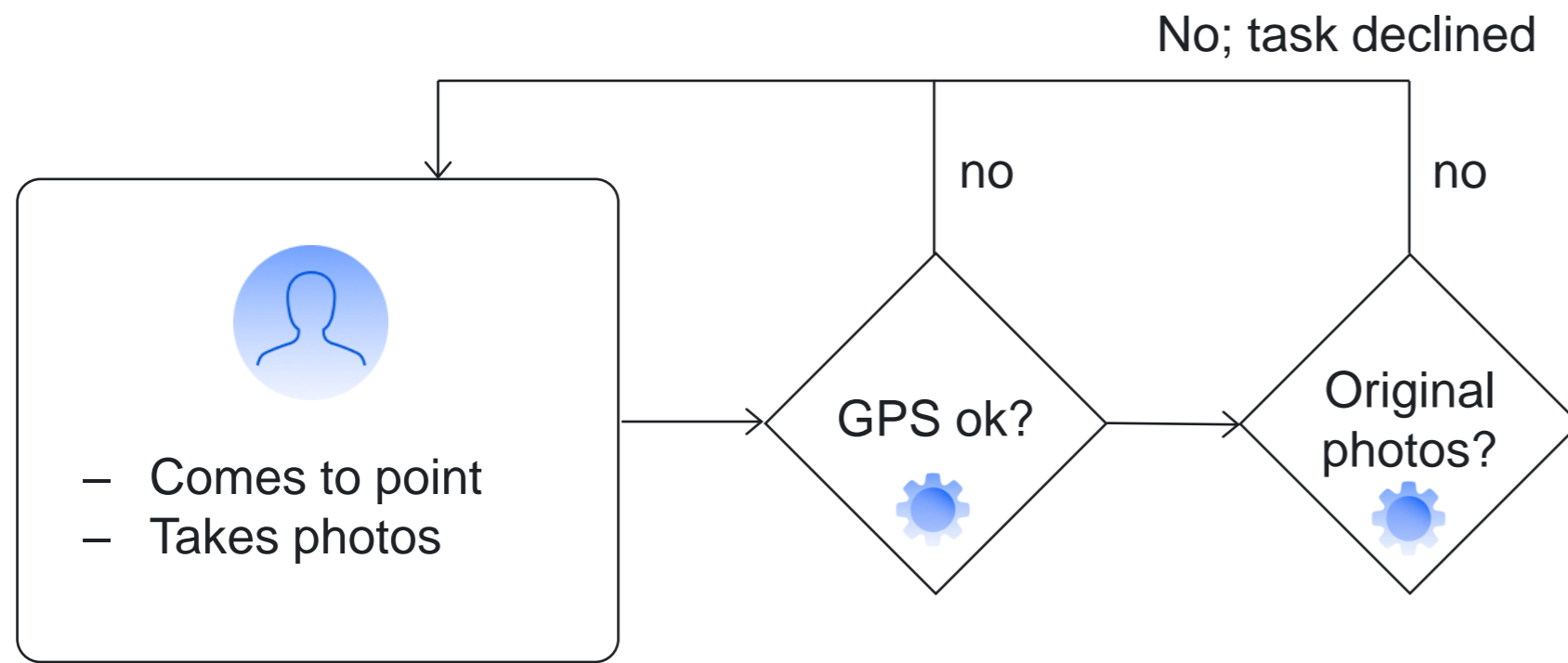
Is the highlighting of all cars made correctly?

Real example: decomposition for a field survey

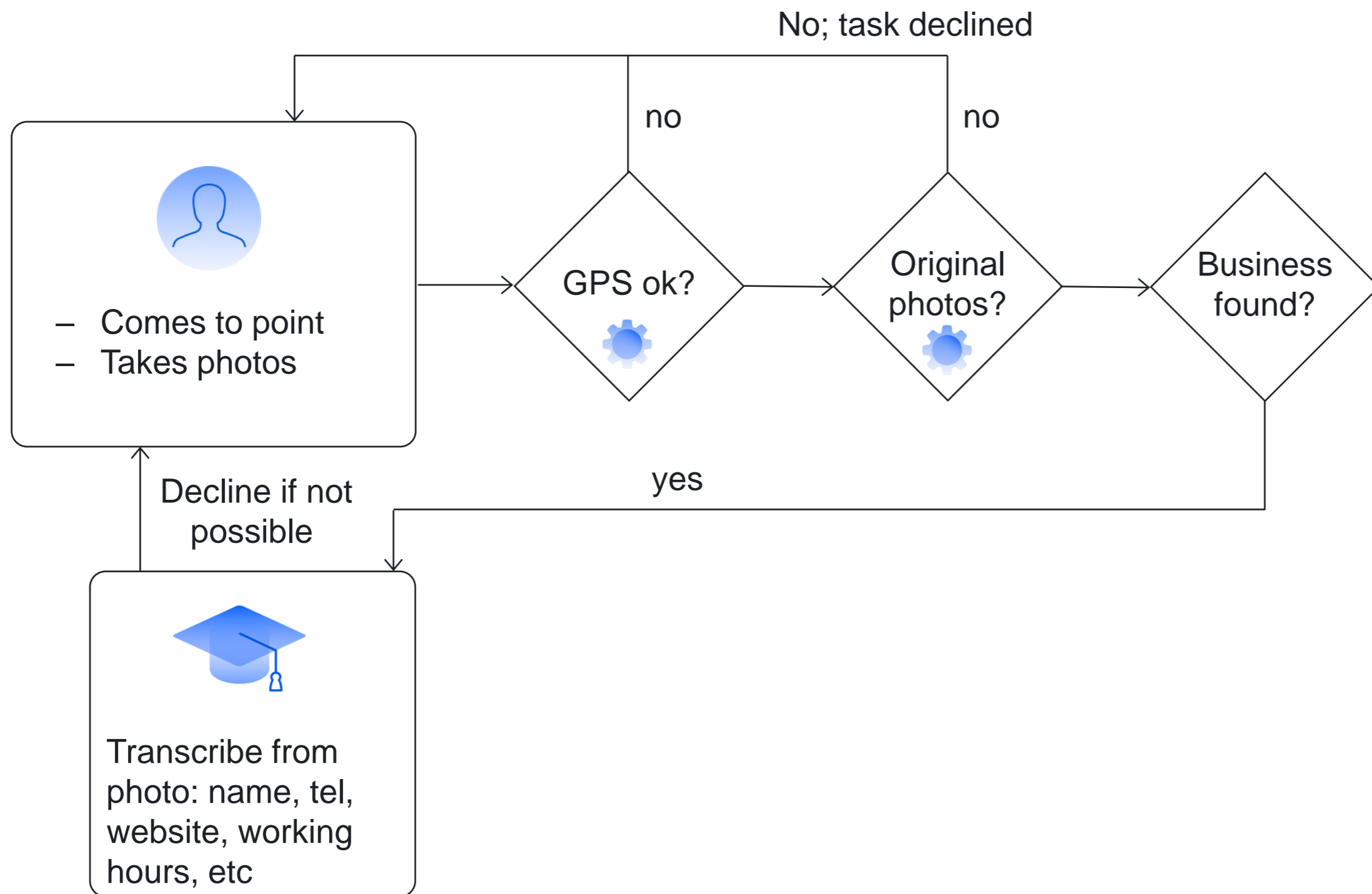


- Comes to point
- Takes photos

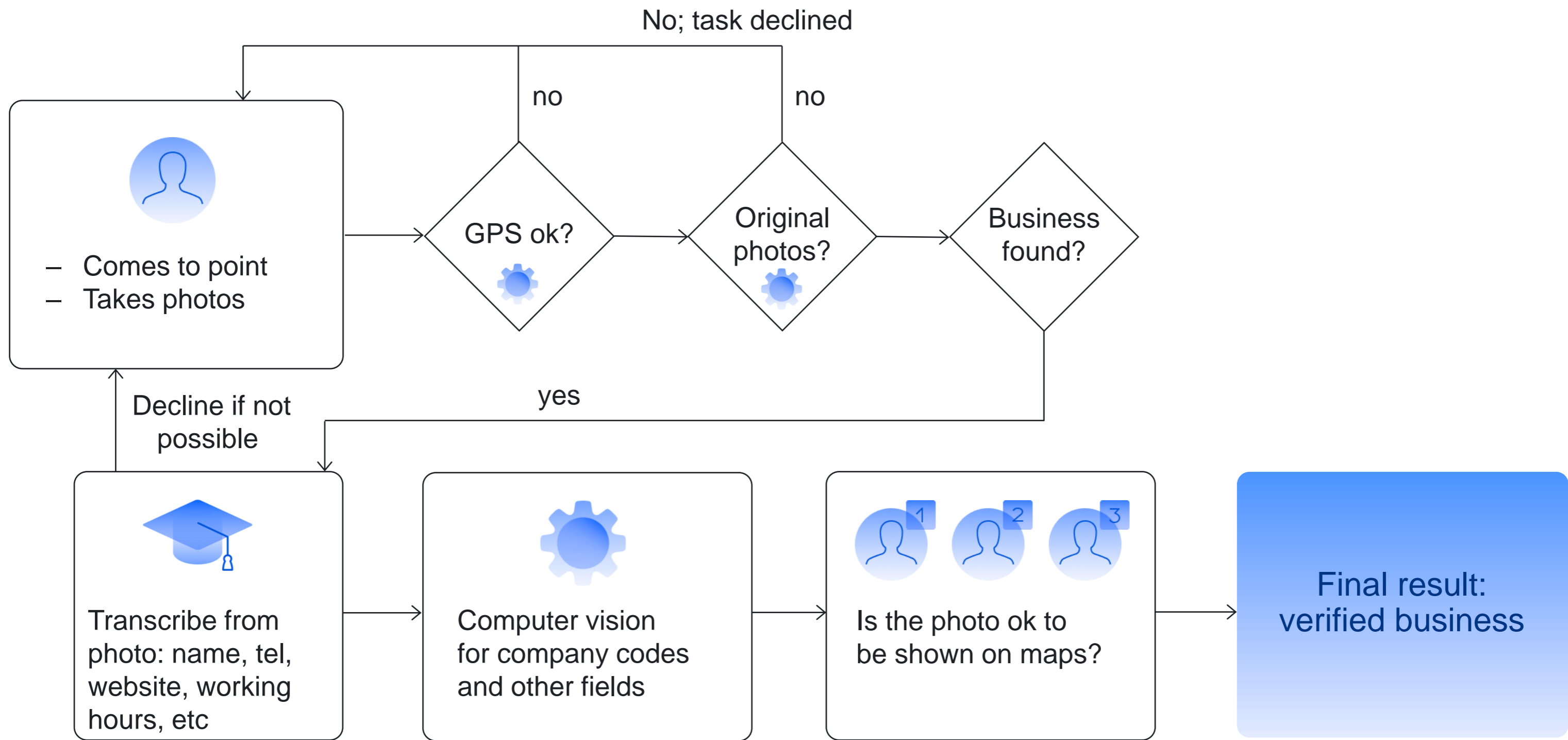
Final result:
verified business



Final result:
verified business



Final result:
verified business



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



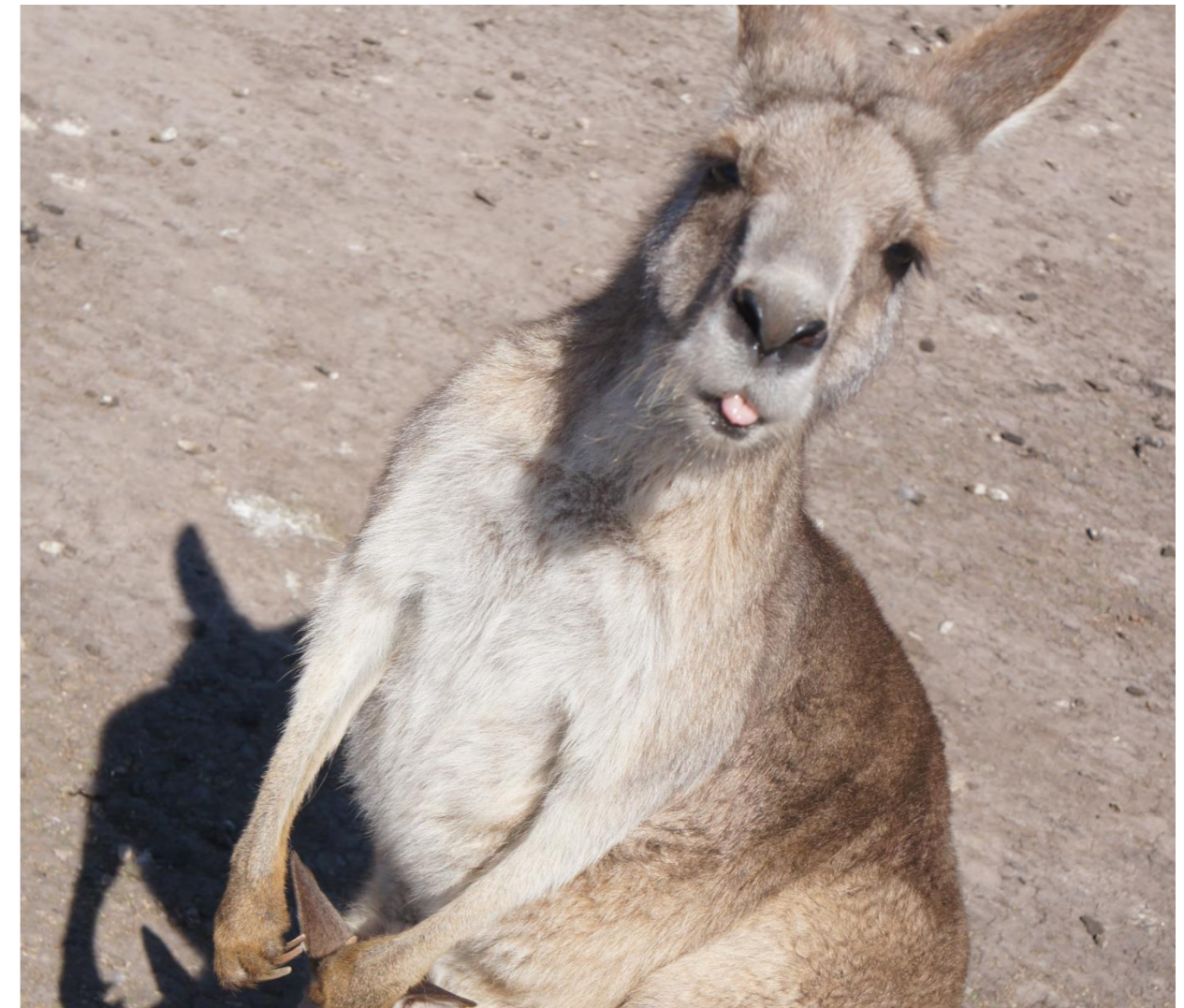
Rare case: many cats

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

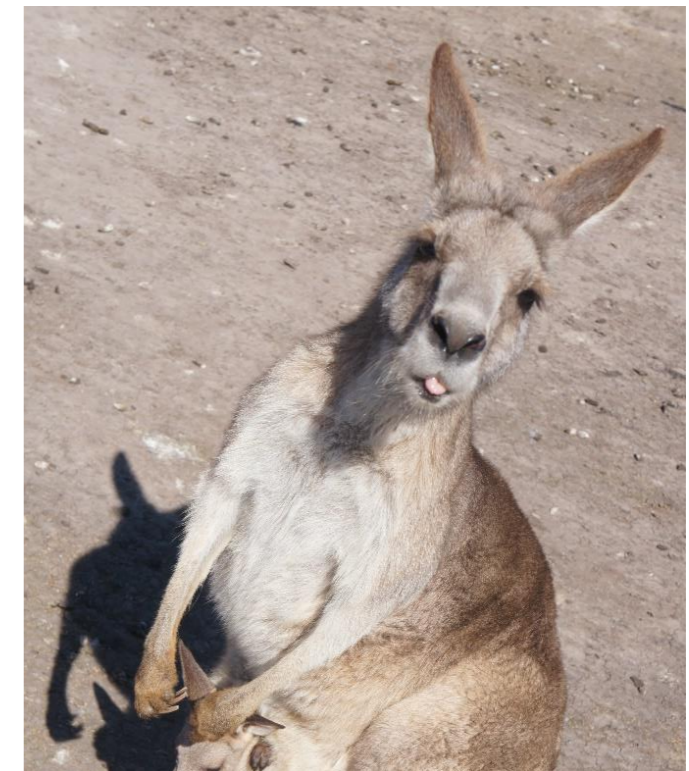
Rare case: image has not been shown

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

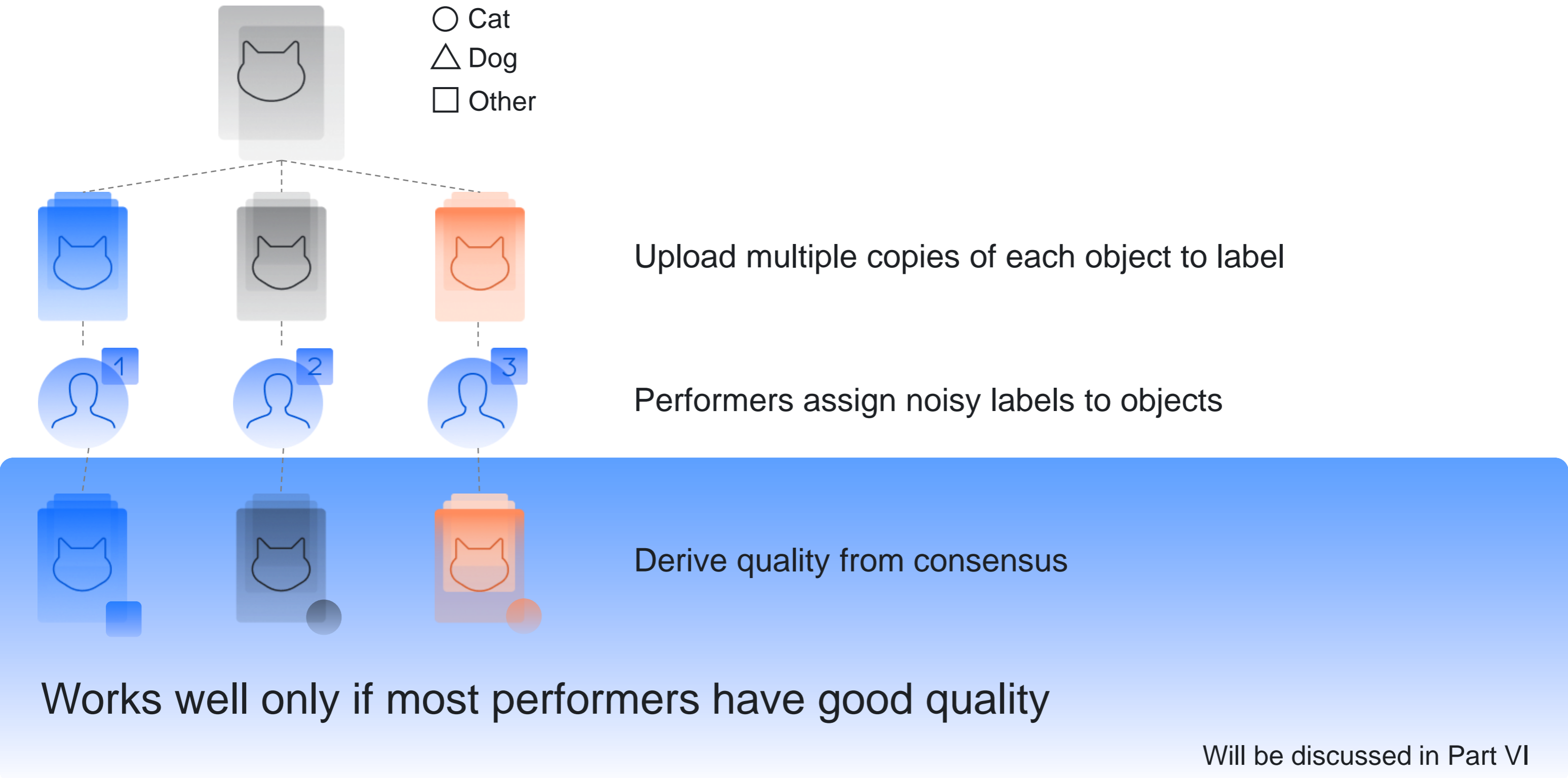
Tricks to remove bots and cheaters

- ▶ 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)

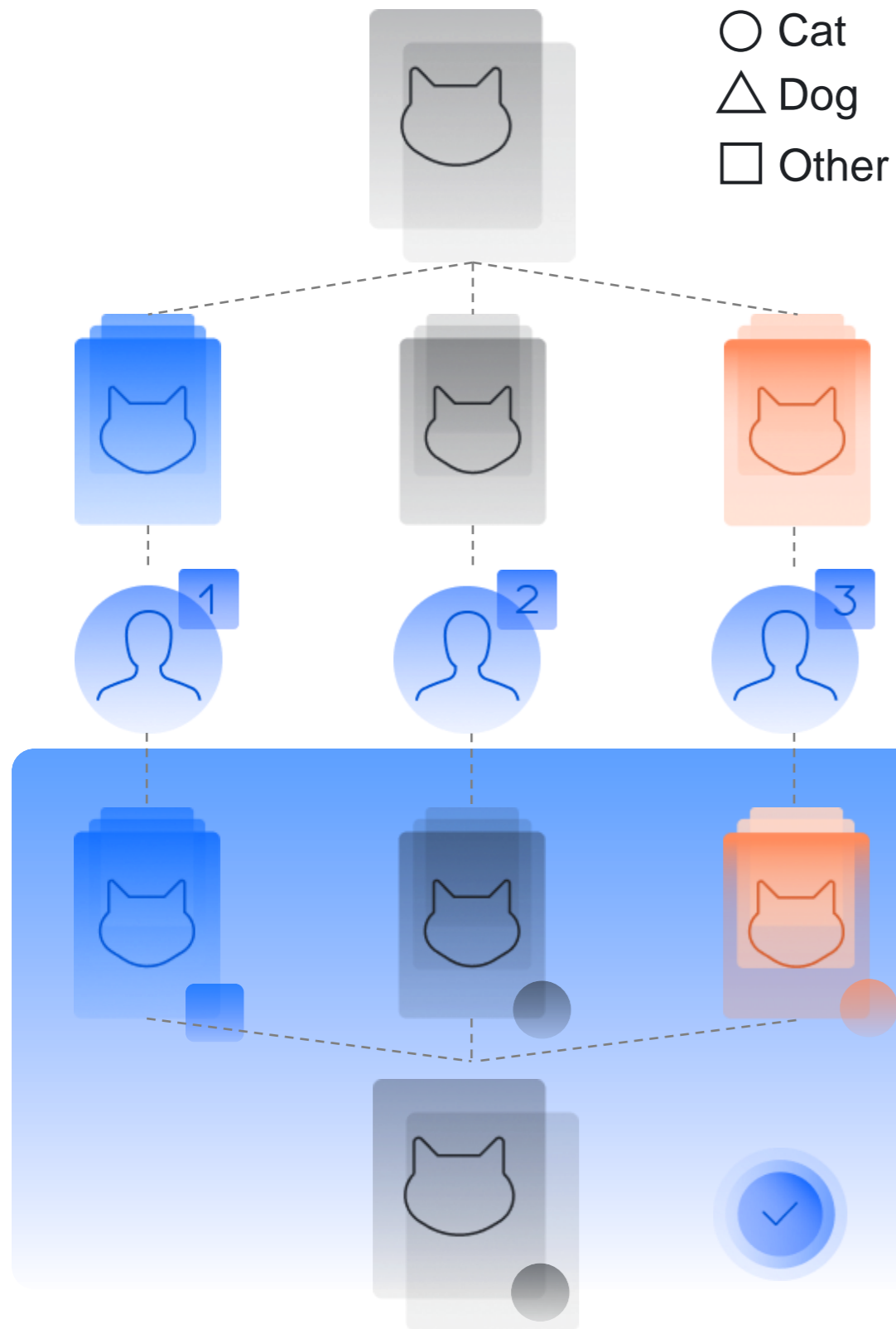
Consensus between performers



Aggregation

Aggregation

- Cat
- △ Dog
- Other



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:

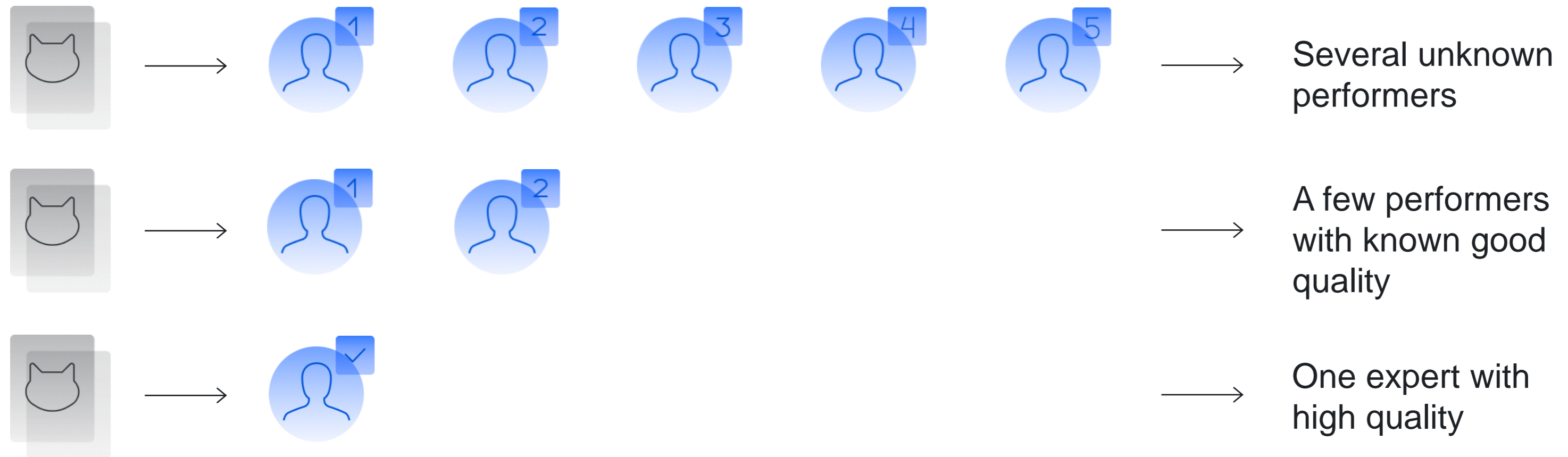
- Assign the most popular answer (Majority Vote)
- There are more sophisticated methods

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

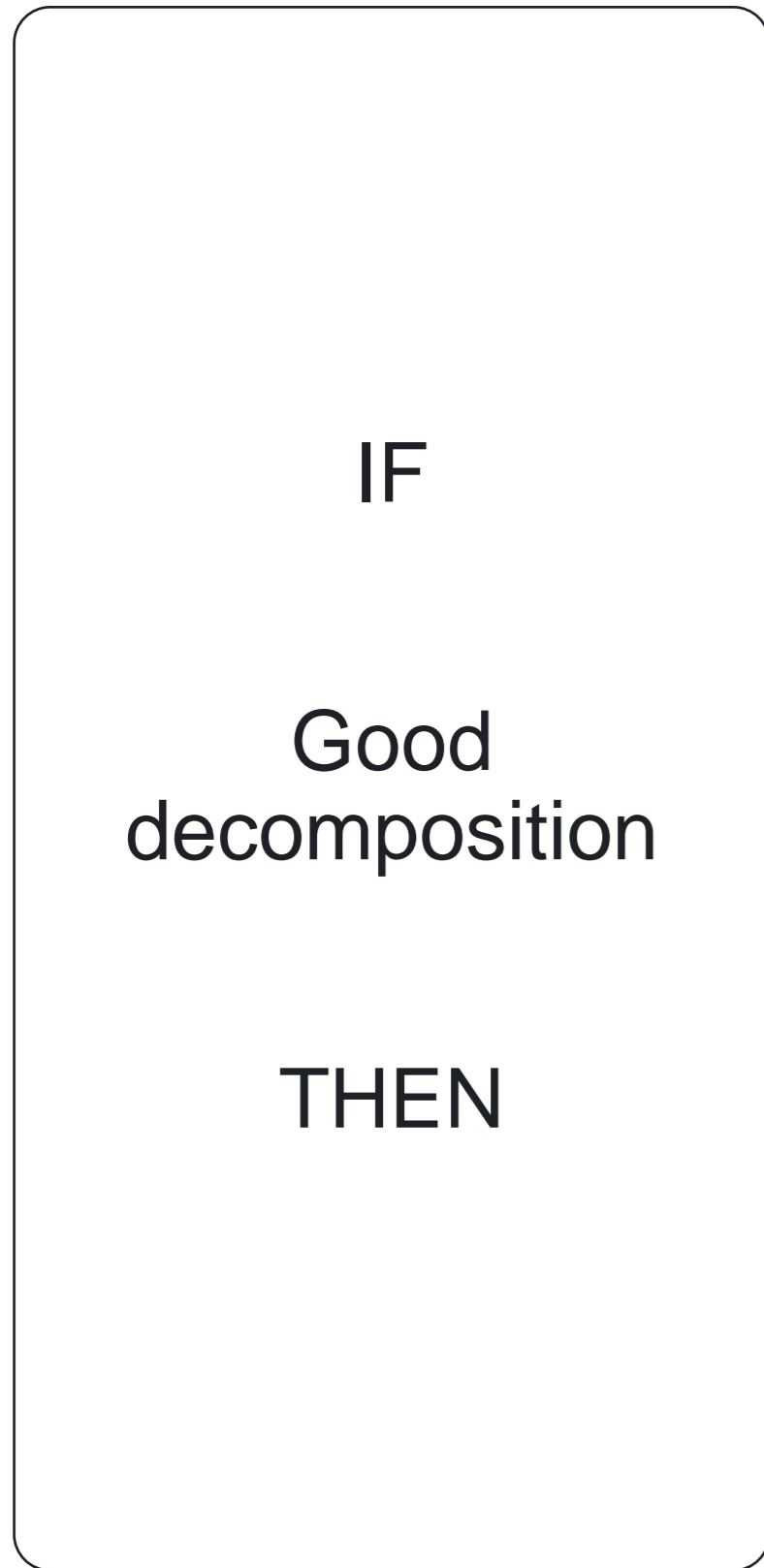
- ▶ The lower supply of performers is (e.g. due to specific skills), the higher price
- ▶ How quickly do you need accomplished tasks (latency)?

Result quality

- ▶ Incentivize better performance by a quality-dependent price

Will be discussed in Part VIII

Good decomposition is the key
to success



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

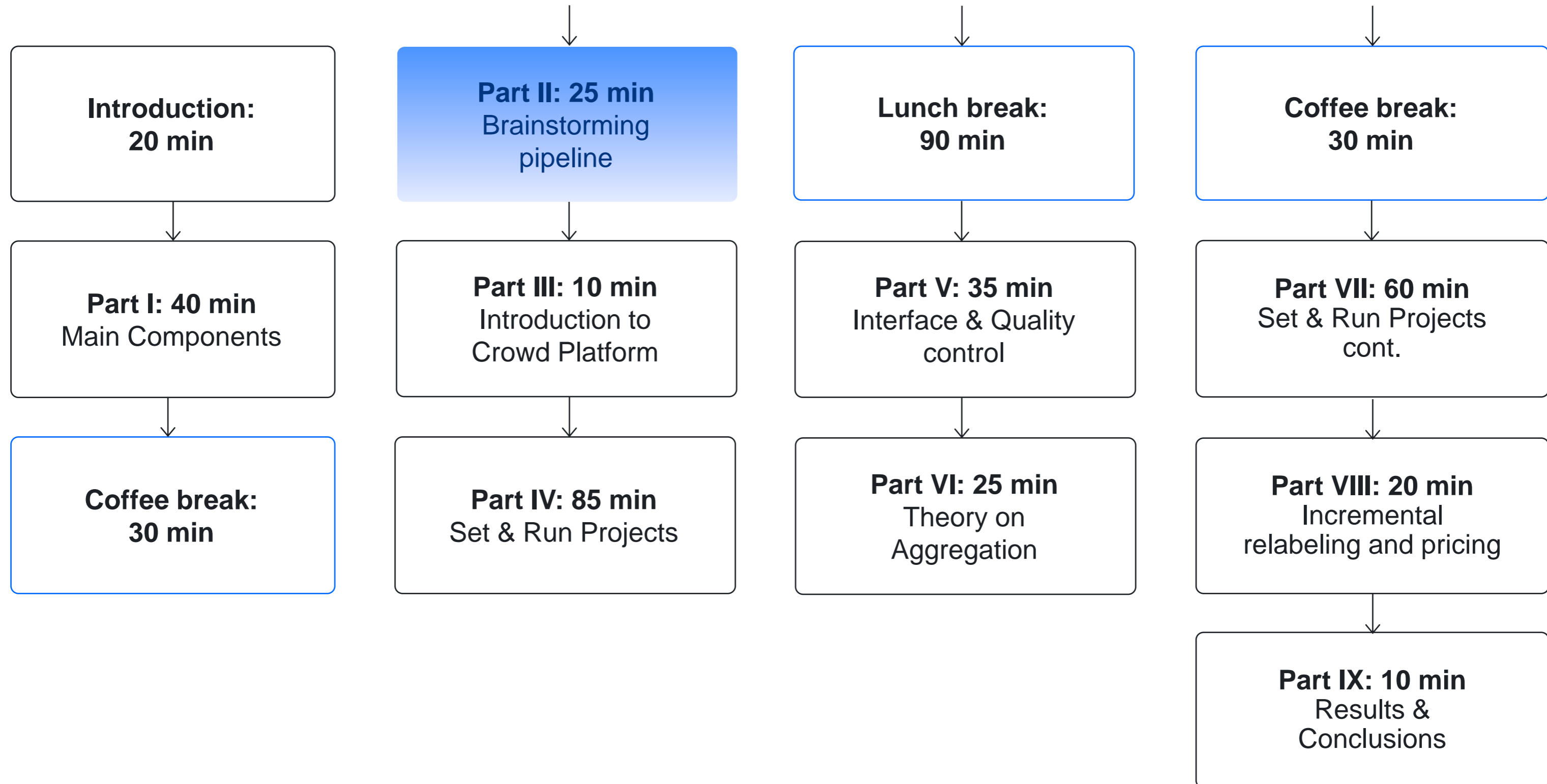
Part II

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

The screenshot shows the ASOS website interface. At the top, there is a navigation bar with the ASOS logo, 'WOMEN' and 'MEN' tabs, a search bar, and icons for user profile, heart, and shopping bag. Below the navigation bar, there are category links: 'New in', 'Clothing', 'Shoes', 'Accessories', 'Activewear', 'Face + Body', 'Living + Gifts', 'Brands', 'Sale' (highlighted in red), 'Marketplace', and 'Inspiration'. A 'UTLET' button is also visible. The breadcrumb trail reads: 'Home > Women > Dresses > Wednesday's Girl midi dress in smudge spot print'. The main product image shows a woman in a blue dress with white polka dots. To the left of the main image are four smaller thumbnail images and a 'VIDEO' button. Below the main image is a 'SHARE' button. A 'SELLING FAST' badge is in the bottom right corner of the main image. To the right of the main image is a 'YOU MIGHT ALSO LIKE' section with four product cards. Each card has a 'heart' icon and the word 'EXCLUSIVE'. Below the cards are navigation arrows and a progress indicator with four dots.

ASOS WOMEN MEN Search for items, brands and inspiration

New in Clothing Shoes Accessories Activewear Face + Body Living + Gifts Brands **Sale** Marketplace Inspiration **UTLET**

Home > Women > Dresses > Wednesday's Girl midi dress in smudge spot print

YOU MIGHT ALSO LIKE


- Wednesday's Girl midi dress in smudge spot print **£22.00**
- Wednesday's Girl mini smock dress in daisy print **£20.00**
- ASOS DESIGN tiered long sleeve smock maxi dress ... **£42.00**
- New Look tiered smock midi dress in multi colour... **£20.50** ~~£25.99~~

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**

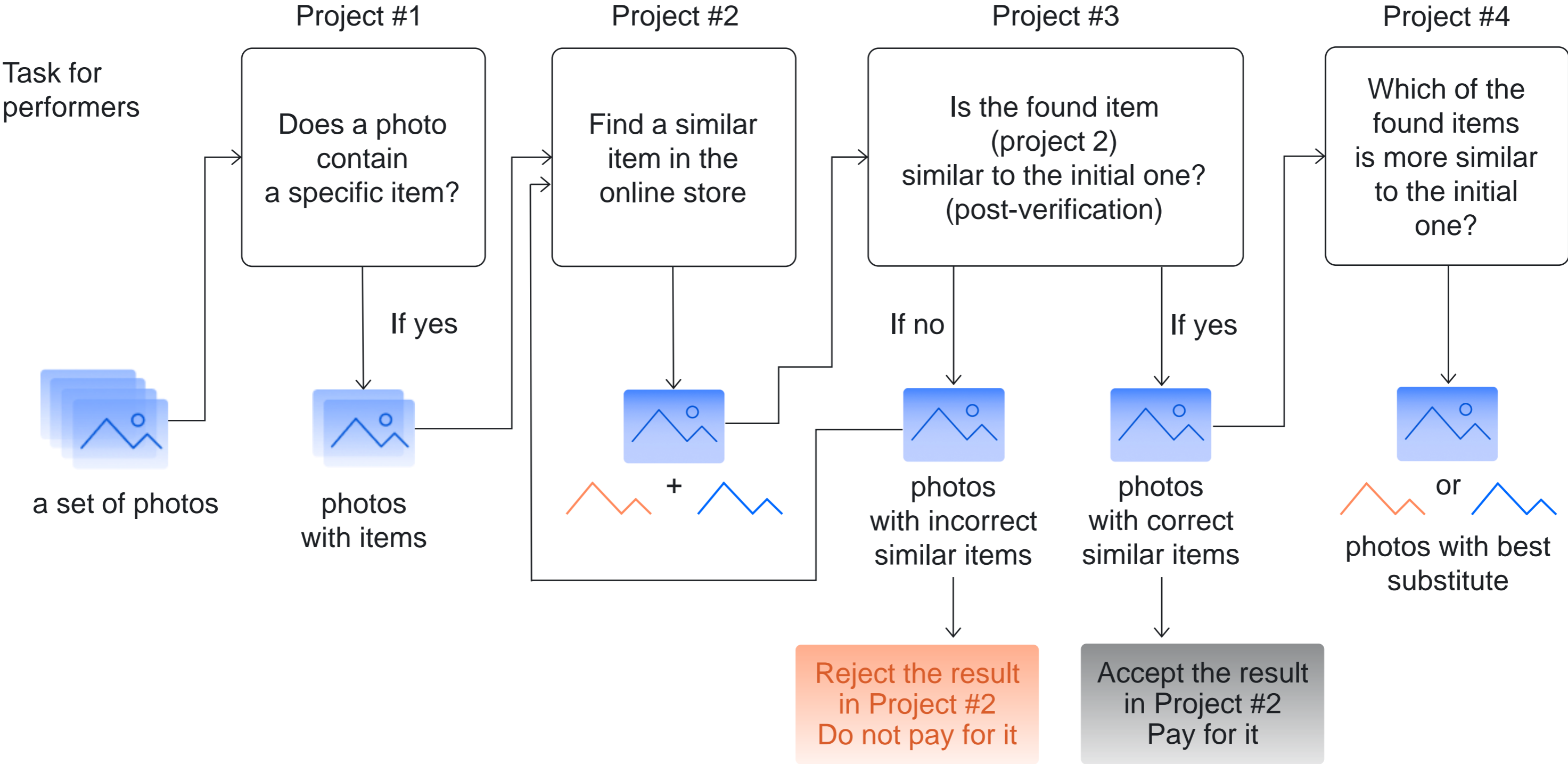
Example: we decide to find the best substitute for the shoes, so our pipeline would be like..

During your practice:

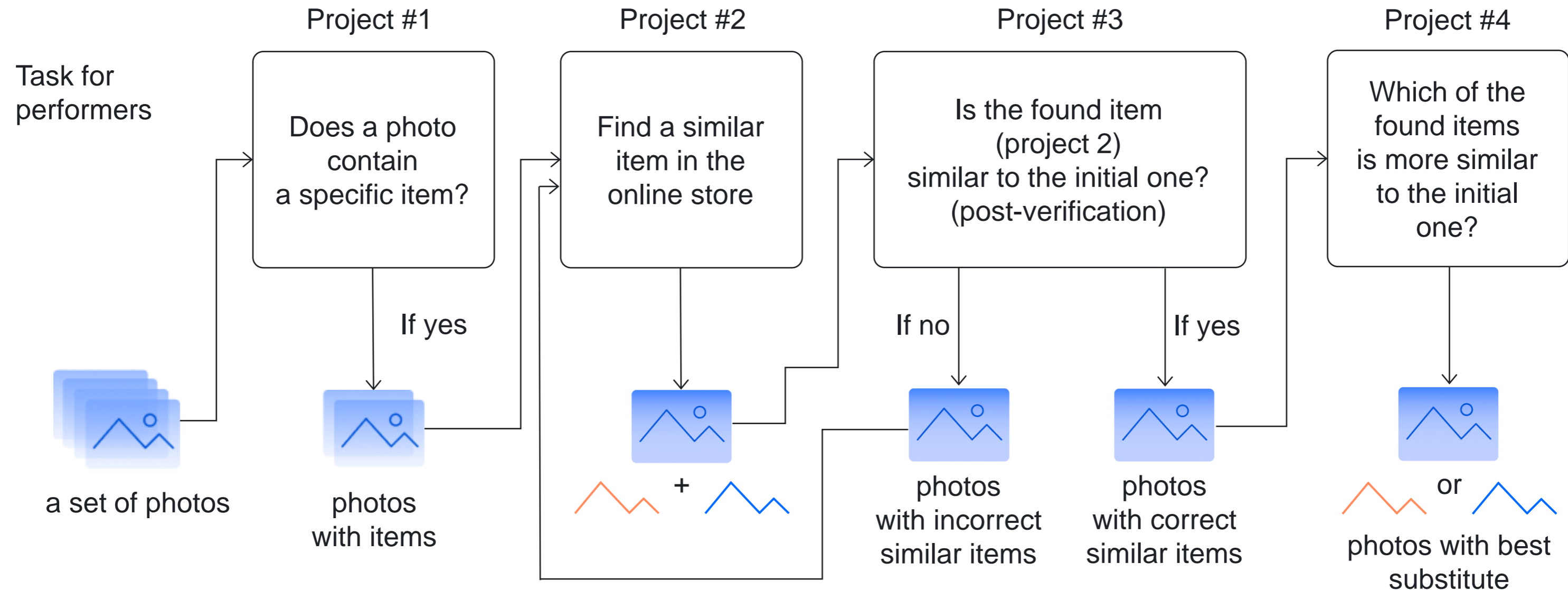
Discuss in groups how you would design a crowd pipeline to find the best substitute!

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



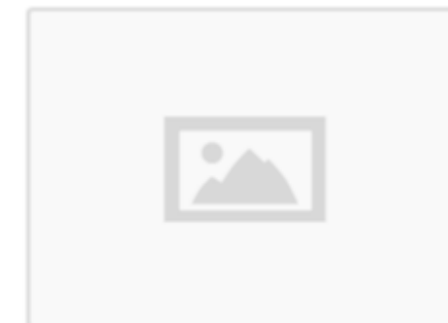
Find the same **shoes** on ASOS

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



Check that the uploaded image matches the product in the store.

[Check the item](#)

Are these **shoes** similar to each other?

Shoes must be the same color and the same style.

Yes No

Project #4: Decide which substitute works best

Task

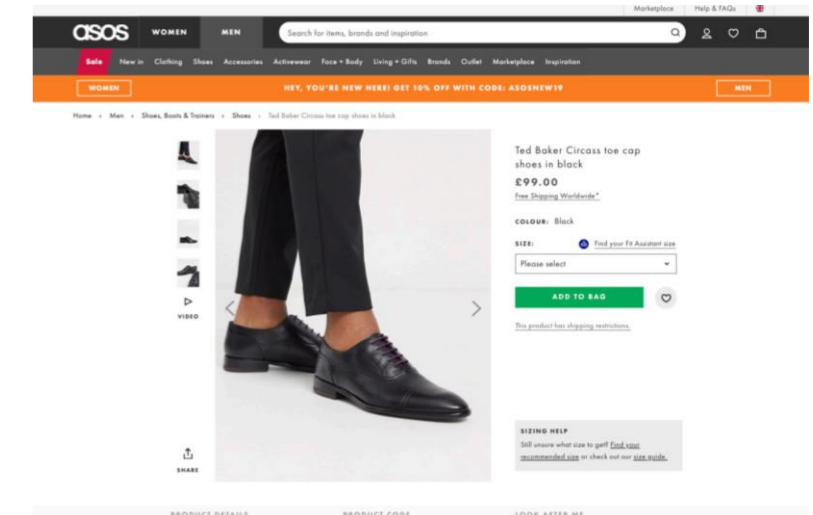
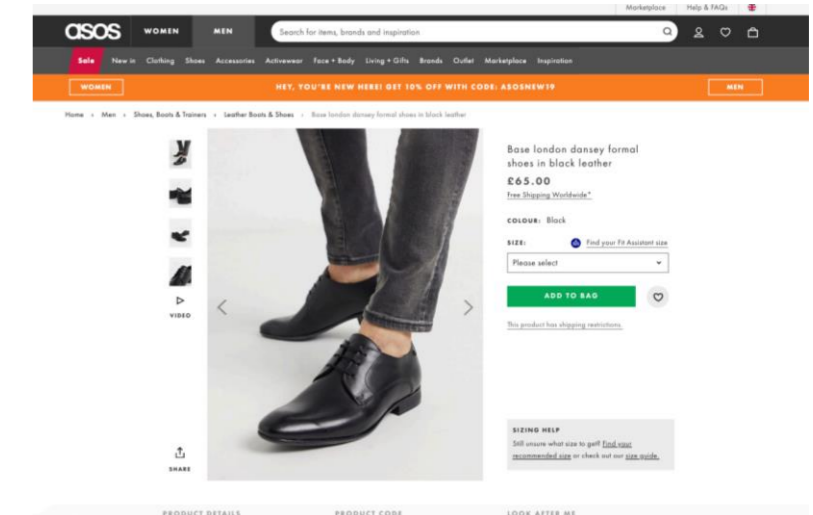
- ▶ Which of the items is similar to the initial one?

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

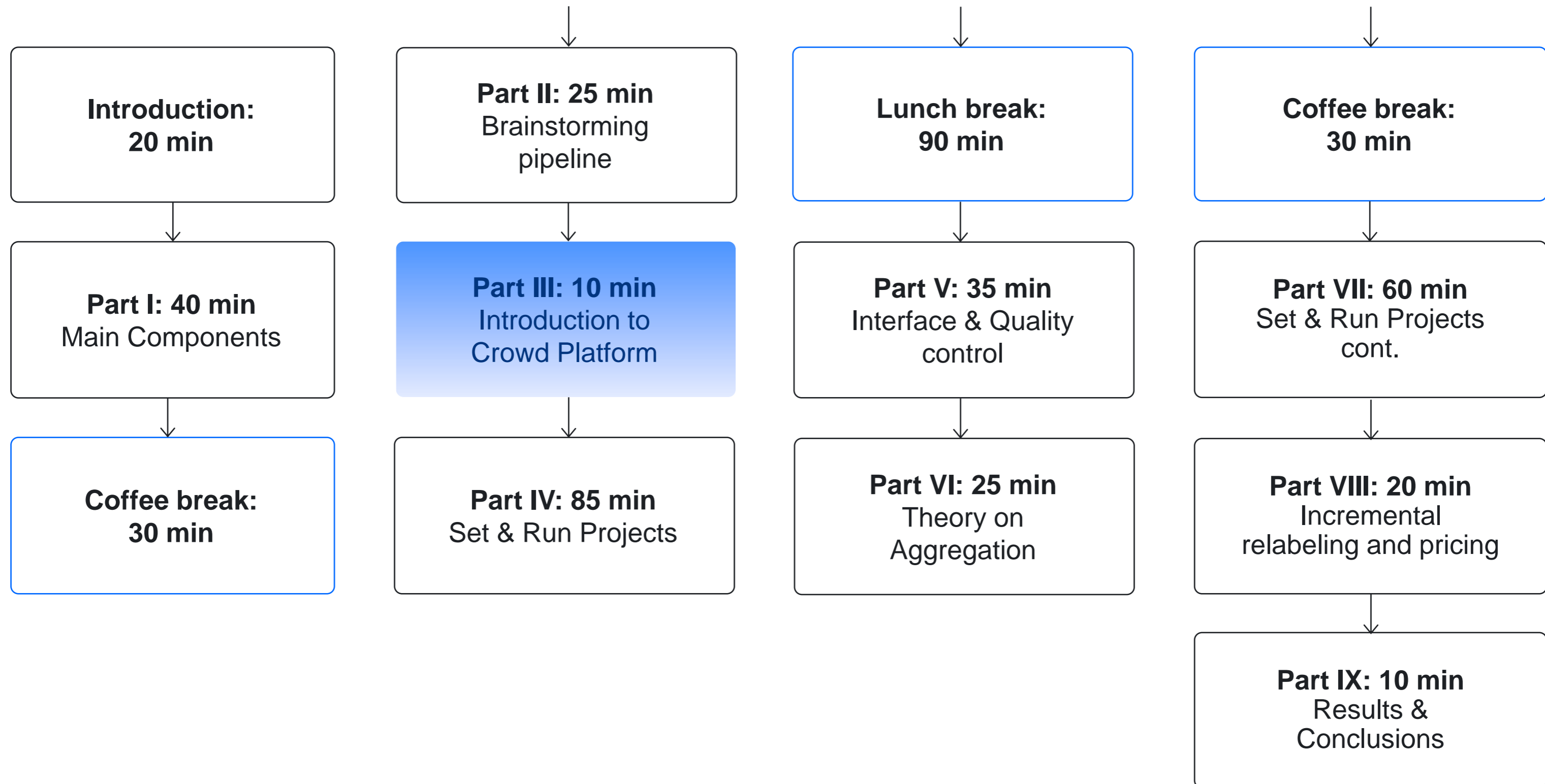


Part III

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

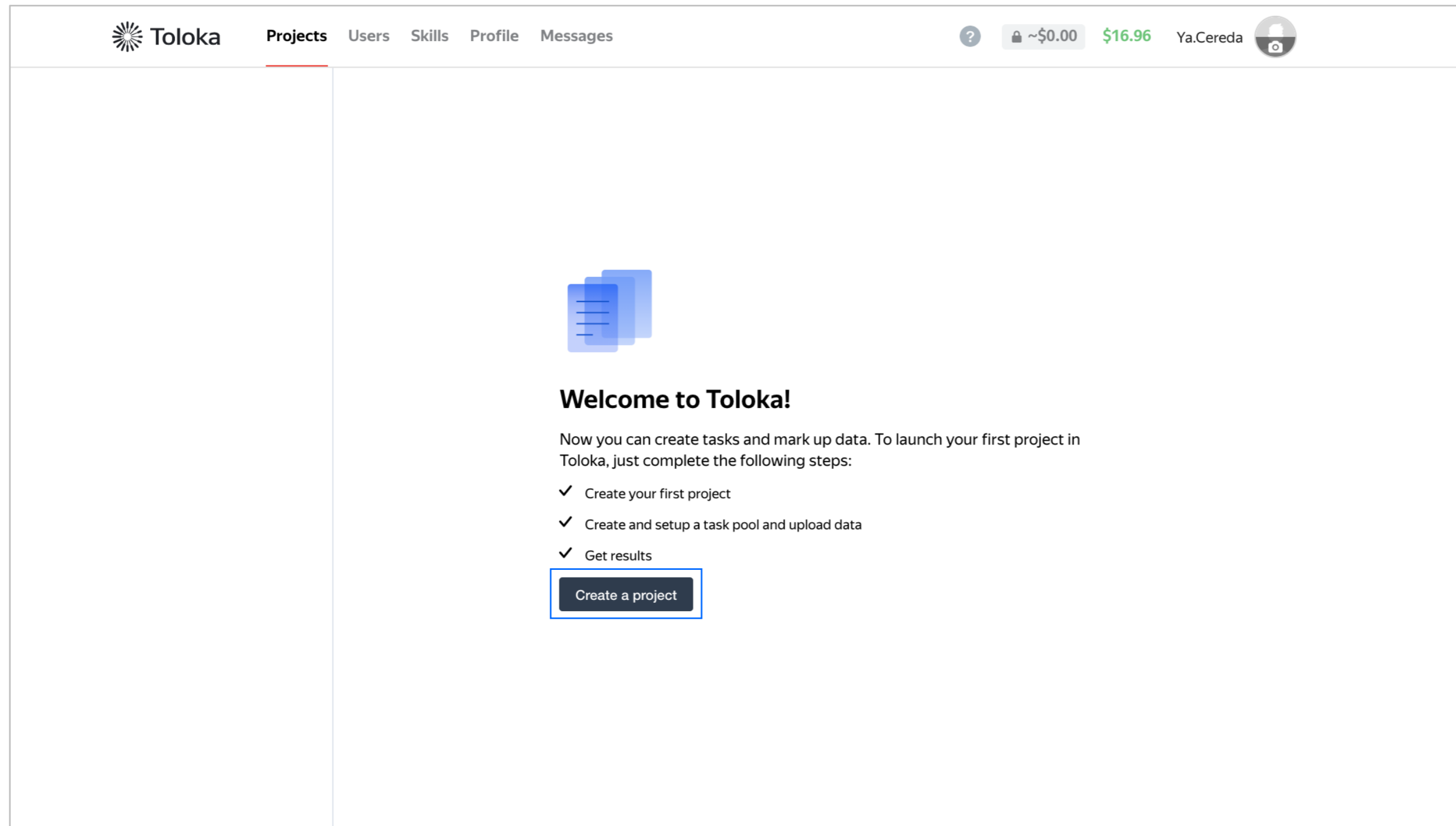
Task



- ▶ A particular input data
- ▶ Results for it from performers

Project: creation & configuration

Project: creation



The screenshot shows the Toloka user interface. At the top left is the Toloka logo. The navigation menu includes 'Projects', 'Users', 'Skills', 'Profile', and 'Messages'. On the right side of the header, there is a help icon, a balance indicator showing '~\$0.00' and '\$16.96', the user name 'Ya.Cereda', and a profile picture icon.

The main content area features a blue icon representing a list or document. Below it, the heading 'Welcome to Toloka!' is displayed. A paragraph of text reads: 'Now you can create tasks and mark up data. To launch your first project in Toloka, just complete the following steps:'. This is followed by a list of three steps, each with a checkmark:

- ✓ Create your first project
- ✓ Create and setup a task pool and upload data
- ✓ Get results

At the bottom of the list, there is a button labeled 'Create a project' which is highlighted with a blue border.

Project: configure name and instruction

The screenshot shows the Toloka 'New project' configuration interface. At the top, the Toloka logo is on the left, and navigation links for 'Projects', 'Users', 'Skills', 'Profile', and 'Messages' are in the center. On the right, there is a user profile for 'Ya.Cereda' with a balance of '\$16.96' and a locked account status of '~\$0.00'. Below the navigation bar, the title 'New project' is displayed on the left, and three buttons ('Back to the old interface', 'Cancel', 'Continue later') are on the right. The main content area is titled '1 General information' and contains two input fields: 'Name to show performers *' with the text 'Does the image contain traffic lights?' and 'Description for performers' with the text 'Select images that contain at least one traffic light.'. Below these fields is a '+ Private comment' link and a 'Save' button. To the right of the input fields is a preview box showing a five-star rating, the project title 'Does the image contain traffic lights?', the instruction 'Select images that contain at least one traffic light.', and pricing information: '0\$ per task' and '~0\$ per hour'.

Project: configure in/out data types

Data specification ?

Input data

image (URL) ●

Add field

Output data <>

Title: result ×

Type: boolean ▾

Allowed values: Any ▾


Required

Array



Delete Save

Add field

Project: configure interface

2 Task interface 

Editor

HTML / JS / CSS  Template builder 

```
1  {{img src=image width="100%" height="400px"}}
2
3  <div>Are there traffic lights in the picture?<div>
4  {{field type="radio" name="result" value="ok" label="Yes" hotkey="1"}}
5  {{field type="radio" name="result" value="bad" label="No" hotkey="2"}}
6  {{field type="radio" name="result" value="404" label="Failed to load" hotkey="3"}}
7
```

HTML


```
1  exports.Task = extend(TolokaHandlebarsTask, function (options) {
2  |  TolokaHandlebarsTask.call(this, options);
3
```

JS



```
1
```


CSS

Project: use the task preview to see how it works

2 Task interface 

Editor

HTML / JS / CSS  Template builder 



Are there traffic lights in the picture?

1 Yes 2 No 3 Failed to load

Project: saving

Edit project

[Back to the old interface](#)

- ✓ General information
- ✓ Task interface
- ✓ Instructions for performers
- ✓ Translations

Pool: creation & configuration

Pool: creation

Are there shoes in the picture? — active

Project actions ▾

Statistics for 7 days

Submitted tasks	Spent	Quality: control tasks	Quality: training tasks	Average submit time	Users	Banned users
0	0 \$	-	-	-	0	0

Pools Training Statistics Quality control

Active and closed Archived Filters Search

Add a pool

Pools can be archived manually or automatically (automatic archiving applies to pools with no activity for 30 days).

Title ↕	Priority ↕	Progress	Status ↕	Started ↕	To be completed
---------	------------	----------	----------	-----------	-----------------

To launch a project, you first need to add a pool, set user filters and quality control rules, and upload tasks.

50 ▾

Pool: configure name and description

POOL NAME (VISIBLE ONLY TO YOU) ?

Use project description

PUBLIC DESCRIPTION ?

Add a private description

Pool: configure pricing and overlap

Price per task suite

Each task suite can have one or multiple tasks on the same page. Enter the total price for all tasks in the suite.

PRICE IN US DOLLARS ?

0.01



FEE ?

0.005

+ Dynamic pricing

Overlap

Specify how many performers you want to complete each task in the pool.

OVERLAP ?

3



DYNAMIC OVERLAP ?

Off

Pool: configure timing and post verification

Parameters

TIME PER TASK SUITE IN SECONDS ?	<input type="text" value="300"/>	POOL CLOSING DATE ?	<input type="text" value="2022-08-24"/>
KEEP TASK ORDER ?	<input type="checkbox"/> No	WAITING TIME FOR THE POOL TO CLOSE IN SECONDS ?	<input type="text" value="0"/>
		POOL PRIORITY WITHIN THE PROJECT ?	<input type="text" value="0"/>

Pool: filter performers by their profiles

Performers

[Copy settings from...](#)

Filter performers who can access the task.
Toloka has users from different countries,
so don't forget to filter by language and region. [Learn more](#)

ADULT CONTENT ? Yes

Add filter ▼ Create a skill

PERFORMER PROFILE

Languages ▼ = English × 🗑️ +

Pool: filter performers by computed properties

COMPUTED

Browser = YANDEX_BROWSER

AND

COMPUTED

Client = Toloka web version

Pool: filter performers by custom skills

The image shows a user interface for filtering performers. At the top, there is a light gray bar containing two buttons: "Add filter" with a downward arrow and "Create skill". Below this bar is a section titled "SKILLS" in a light gray background. Under "SKILLS", there is a filter rule: a text box containing "Segmentation" with a close icon (X), followed by a comparison operator ">" and a value "70" in a text box with a close icon (X). To the right of the value box are two icons: a trash can and a plus sign (+).

Pool: control quality via a rule on golden set

GOLDEN SET ?

History size

If

and

then

Pool: control quality via a rule on majority vote

MAJORITY VOTE ?

Accept as majority

History size


If

and


then

Pool: control quality via a rule on post verification

OFFLINE ACCEPTANCE ?

History size 


If

and 




then

Task uploading & golden set creation

Task: uploading

 Does the image contain traffic lights? — closed Statistics Download results Edit


Download the sample file, add your task data, and upload the file to the pool.
The sample file uses TSV format, which is the same as CSV but using tab as the separator.
Make sure you choose UTF-8 encoding when saving the file. [Learn more in the guide.](#)

-  [Template for general tasks.tsv](#)
-  [Template for control tasks.tsv](#)
-  [Template for training tasks.tsv](#)

Upload

0 task pages	0 training tasks
0 tasks	0 control tasks


0 %
Completed 0




Task: uploading

File upload settings ?


Tasks per page



By empty row



Set manually



Smart mixing

Main tasks


Training tasks

Control tasks

[Show advanced settings](#)

[Sample file for uploading tasks](#) Close Upload





Task: edit for creation of control tasks

 Does the image contain traffic lights? — closed

Statistics Download results Edit

Download the sample file, add your task data, and upload the file to the pool.
The sample file uses TSV format, which is the same as CSV but using tab as the separator.
Make sure you choose UTF-8 encoding when saving the file. [Learn more in the guide.](#)

- [Template for general tasks.tsv](#)
- [Template for control tasks.tsv](#)
- [Template for training tasks.tsv](#)

 Upload	 Files	 Delete	 Edit
0 task pages	0 training tasks		
100 tasks	0 control tasks		

0 %
Completed 0

0 0

Task: control task creation

Edit tasks

Use main tasks as a starting point to create control tasks or training tasks.

Control tasks are for checking the quality of responses from performers. They contain correct responses to compare with actual responses.

Training tasks are for teaching performers how to complete tasks. They contain correct responses and hints.

[Learn more](#)

Main 100 Control tasks 0 Training tasks 0

Create control tasks

Create training tasks

Download

ID ↕	Overlap ↕	Responses from performers ↕	Last updated ↕
...bdb34908	3	0	08/24/2021 4:23:52 PM

Task: create a control task by answer selection

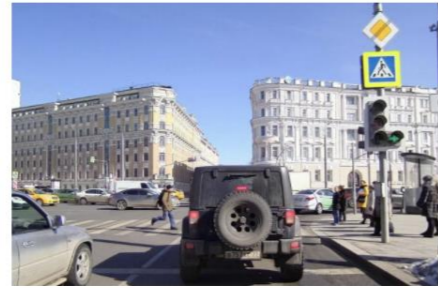
Toloka **Projects** Users Skills Profile Messages ? ~\$0.00 \$16.96 Ya.Cereda

Projects > Does the image contain traffic l... > Does the image contain traffic l... > Uploaded tasks > Edit tasks < >

Create control task

1. Enter correct responses
2. Select the fields to use

Field Value
 result



Are there traffic lights in the picture?
1 Yes 2 No 3 Failed to load


Save and go to next

Distribution of correct responses for control tasks ?

Create control tasks to see charts of response distribution.

**Run the pool &
result downloading**

Pool running

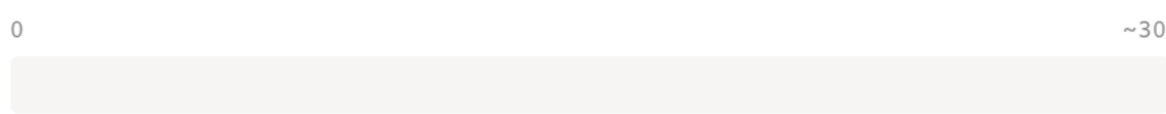
 Does the image contain traffic lights? — closed Statistics Download results Edit

Download the sample file, add your task data, and upload the file to the pool.
The sample file uses TSV format, which is the same as CSV but using tab as the separator.
Make sure you choose UTF-8 encoding when saving the file. [Learn more in the guide.](#)


- Template for general tasks.tsv
- Template for control tasks.tsv
- Template for training tasks.tsv


Upload Files Delete Edit Preview	
~30 task pages	0 training tasks
90 tasks	10 control tasks

0 %
Completed 0






View and aggregate the results of your tasks




 Does the image contain traffic lights? — open

Statistics |  Download results | ^ | Edit | v

View operations
Dawid-Skene aggregation model
Aggregation by skill


Download the sample file, add your task data, and upload the file to the pool.
The sample file uses TSV format, which is the same as CSV but using tab as the separator.
Make sure you choose UTF-8 encoding when saving the file. [Learn more in the guide.](#)

-  [Template for general tasks.tsv](#)
-  [Template for control tasks.tsv](#)
-  [Template for training tasks.tsv](#)


 Upload |  Files |  Preview

30 task pages	0 training tasks
90 tasks	10 control tasks

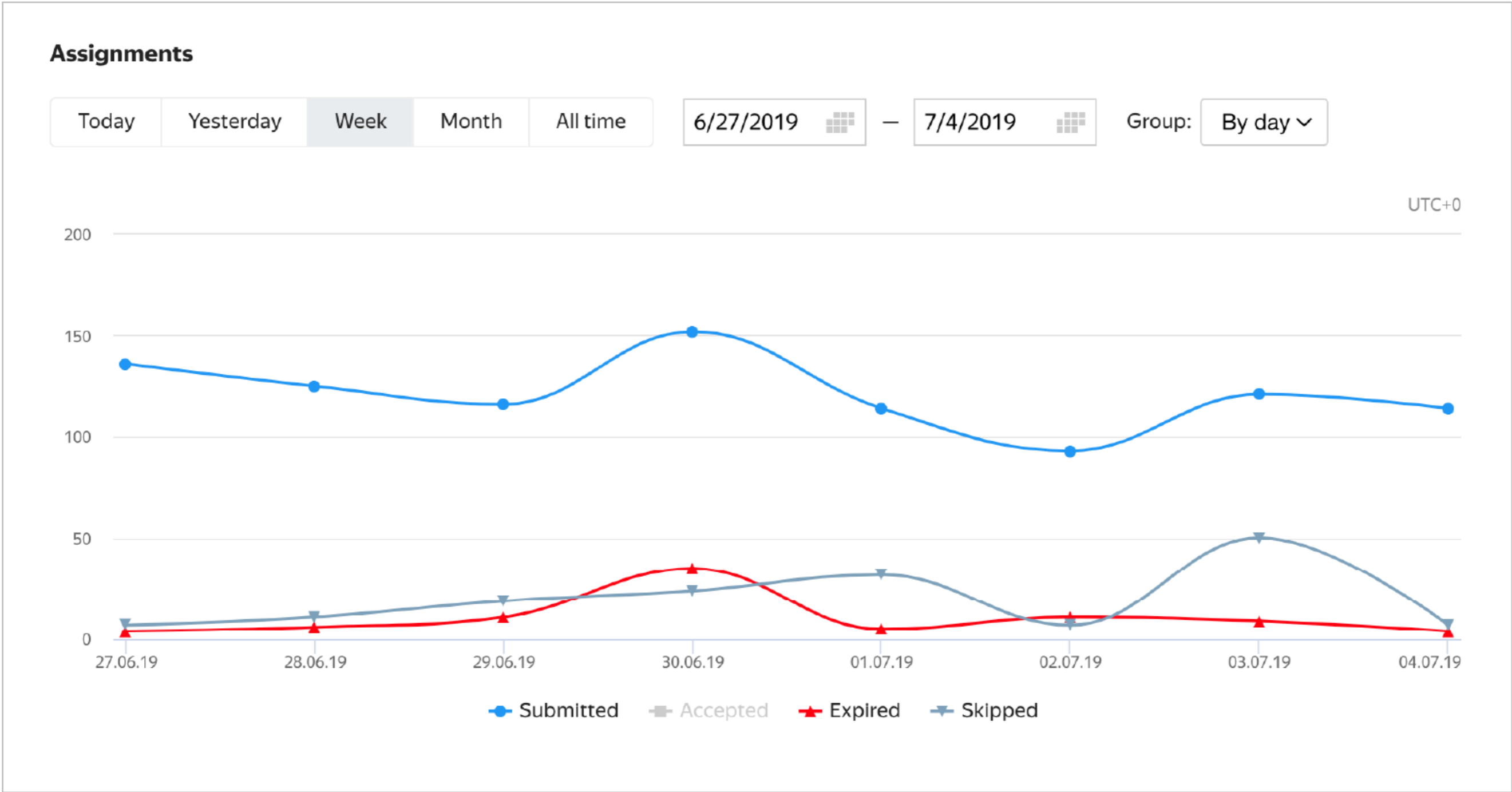
100 %
Completed 30, accepted 30

 View assignments

0 30



Monitor the statistics on how tasks are performed

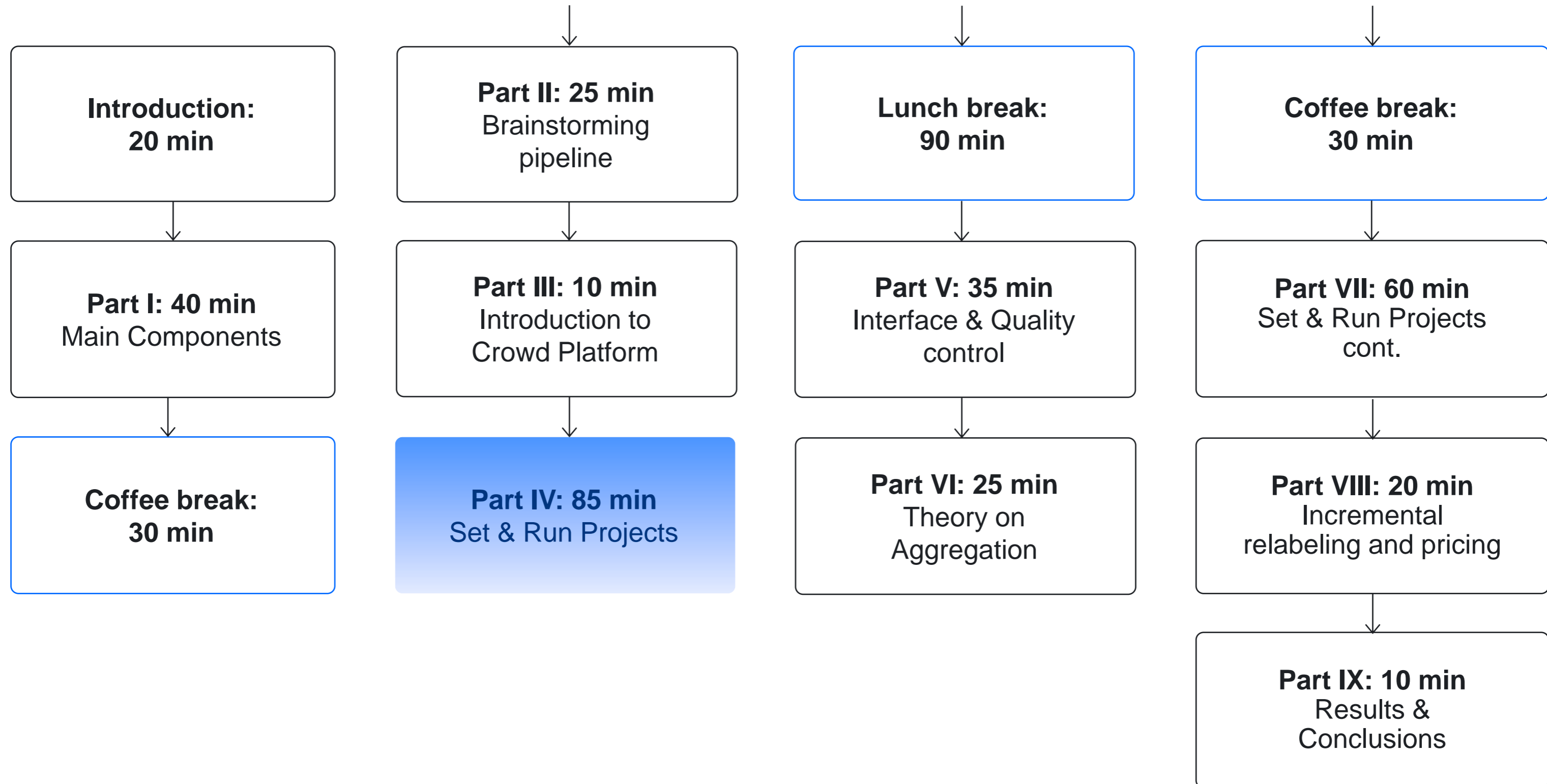


Part IV

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

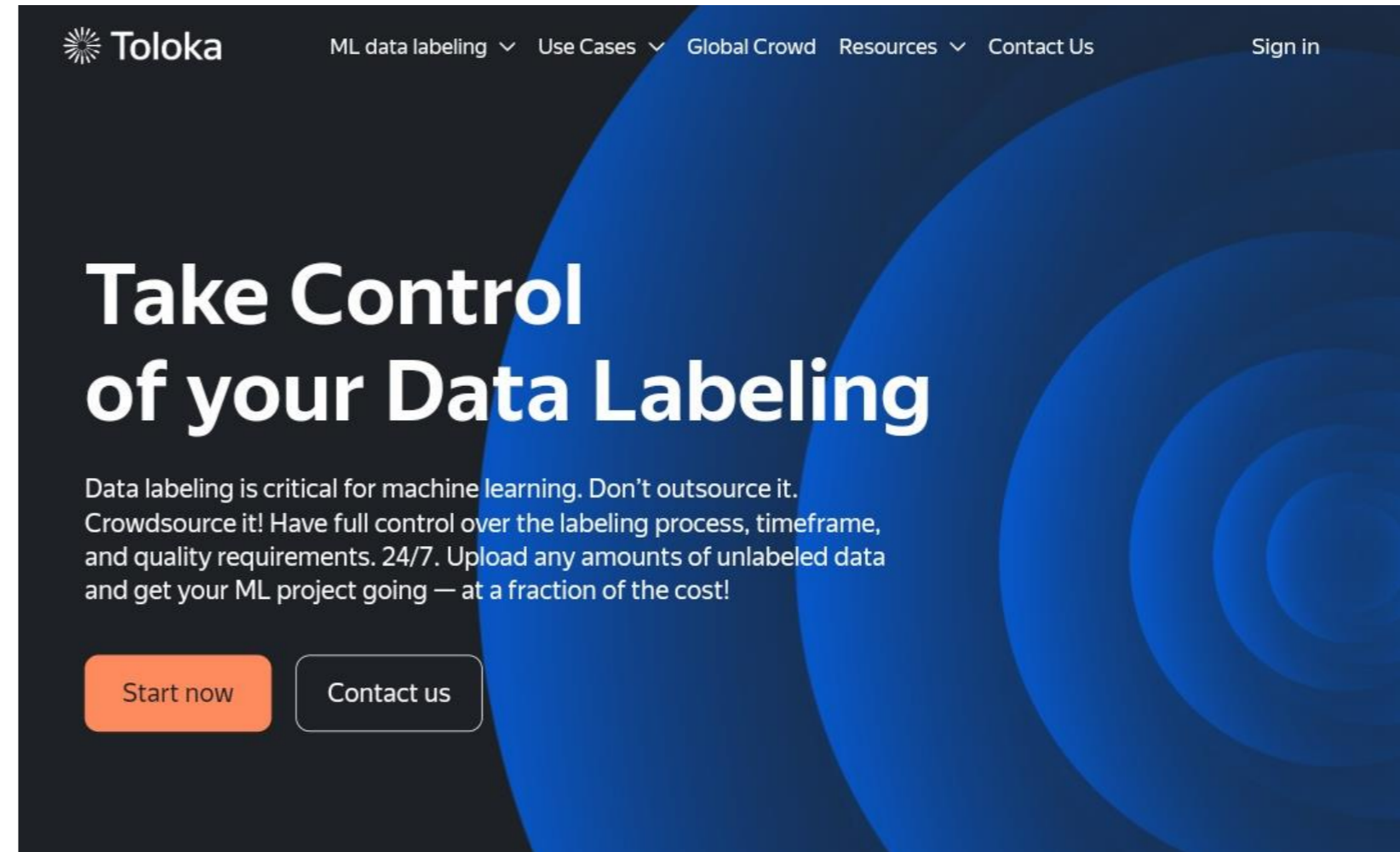
You have Login+Password to sign in Toloka as a requester

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!

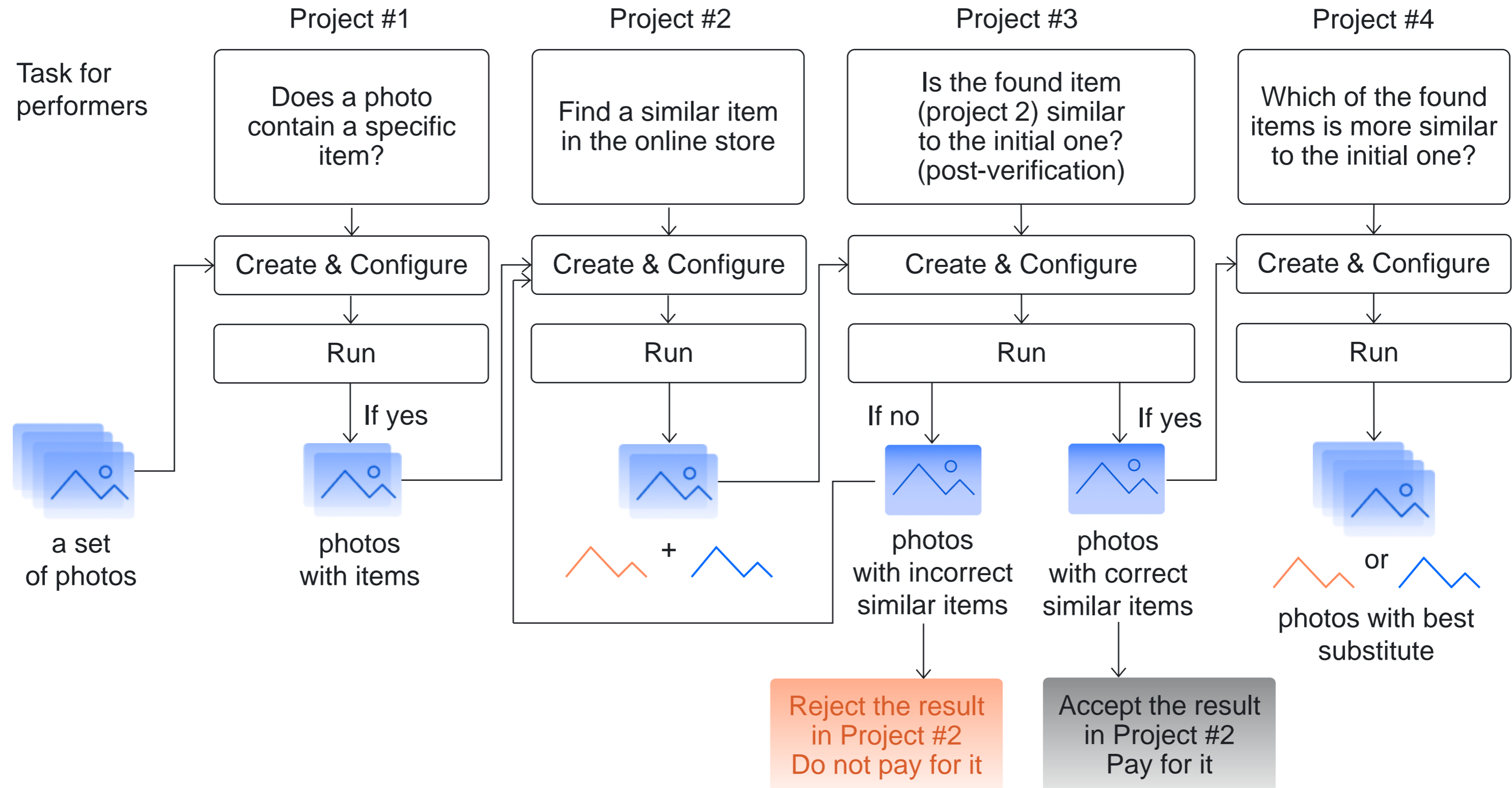
Practice: creating a real crowdsourcing pipeline

Now we will create a real simplified crowdsourcing pipeline

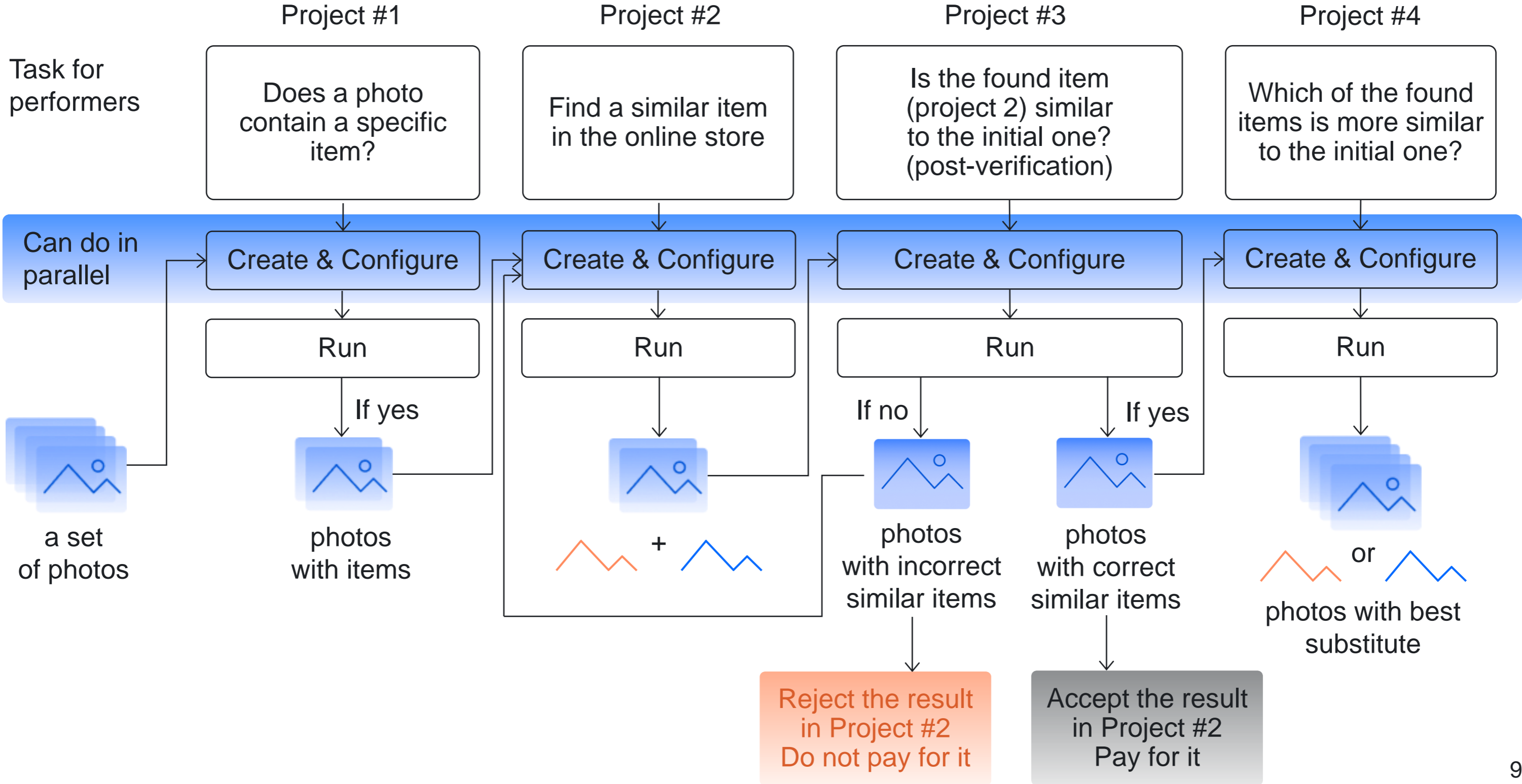
To simplify the task, we ask you to:

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

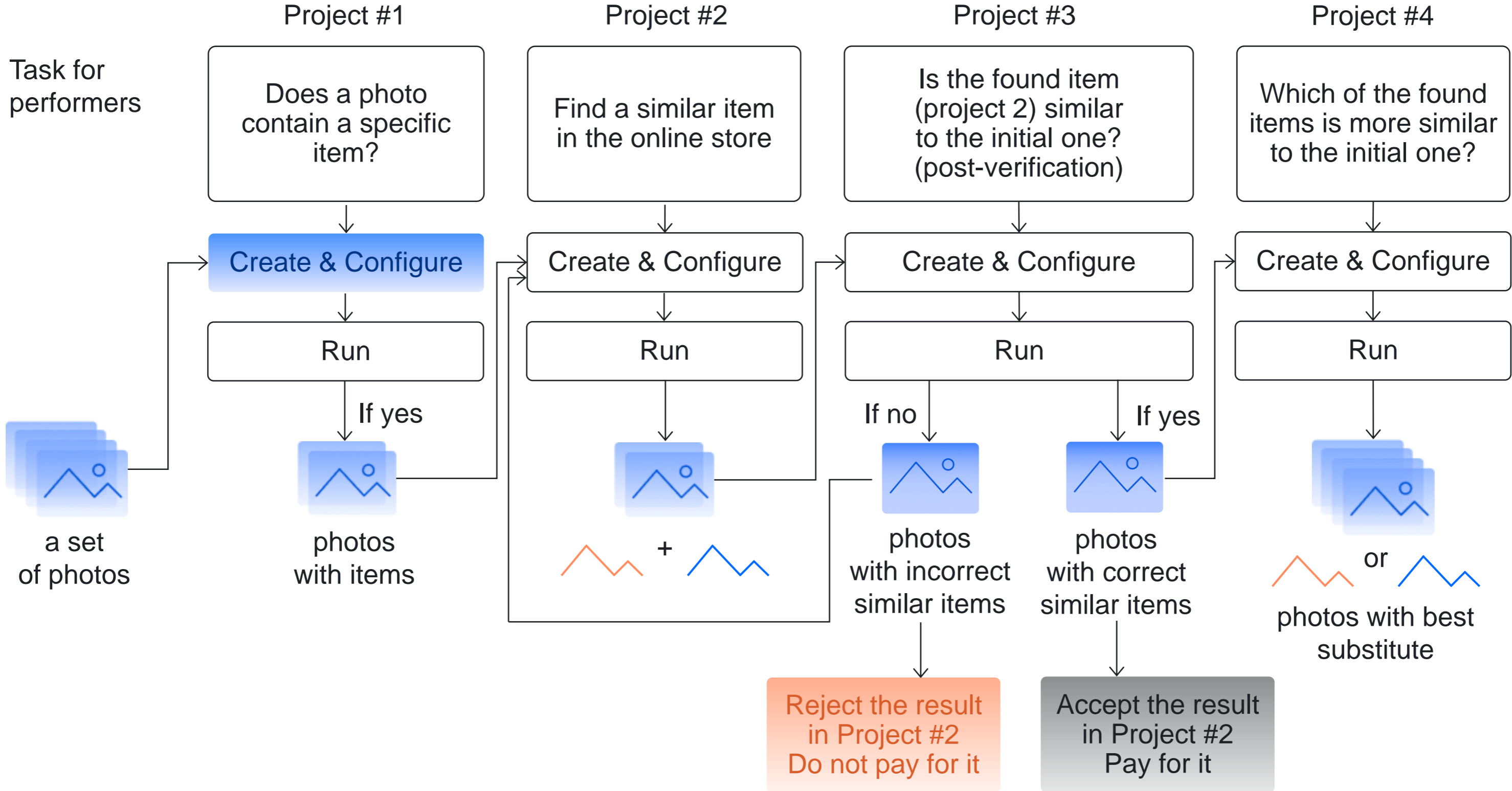
Reminder: we implement and run our pipeline



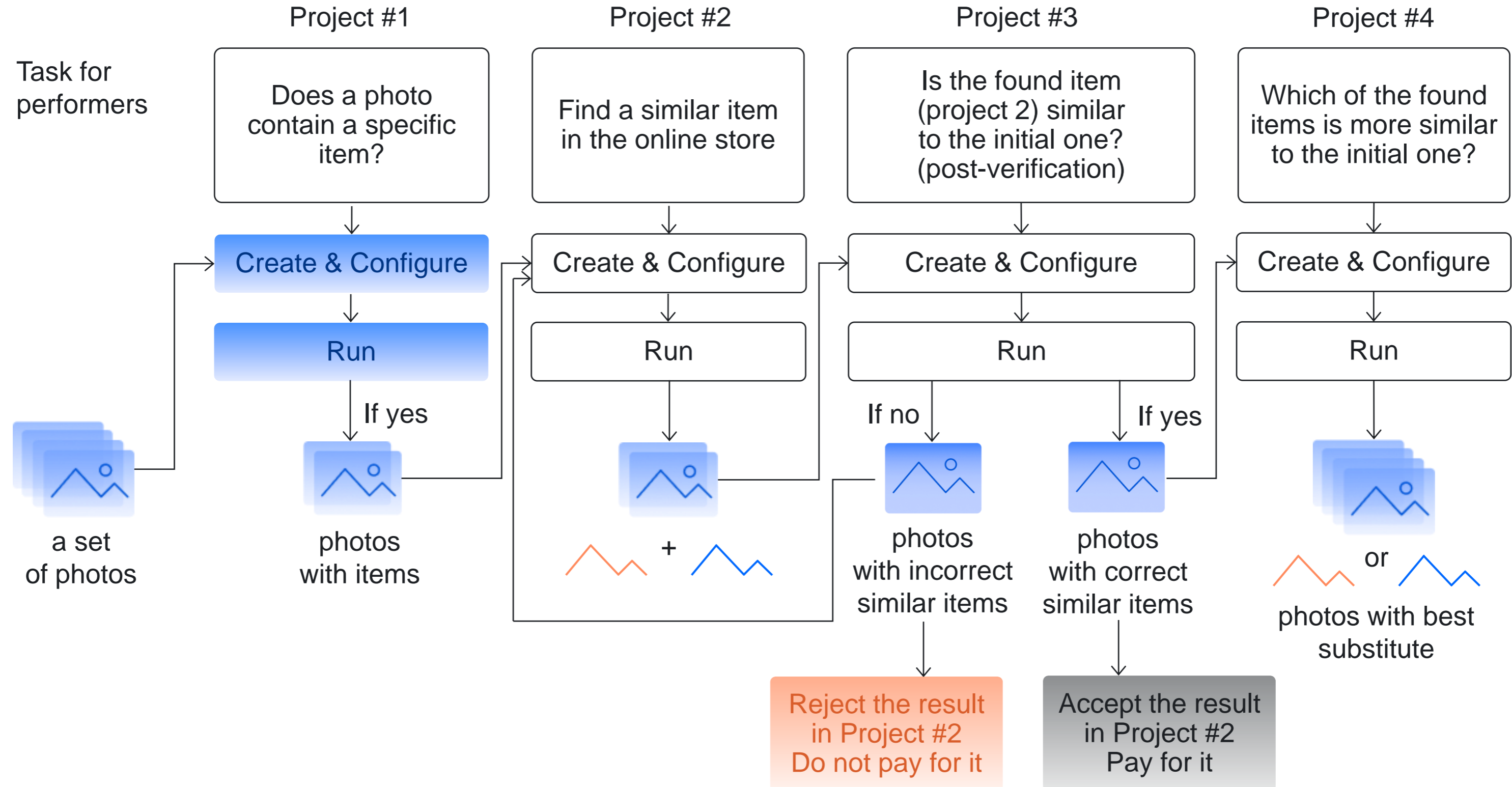
You can divide work within a participant group



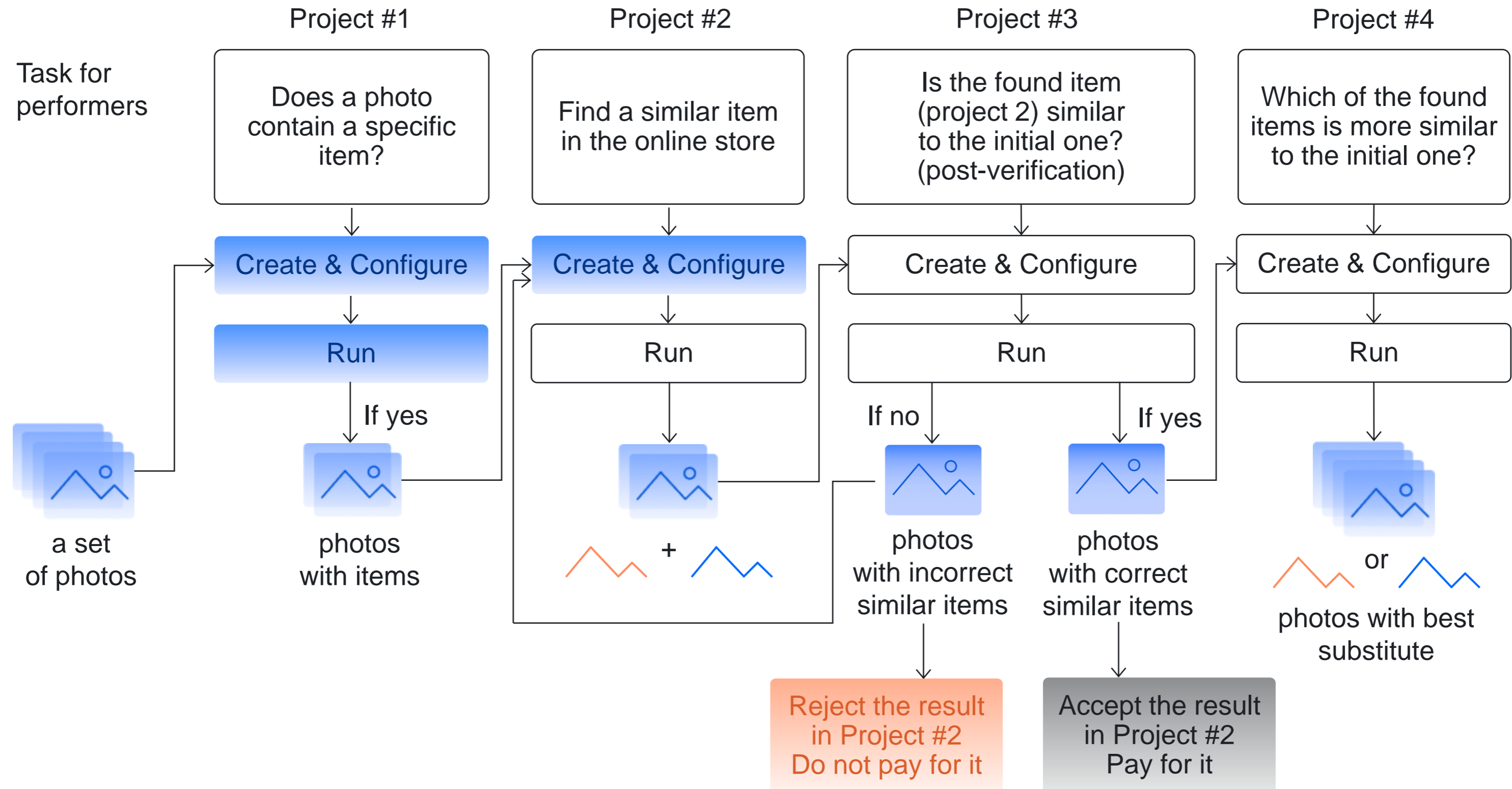
Step #1



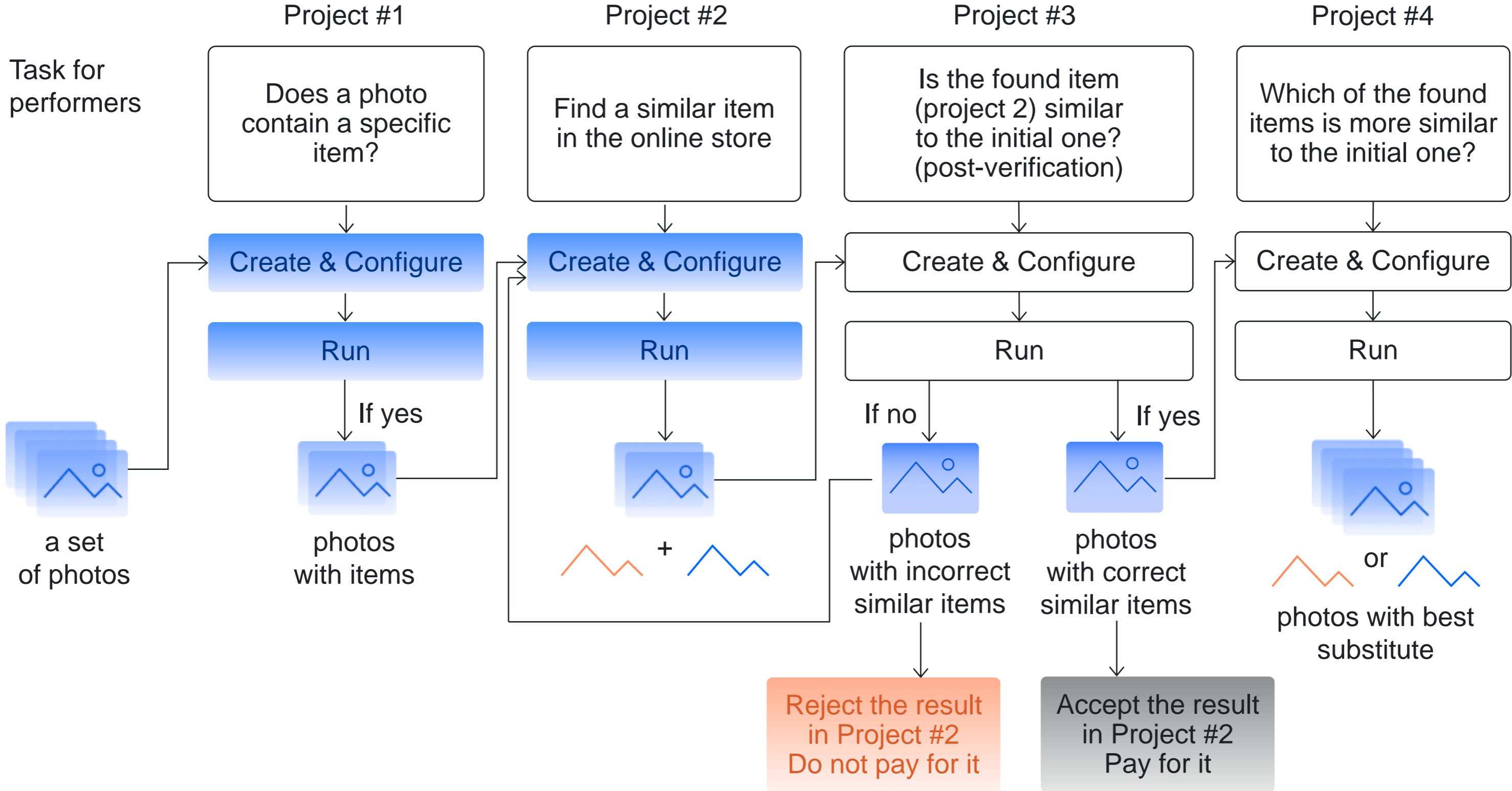
Step #2



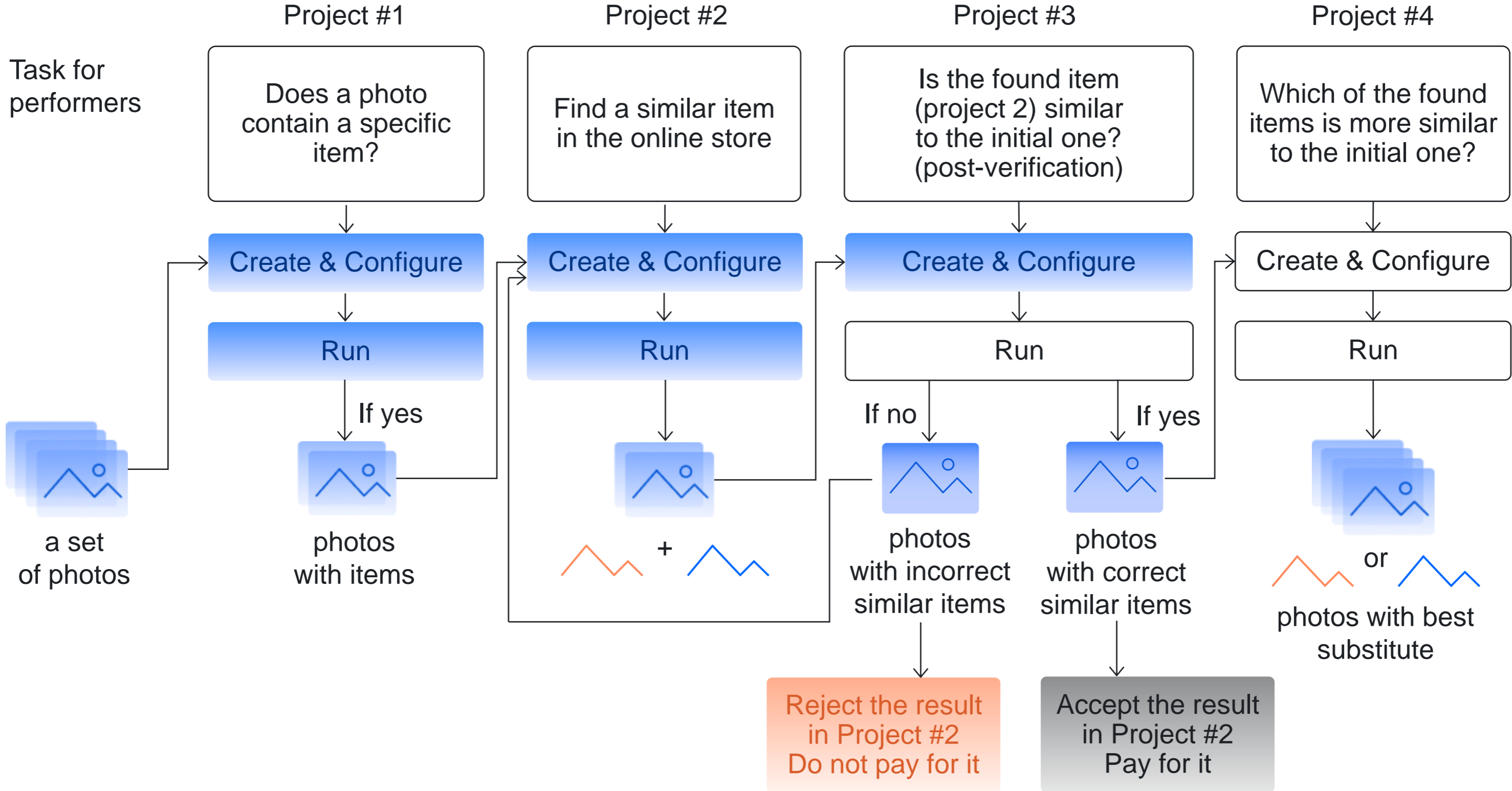
Step #3



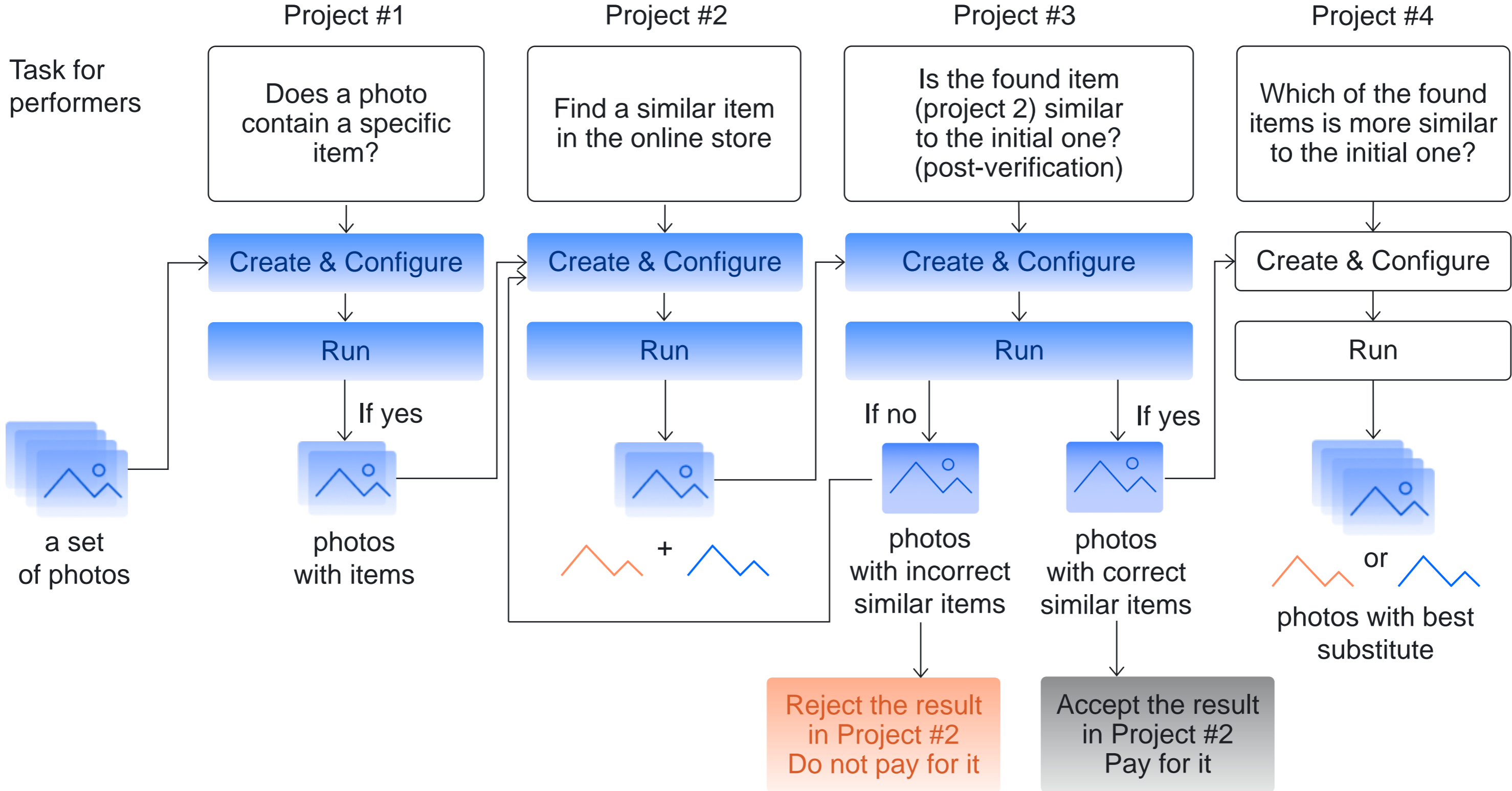
Step #4



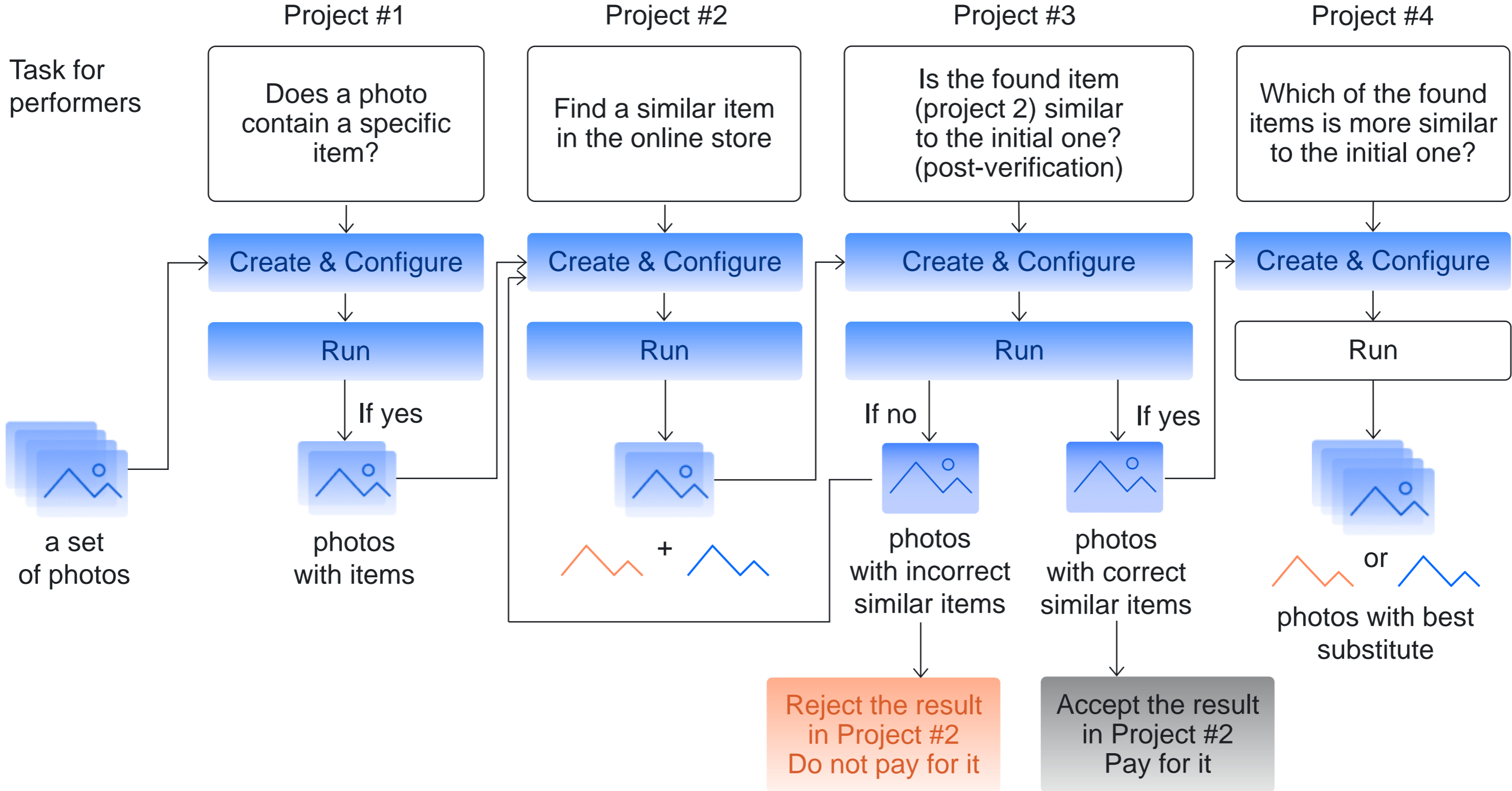
Step #5



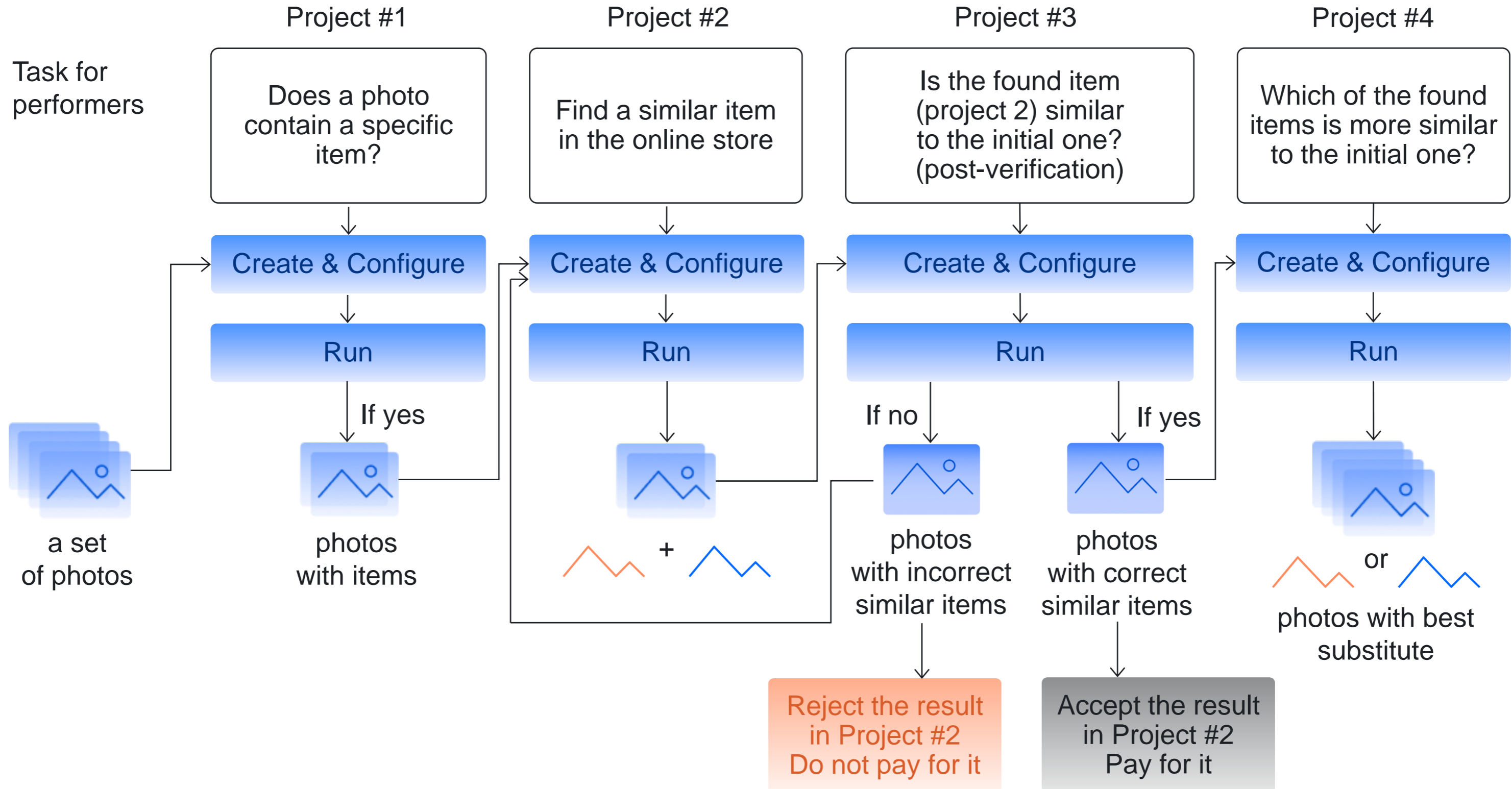
Step #6



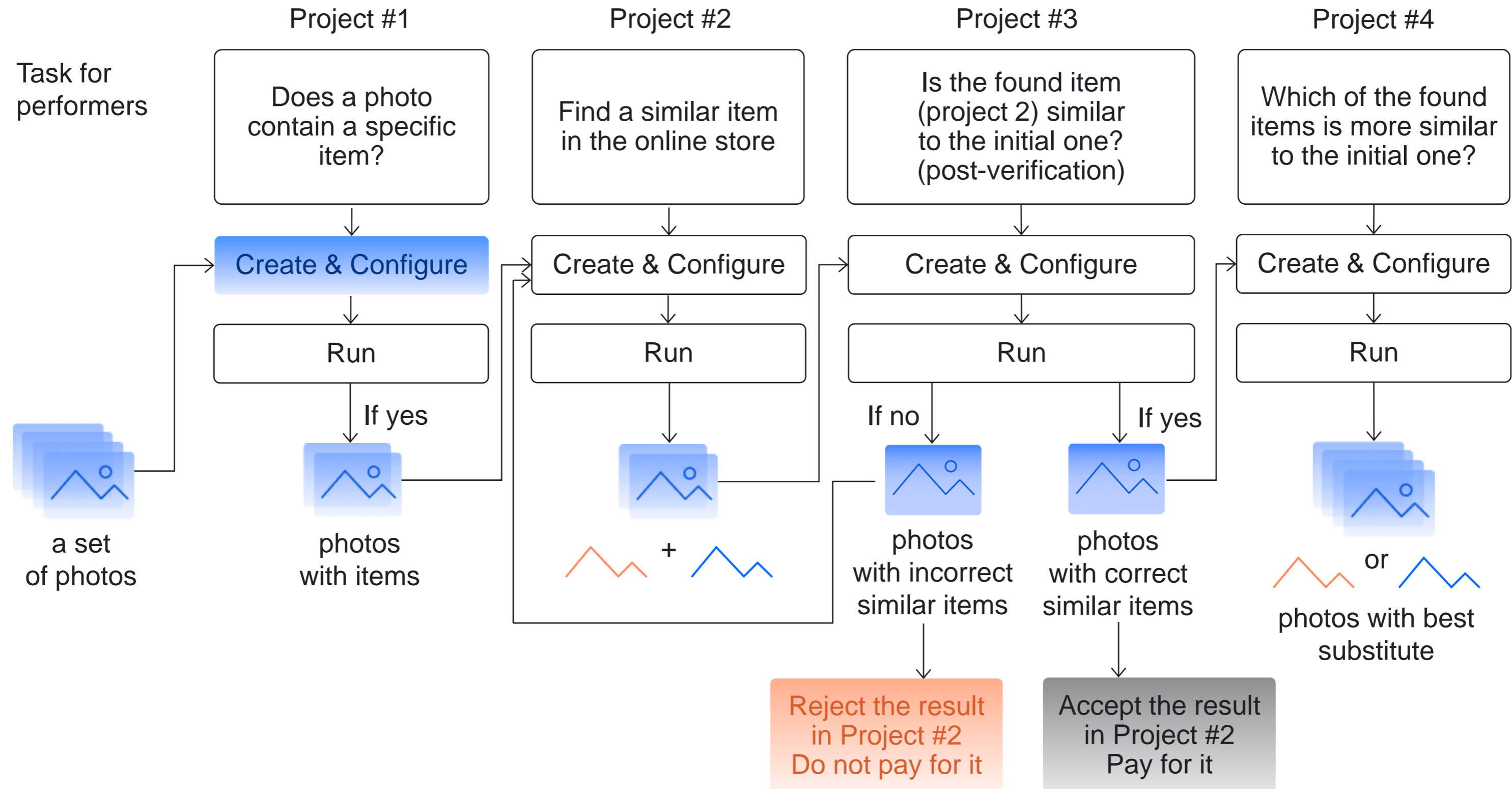
Step #7



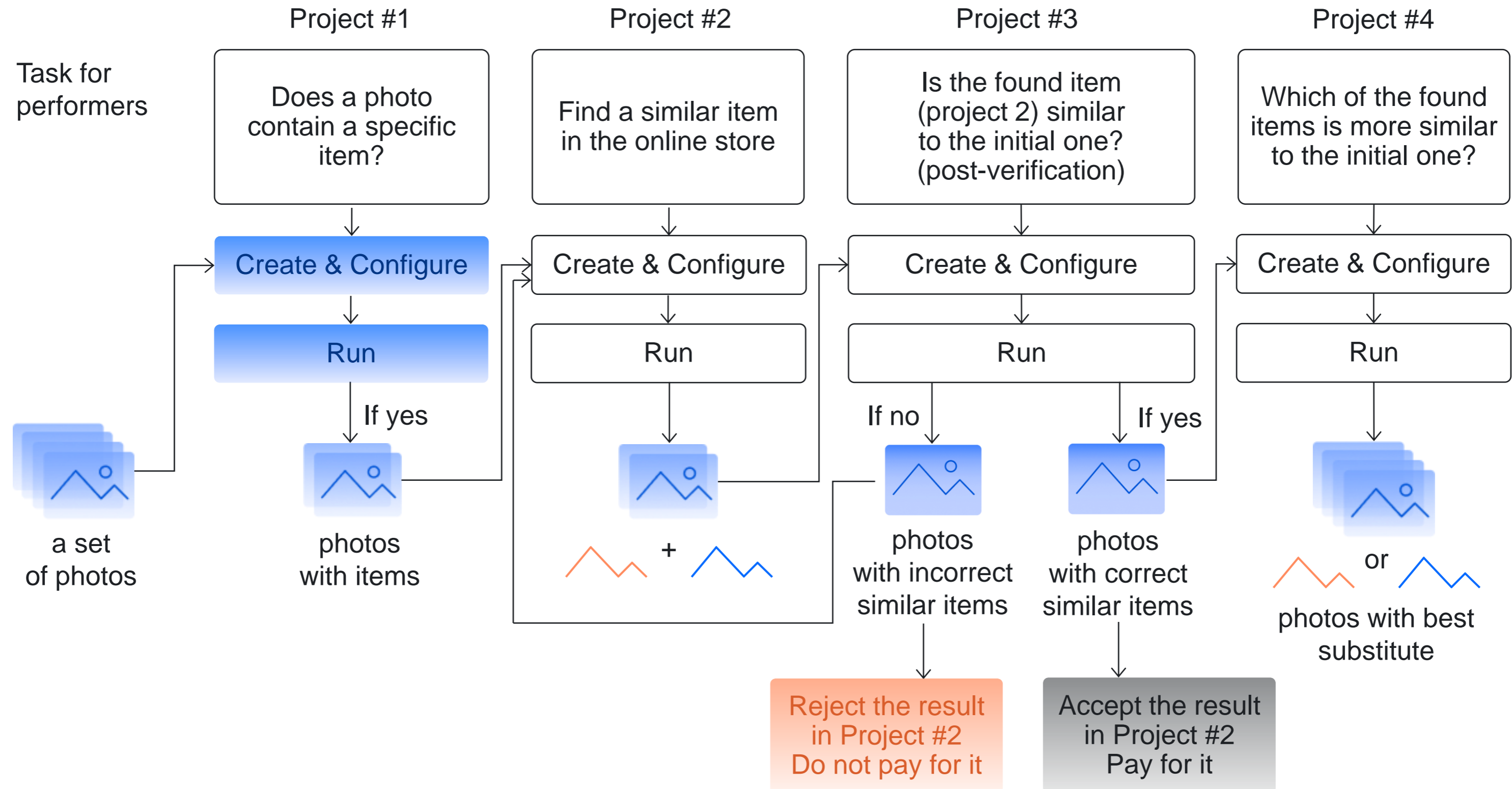
Step #8



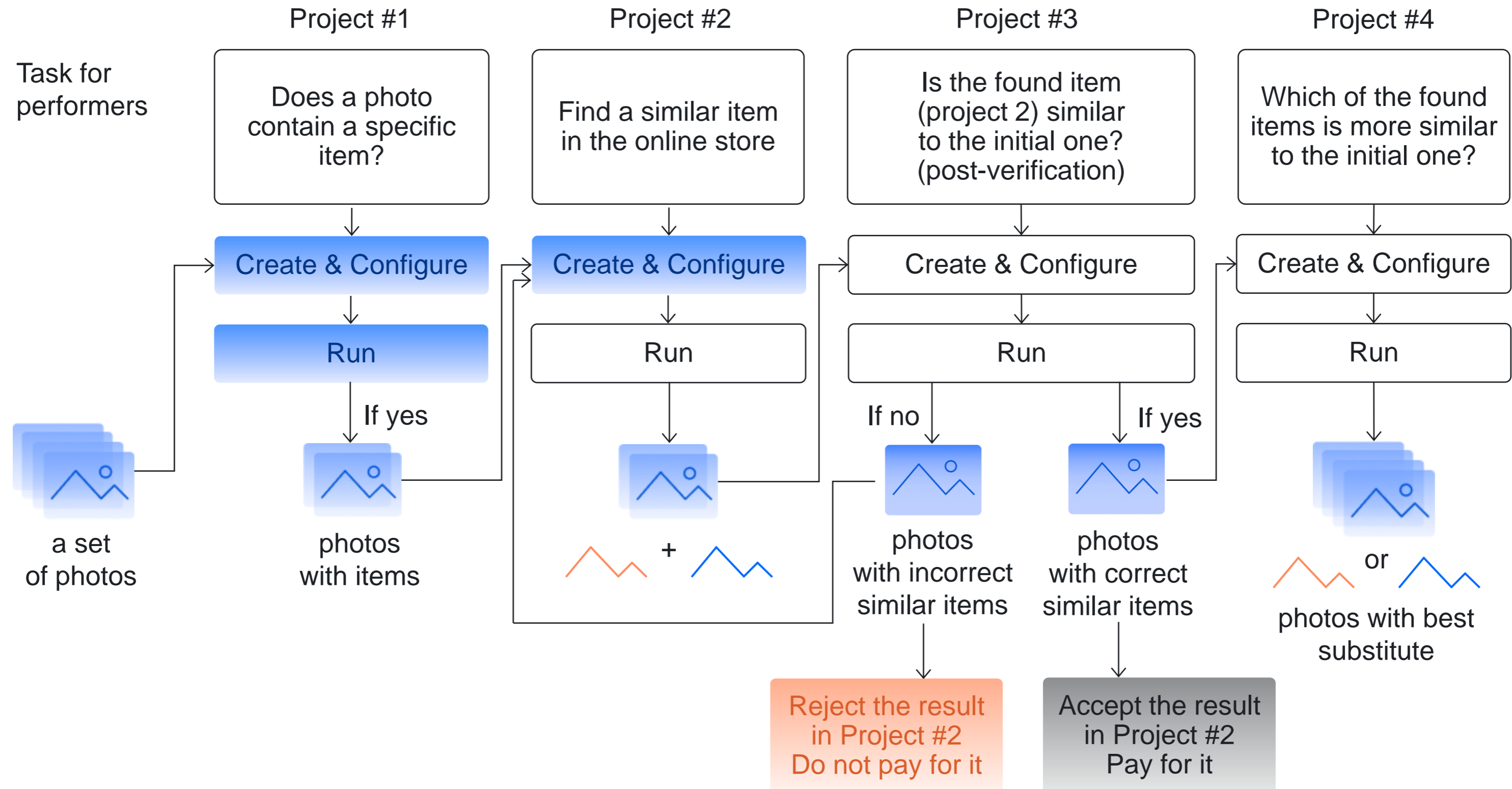
Most of us are at this step



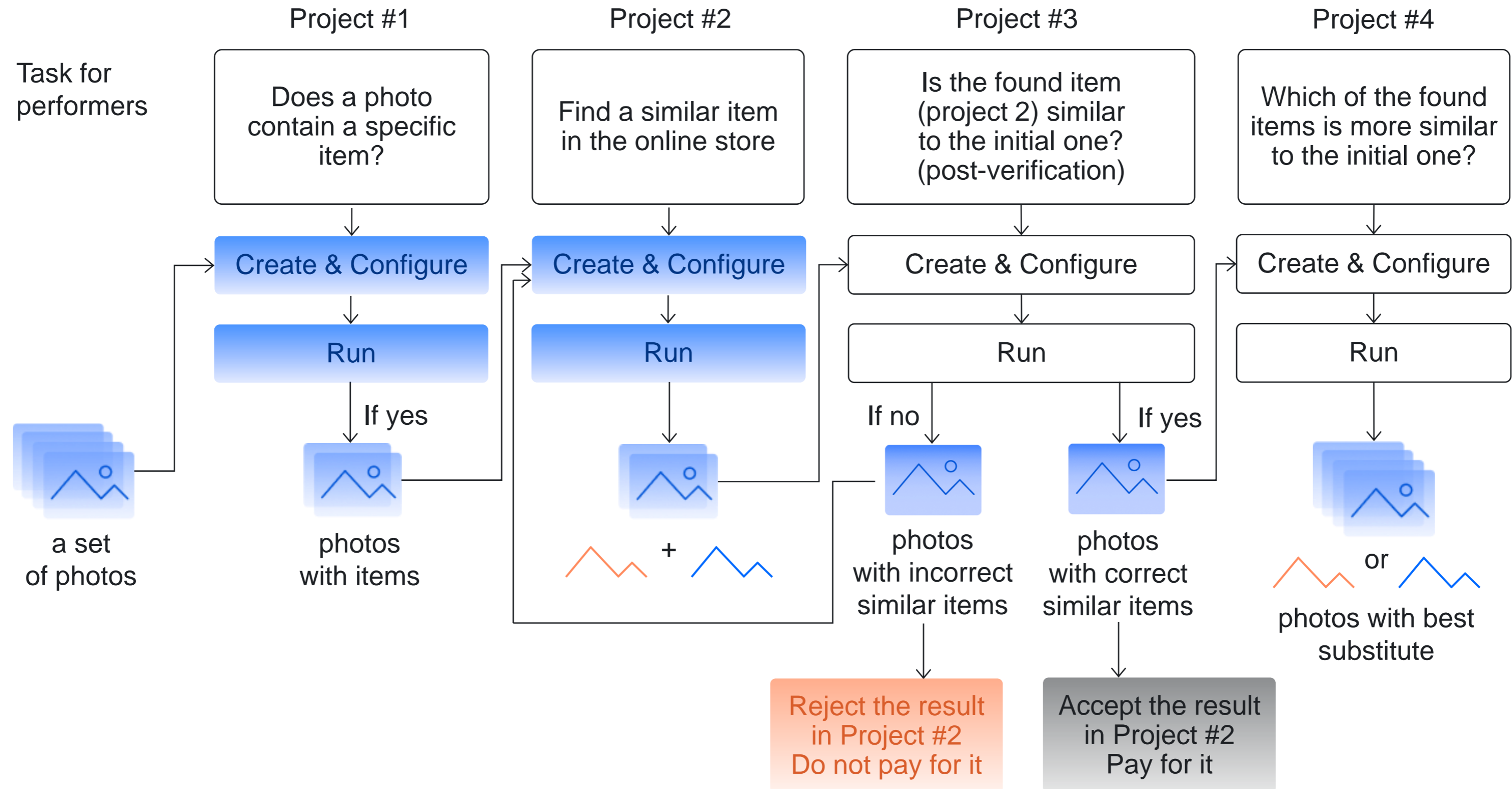
Most of us are at this step



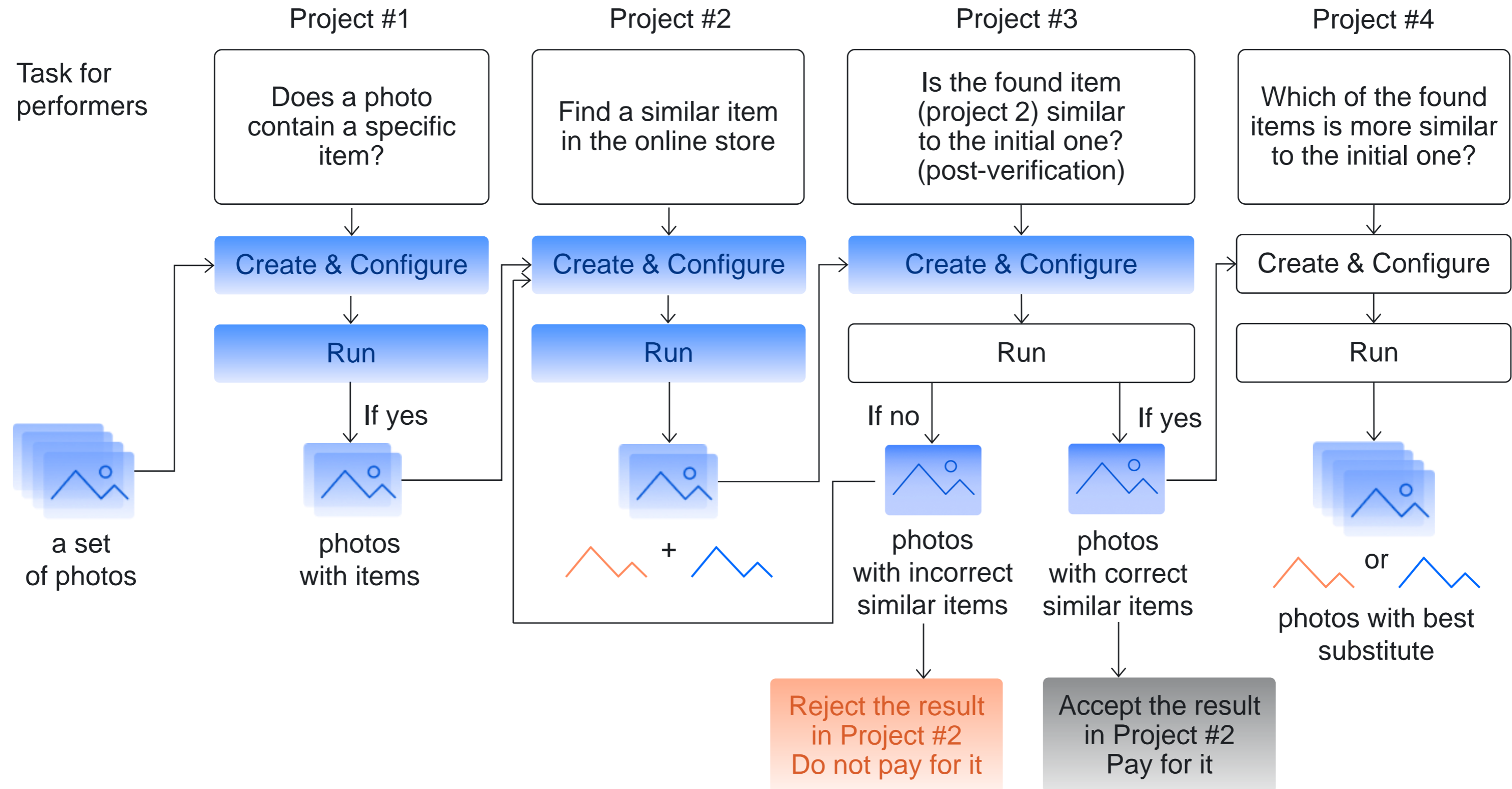
Most of us are at this step



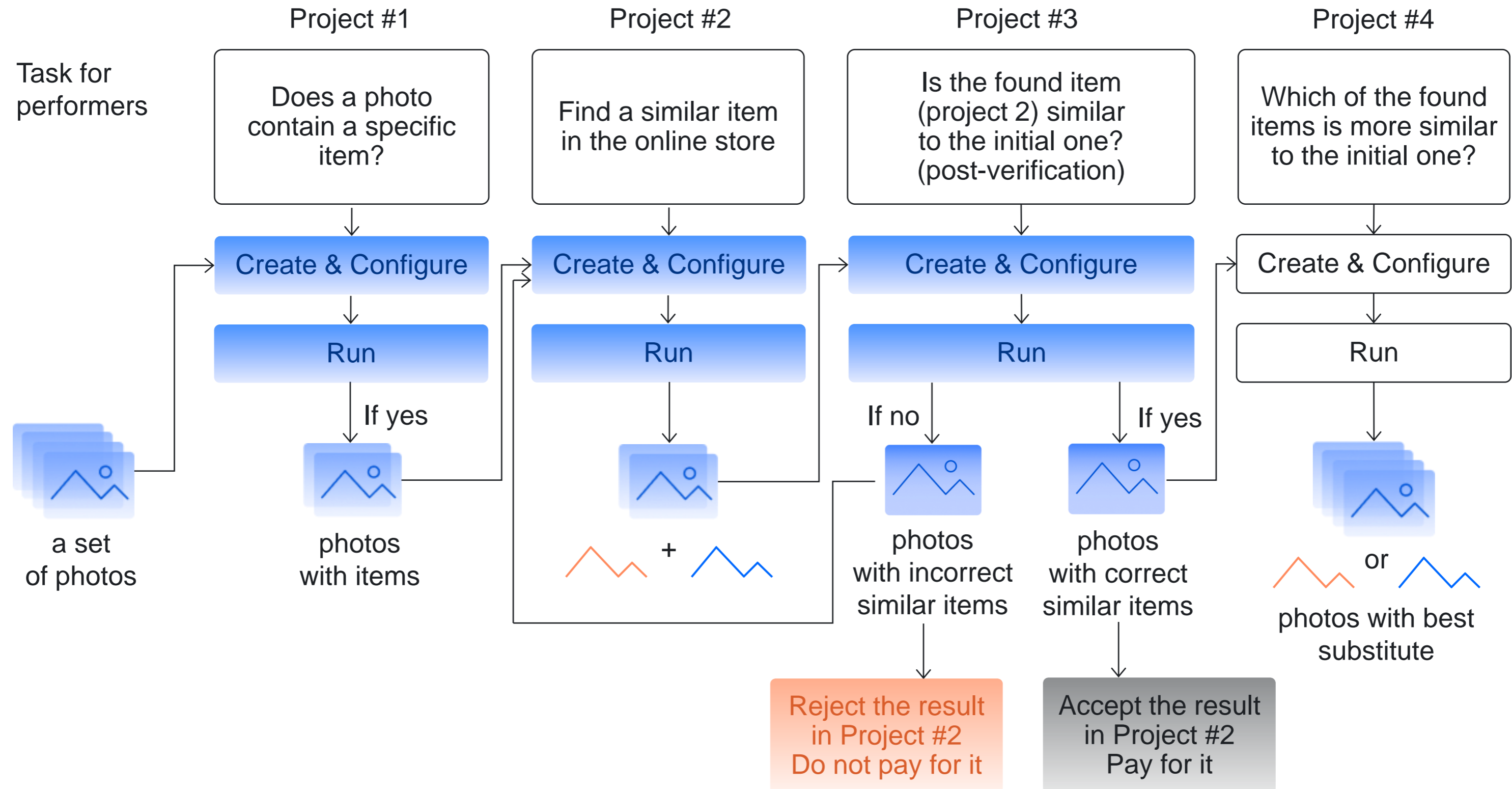
Most of us are at this step



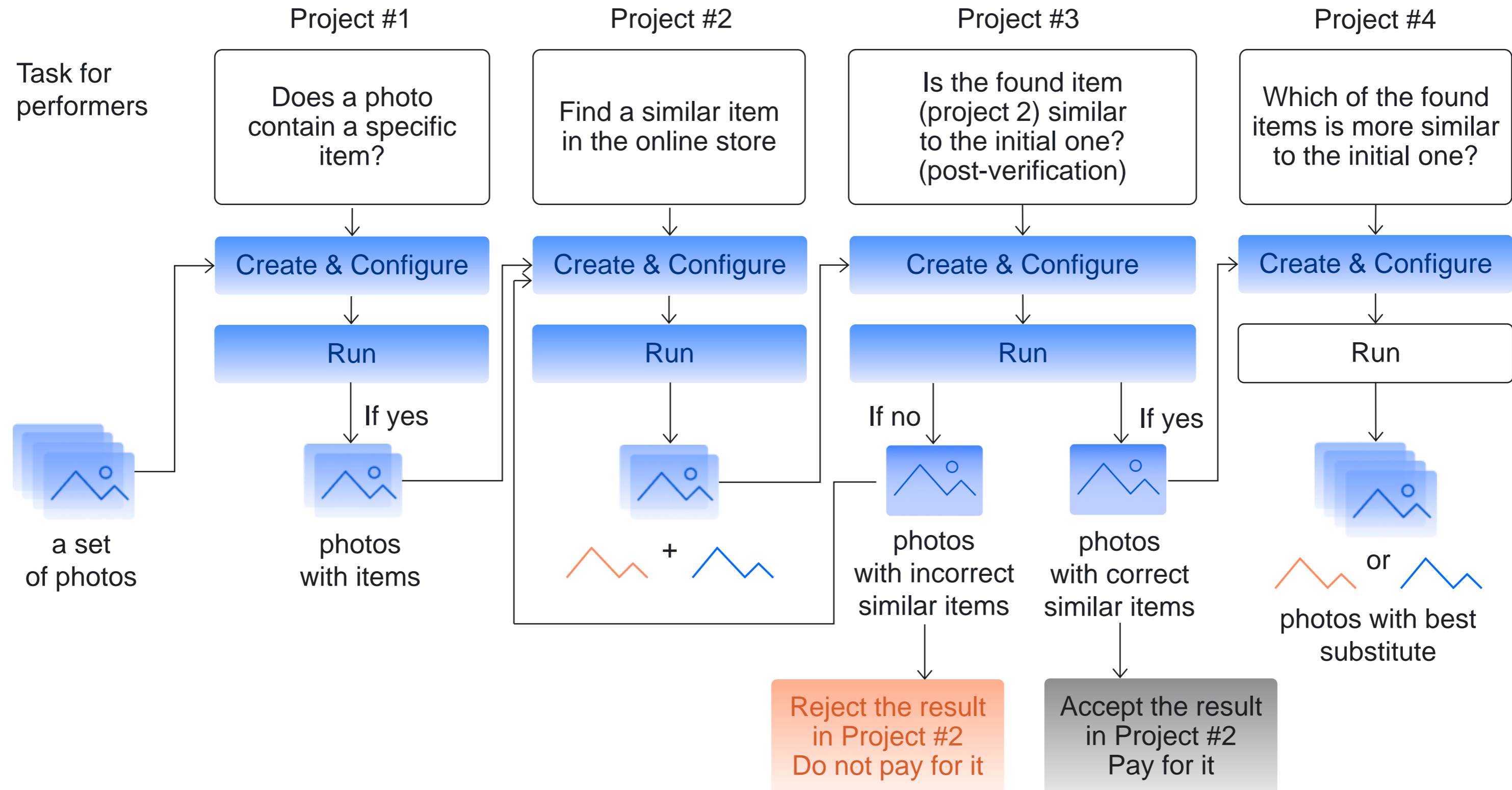
Most of us are at this step



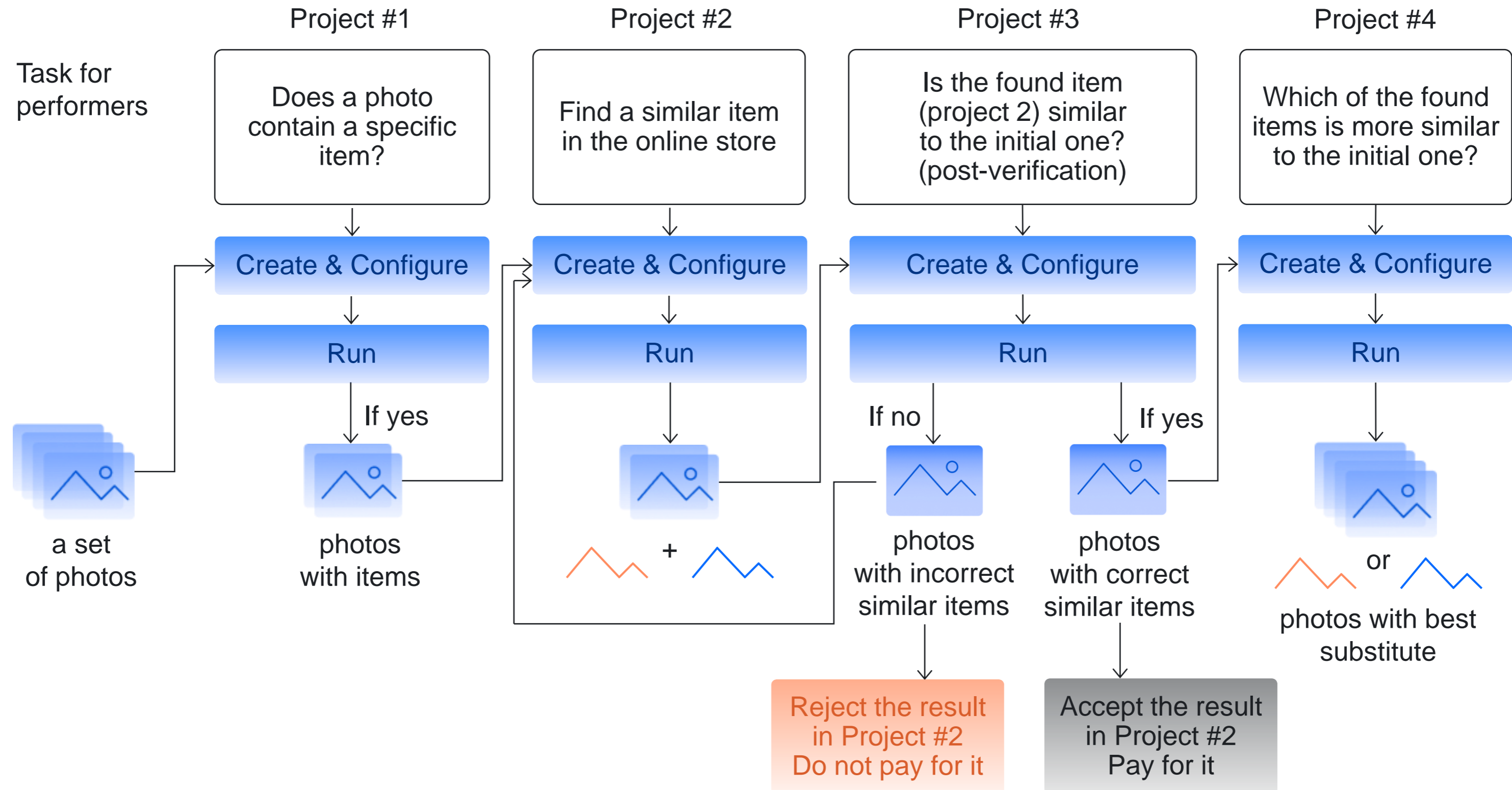
Most of us are at this step



Most of us are at this step



Most of us are at this step

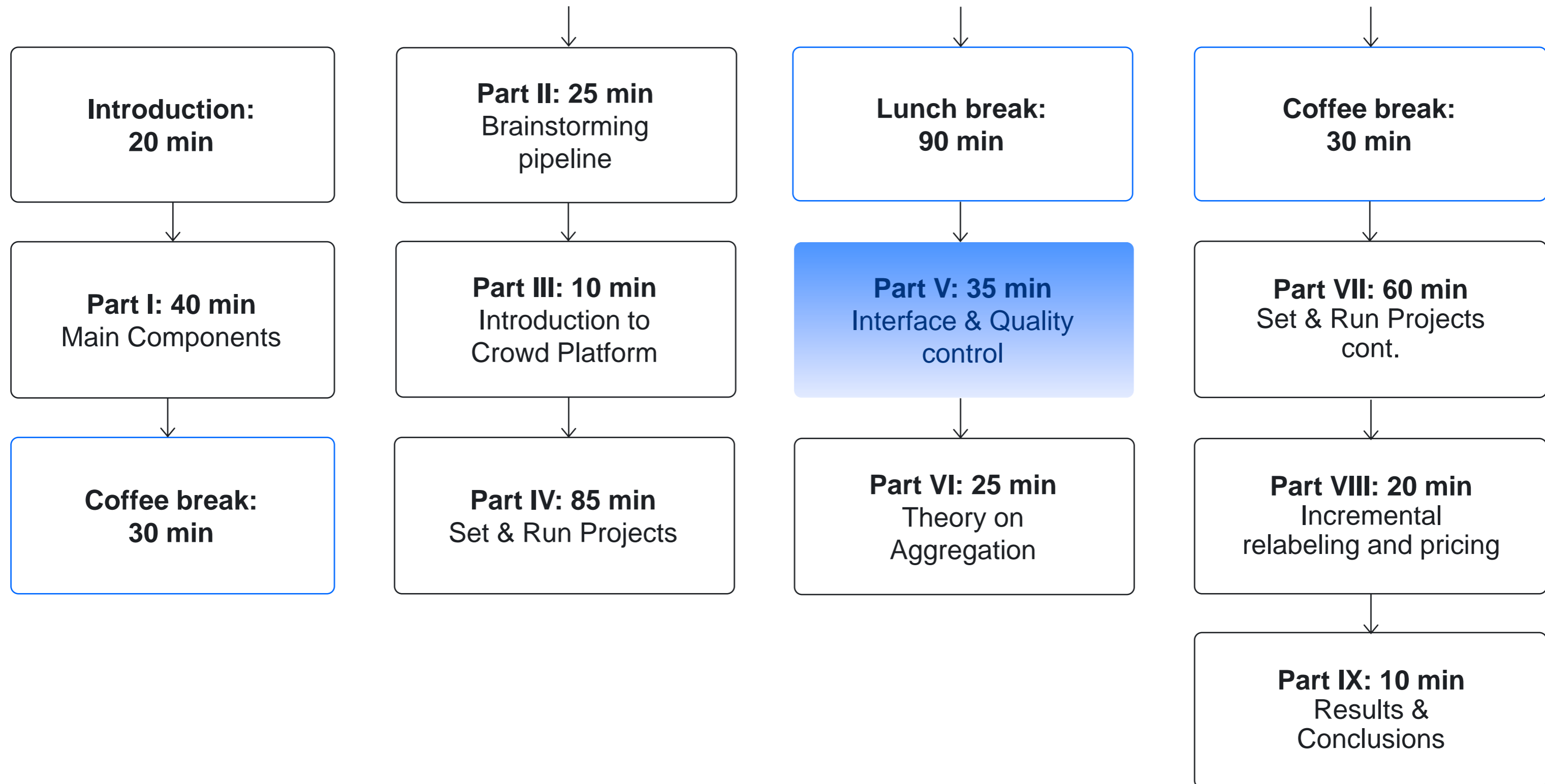


Part V

Effective quality control and task interface: details

Alexey Drutsa,
Head of Efficiency and Growth Division, Toloka

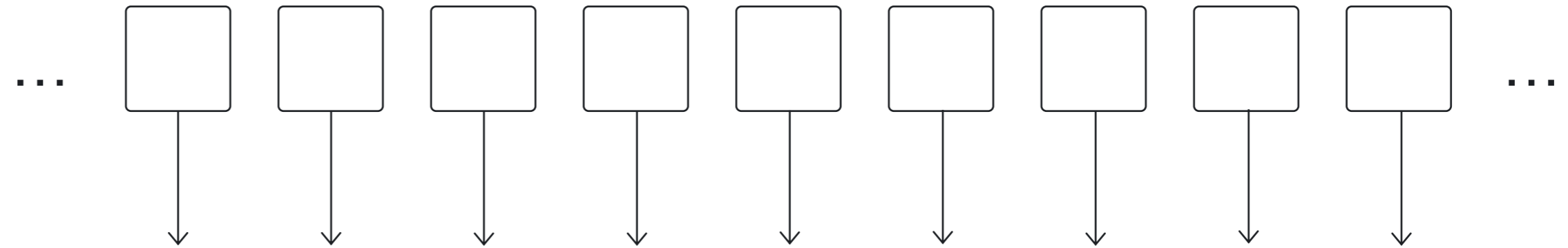
Tutorial schedule



**Quality control:
the rate of correct
answers**

Task sequence

Tasks executed
by a performer



Signals of answer
correctness

y_k y_{k+1} y_{k+2} y_{k+3} y_{k+4} y_{k+5} y_{k+6} y_{k+7} y_{k+8}

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



n , window size

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

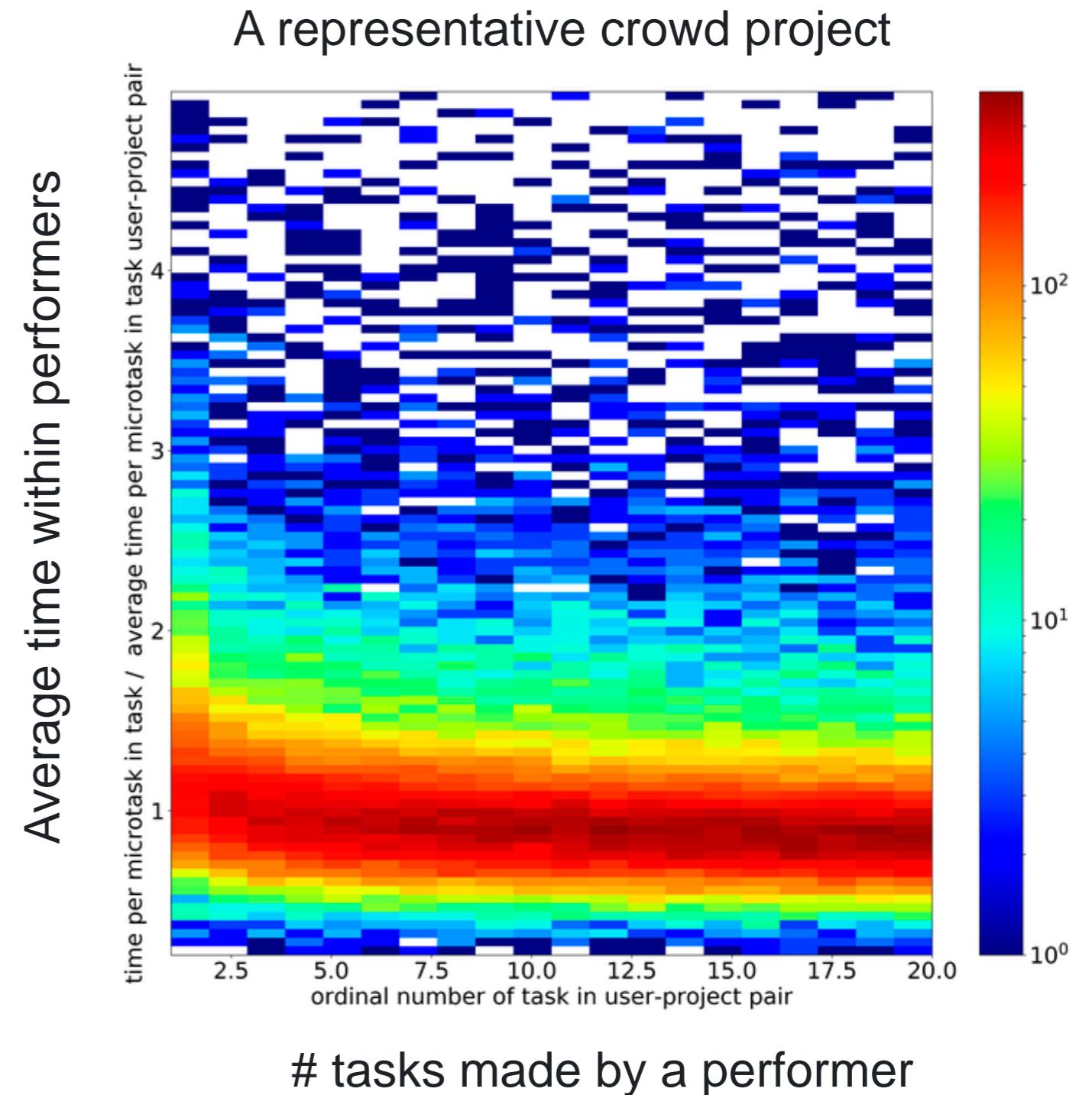
Can be used for automatic decision making

- ▶ 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



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

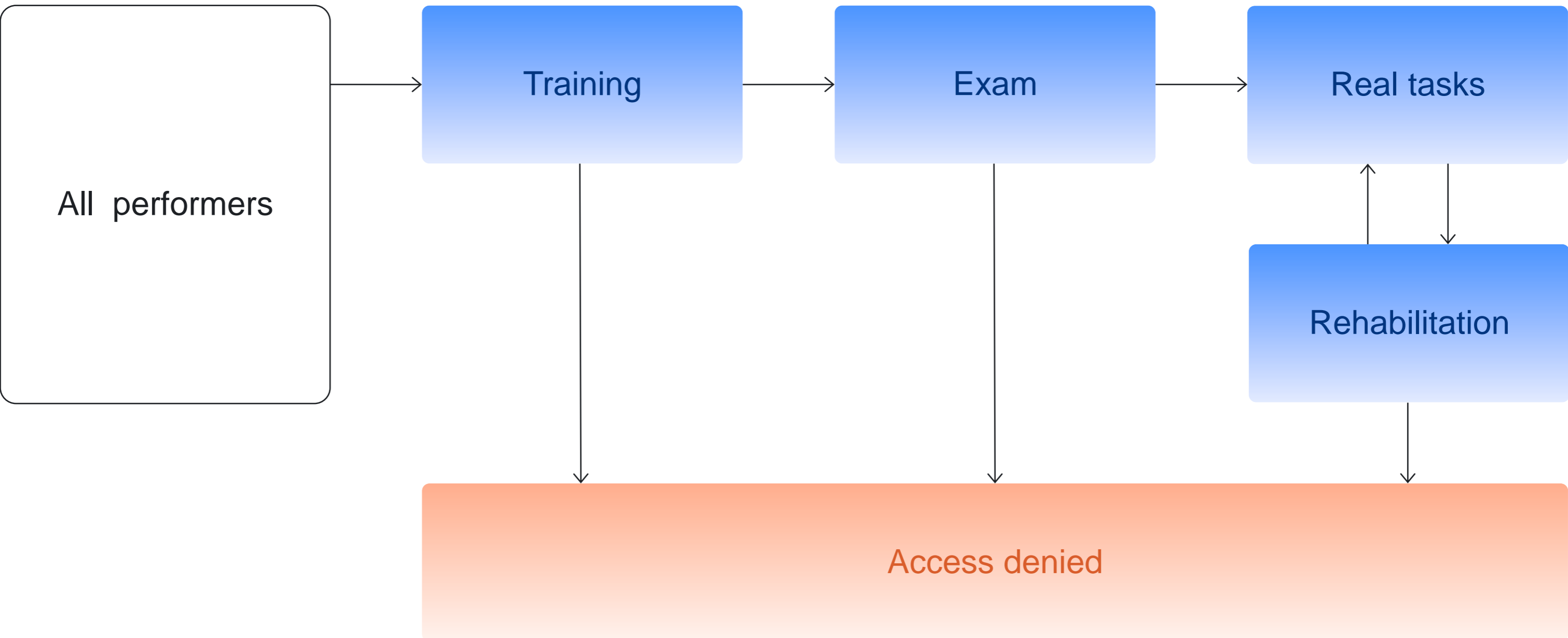
- ▶ All tasks are control ones
- ▶ There are hints that explain incorrect answers

Exam task

Control the results of training

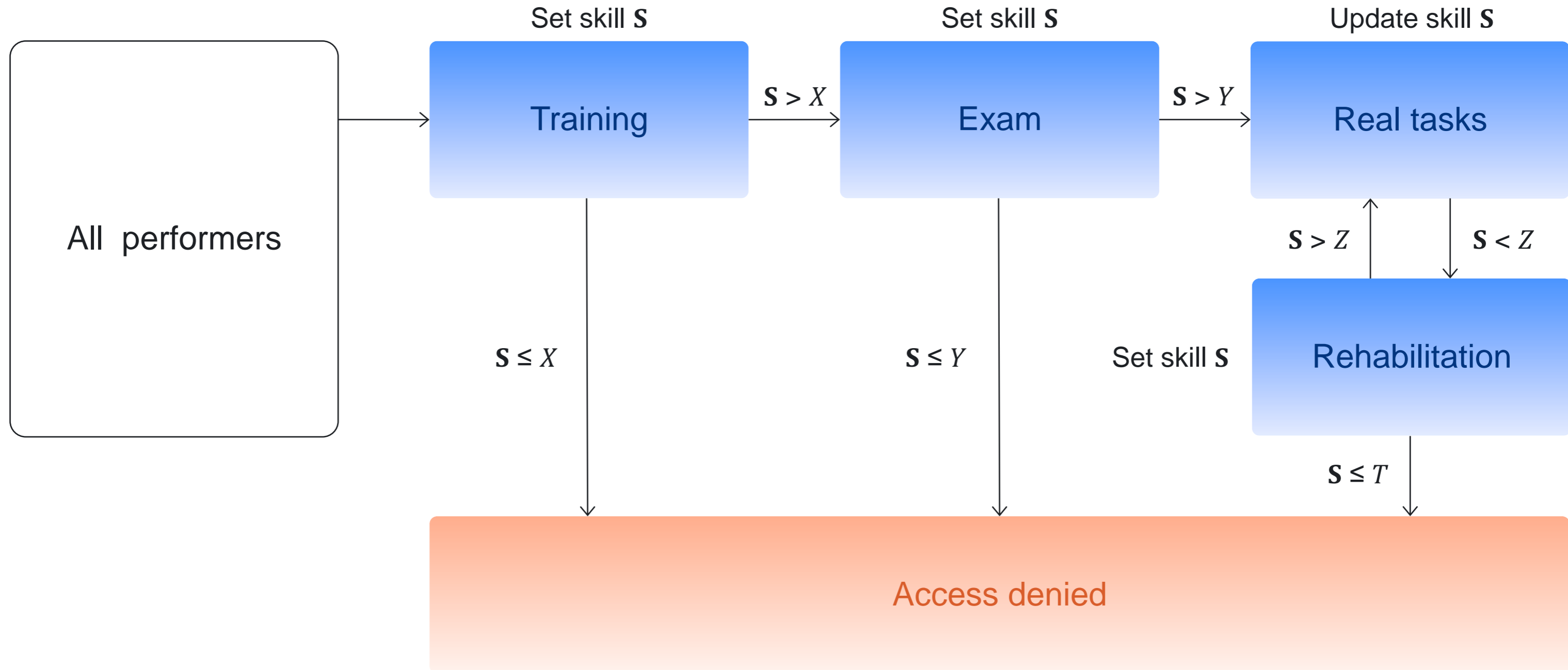
- ▶ All tasks are control ones
- ▶ No hints and explanations
- ▶ A good exam should be:
 - Passable
 - Regularly updated
 - Small

Recommended life cycle of performers



Recommended life cycle of performers

Let quality be controlled by means of a skill S



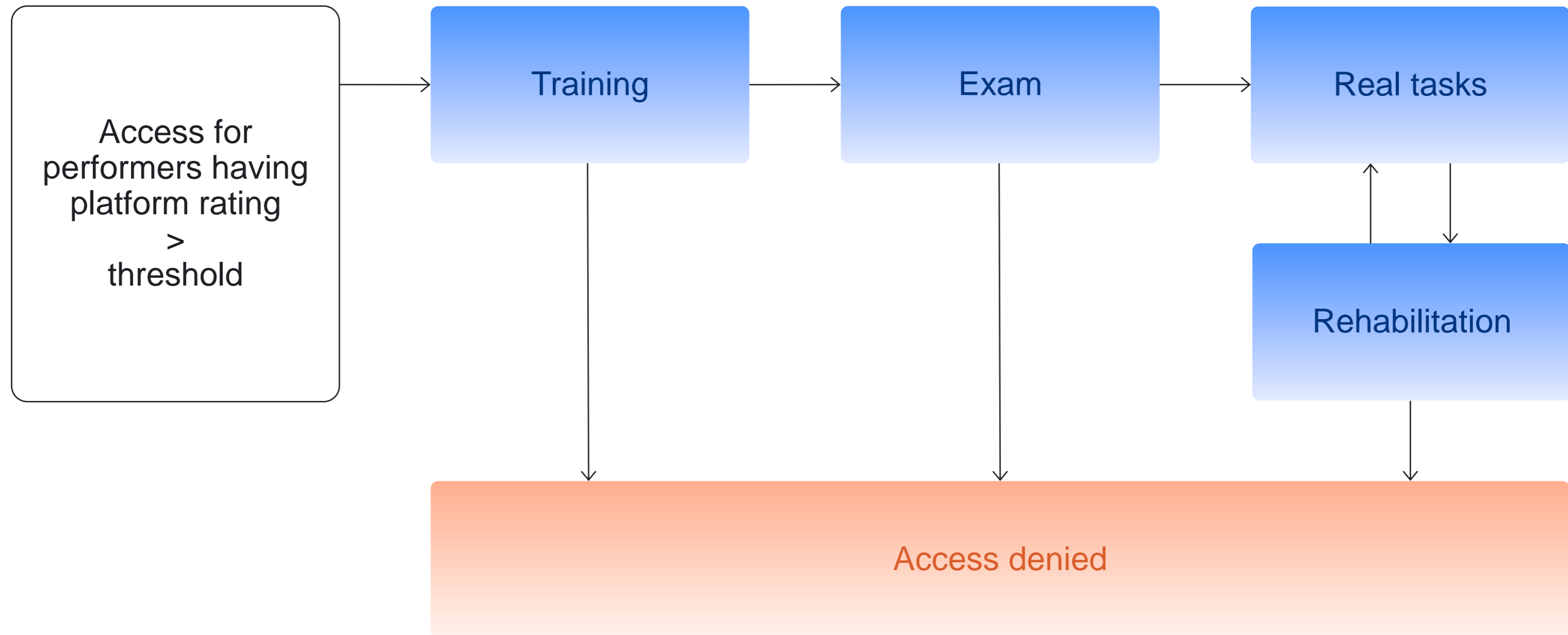
Rehabilitation task

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

The background features a dark blue gradient on the left, transitioning into a lighter blue area on the right. The right side is dominated by a series of overlapping, curved, semi-transparent blue shapes that create a sense of depth and movement, resembling a stylized fan or a series of overlapping pages.

Interface.

Introduction

Task in the eyes of the performers

Web-page with specific features

- ▶ Long run time
- ▶ Repetitive actions
- ▶ Concentration
- ▶ Speed

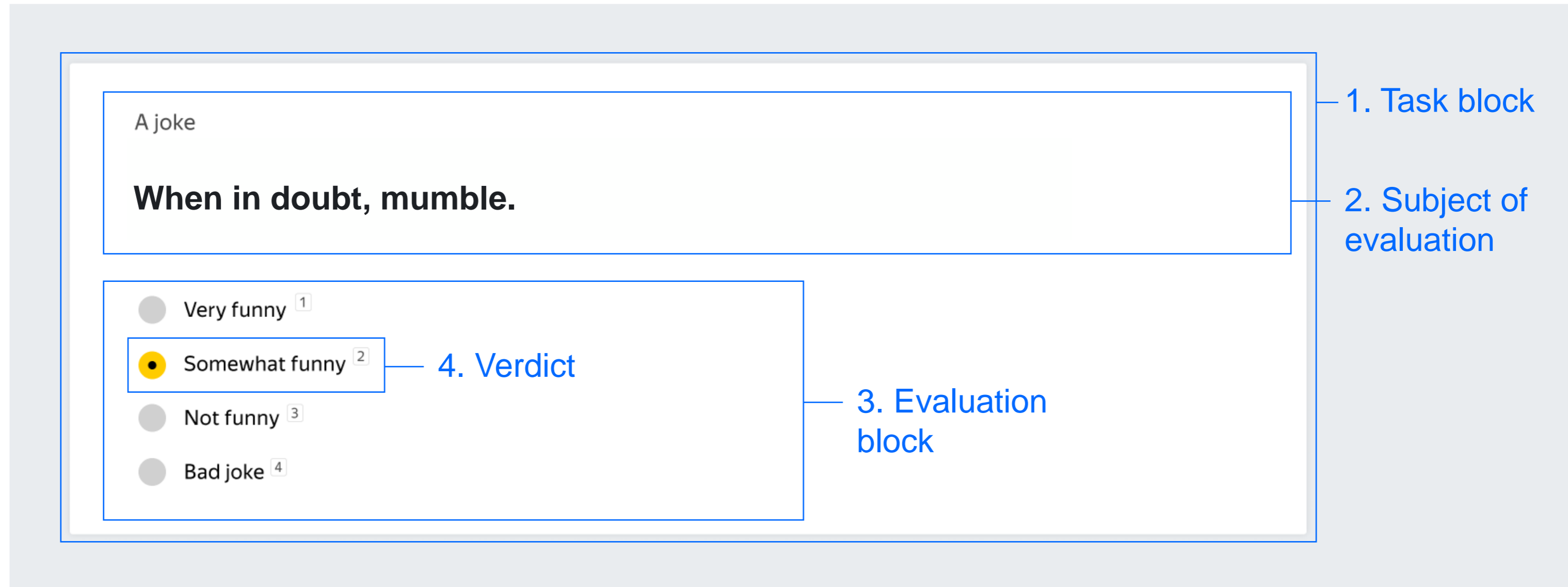
Structure of a task interface

A joke

When in doubt, mumble.

- Very funny ¹
- Somewhat funny ²
- Not funny ³
- Bad joke ⁴

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



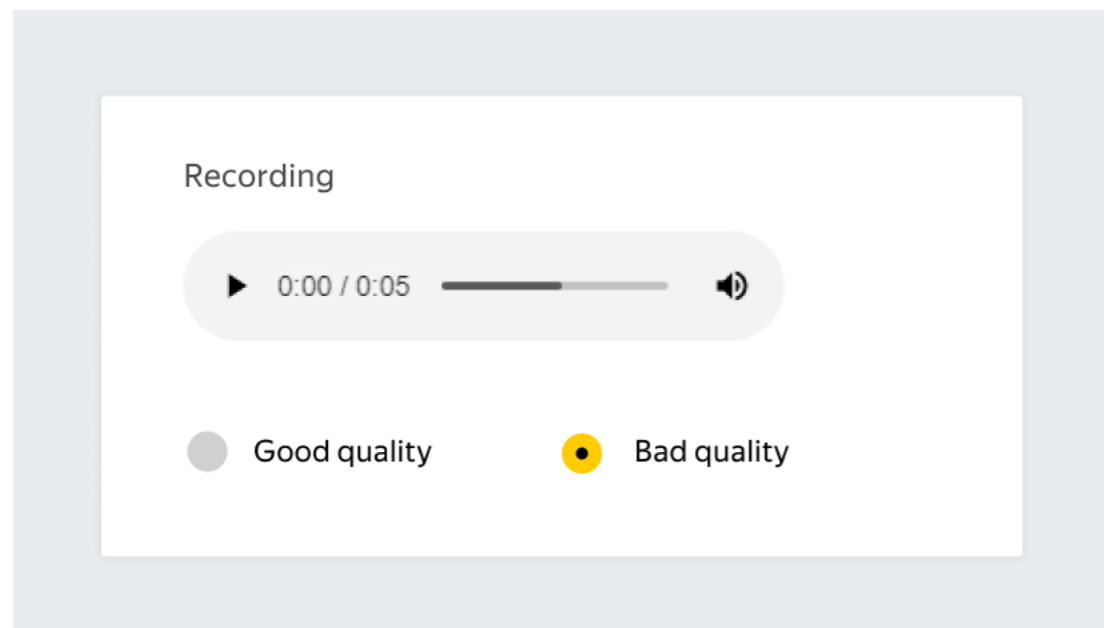
Possible limitations for mobile services:

- ▶ Task difficulty
- ▶ Media Content, Devices, and Browsers

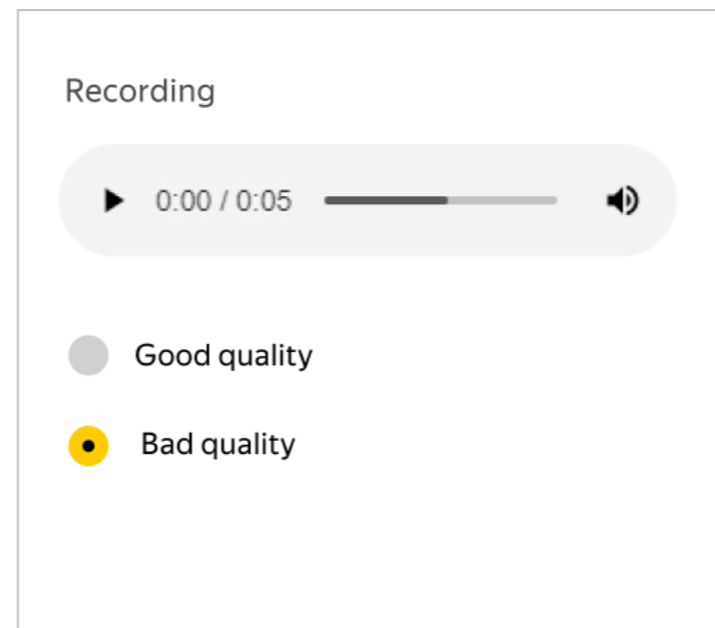
Rule #1. Cross-platform compatibility

Task: evaluate sound quality in wav audio files

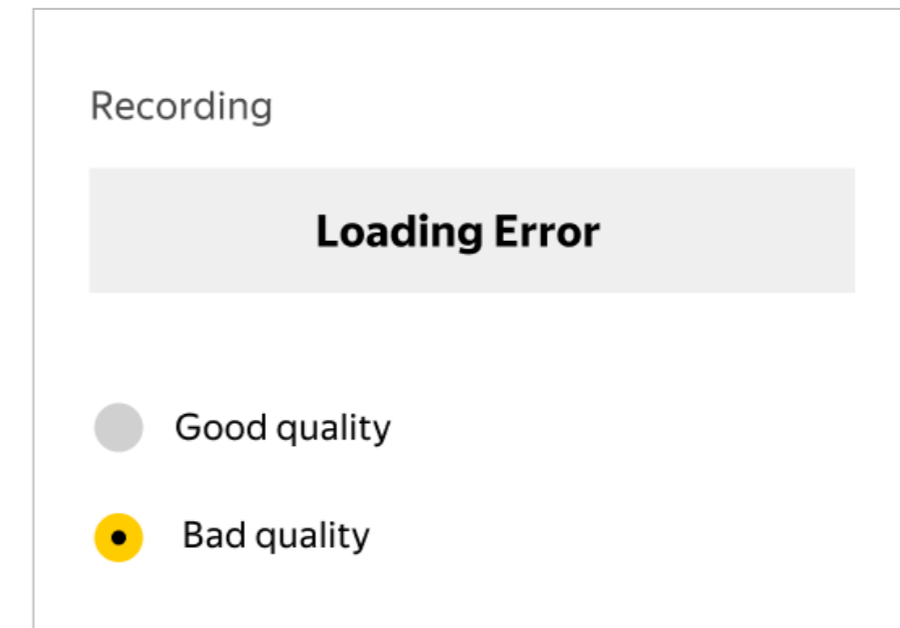
Web version



Android App



IOS App



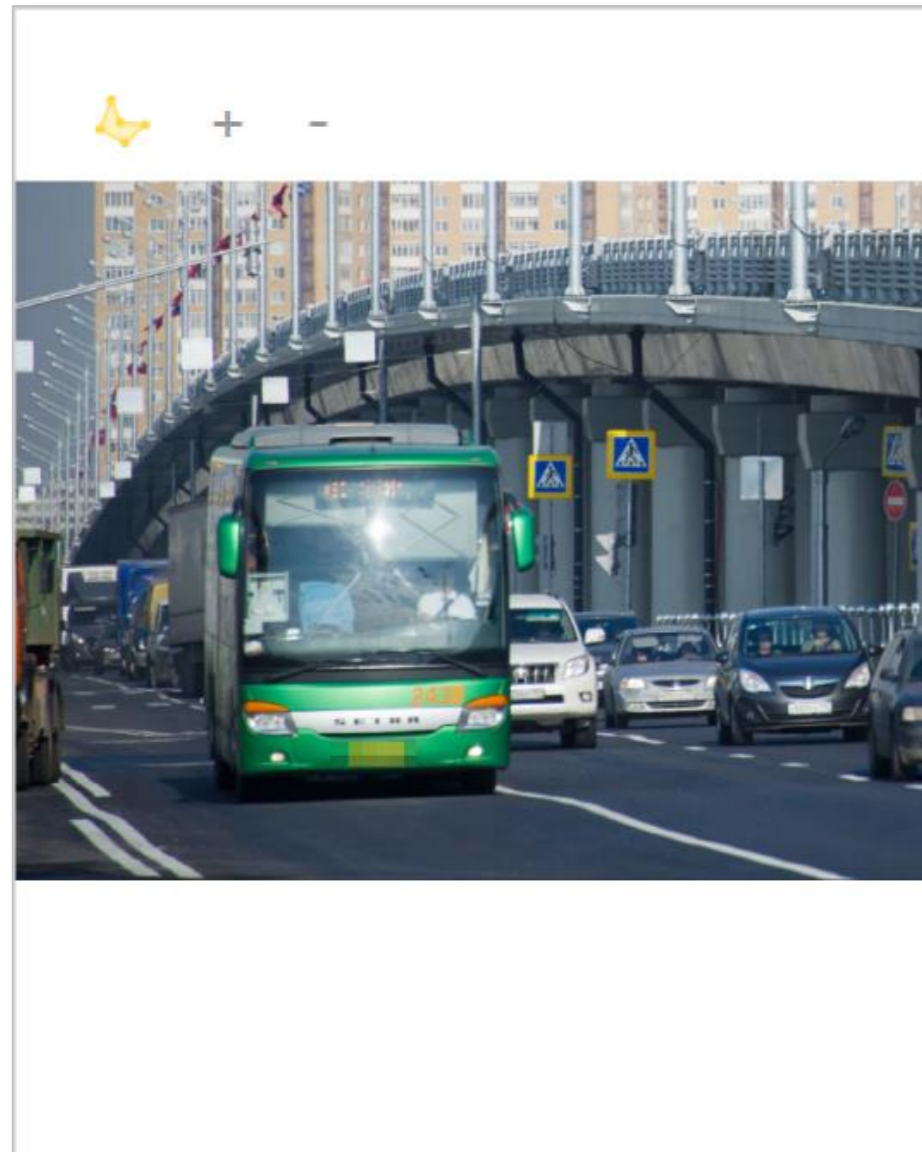
Rule #1. Cross-platform compatibility

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

Phrase [job occupation in New York](#)

Query [New York employment center](#)

Additionally 

Ad headline New York employment center

Text Find a stable job on nycjobs.com

Does the phrase match the query?

Yes ¹ No ²

Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match

Phrase [job occupation in New York](#)

Query [New York employment](#)

Additionally [?](#)

Ad headline [New York employment](#)

Text [Find a stable job on nycj](#)

Does the phrase match the query?

Yes ¹ No ²

Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match

The screenshot shows a search evaluation interface. At the top, there are two fields: 'Phrase' with the value 'job occupation in New York' and 'Query' with the value 'New York employment'. Below these is a section titled 'Additionally' with a question mark icon. Underneath, there are two rows: 'Ad headline' with the value 'New York employment' and 'Text' with the value 'Find a stable job on nycj'. At the bottom, there is a question 'Does the phrase match the query?' followed by two radio button options: 'Yes' (with a superscript '1') and 'No' (with a superscript '2'). The 'No' option is selected. Annotations with blue arrows point to various parts: 'Cut off text' points to the end of the 'Phrase' field; 'Hotkeys' points to the 'Yes' and 'No' radio buttons; and 'Empty space' points to the bottom-left corner of the interface.

Phrase job occupation in New York ← Cut off text

Query New York employment

Additionally ?

Ad headline New York employment

Text Find a stable job on nycj

Does the phrase match the query?

Yes¹ No² ← Hotkeys

Empty space

Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match

Phrase
[job occupation in New York](#)

Query
[New York employment center](#)

Additionally
Ad headline
New York employment center

Text
Find a stable job on nycjobs.com

Does the phrase match the query?

Yes No

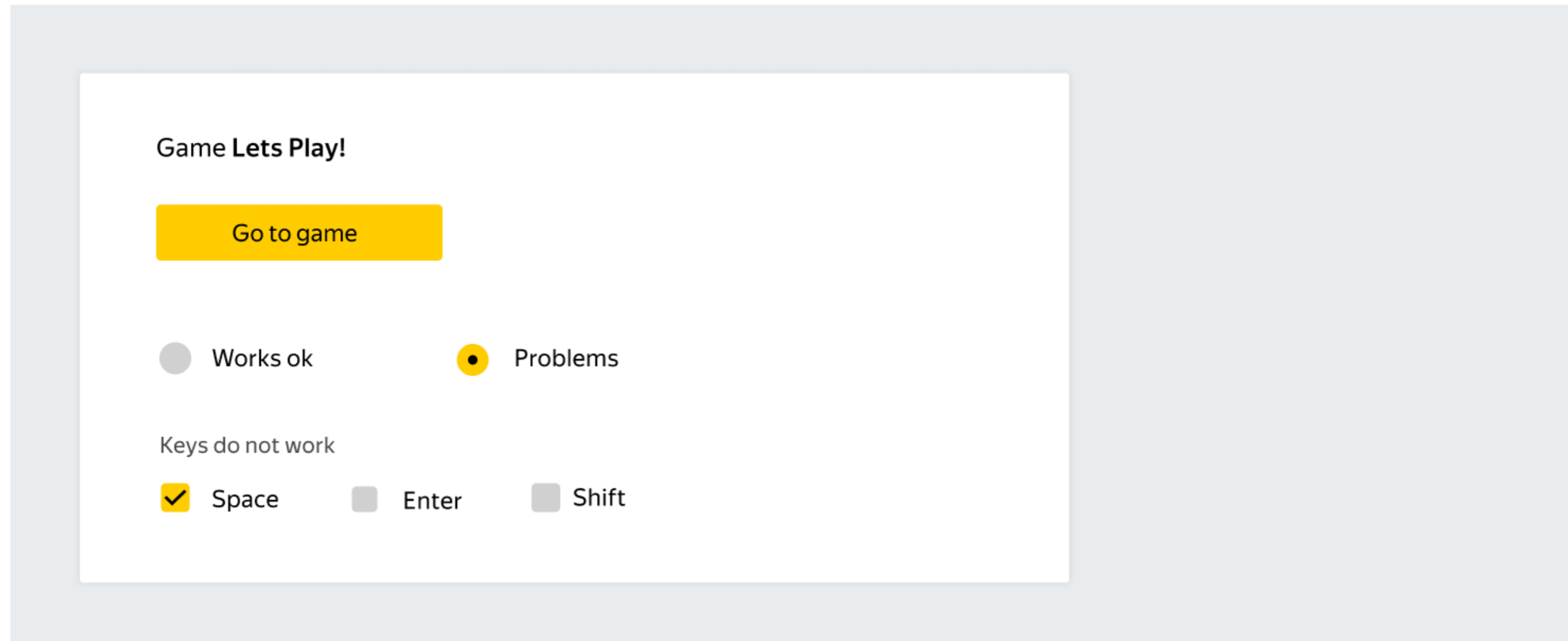
Rule #2. Hotkeys

- ▶ 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

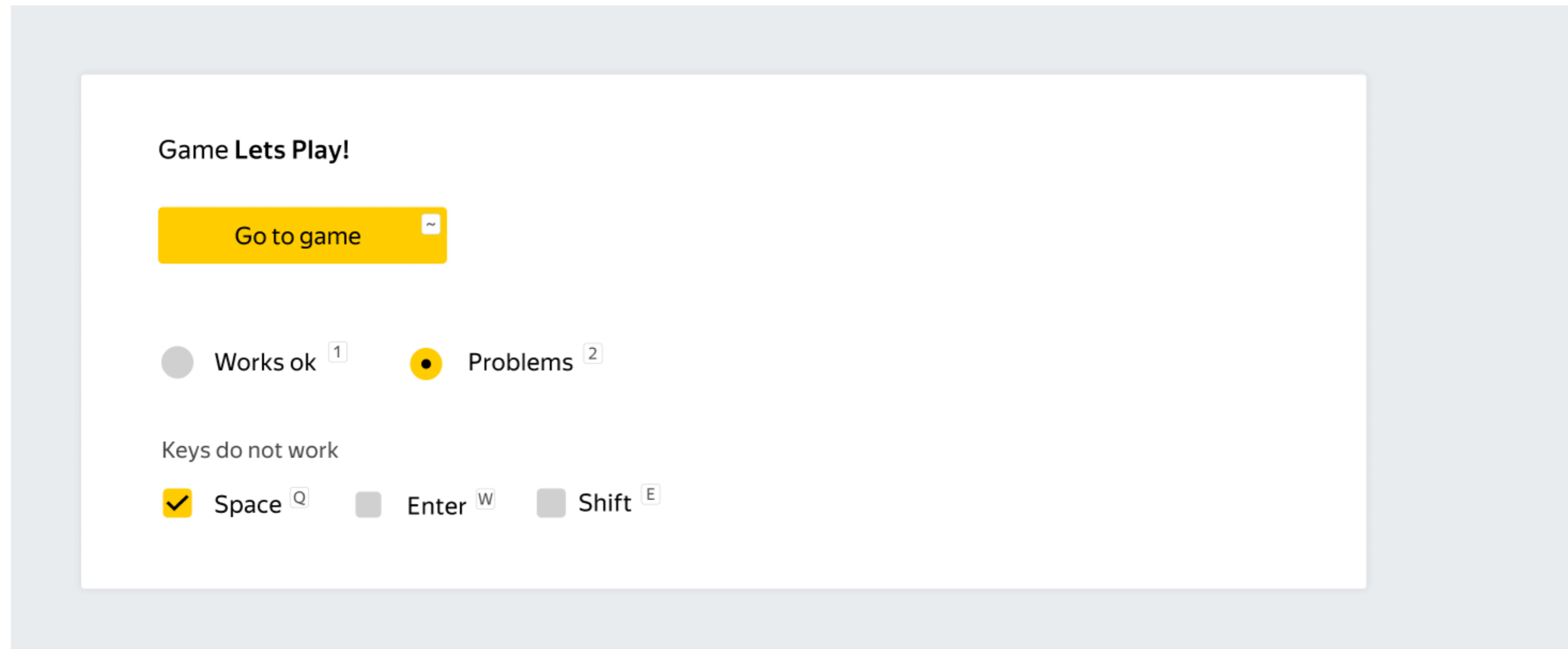
Rule #2. Hotkeys

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)

Game Lets Play!

Go to game ~

Works ok ¹ Problems ² Does not open ³

Keys do not work

Space ^Q Enter ^W Shift ^E

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

Finish the task
as fast as possible!



Performer

Rule #3. Action and data check

Game Lets Play!

Go to game Please, go to the game page

Works ok Problems

Keys do not work

Space Enter Shift

The screenshot shows a user interface for a game. At the top, it says "Game Lets Play!". Below this is a yellow button labeled "Go to game" and a red tooltip that says "Please, go to the game page". Underneath, there are two radio buttons: "Works ok" (which is unselected) and "Problems" (which is selected). Below the radio buttons, it says "Keys do not work" and lists three keys: "Space" (with a checked checkbox), "Enter" (with an unchecked checkbox), and "Shift" (with an unchecked checkbox).

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

Phrase job occupation in New York

Query New York employment center

Additionally

Ad headline Jobs in New York

Text Find a stable job on nycjobs.com

Does the phrase match the query?

[Yes](#)

[No](#)

Rule #6. Avoid experimental design

Extra nesting of the blocks

Unnecessary bright color

Phrase job occupation in New York

Query New York employment center

All text is in one font

Additionally

Ad headline Jobs in New York

Text Find a stable job on nycjobs.com

A lot of empty space on the right side of the block

Does the phrase match the query?

Yes No

Odd display of verdicts

2 types of patterns

Rule #6. Avoid experimental design

Phrase **job occupation in New York**

Query **New York employment center**

Additionally

Ad headline **Jobs in New York**

Text **Find a stable job on nycjobs.com**

The phrase match the query ¹

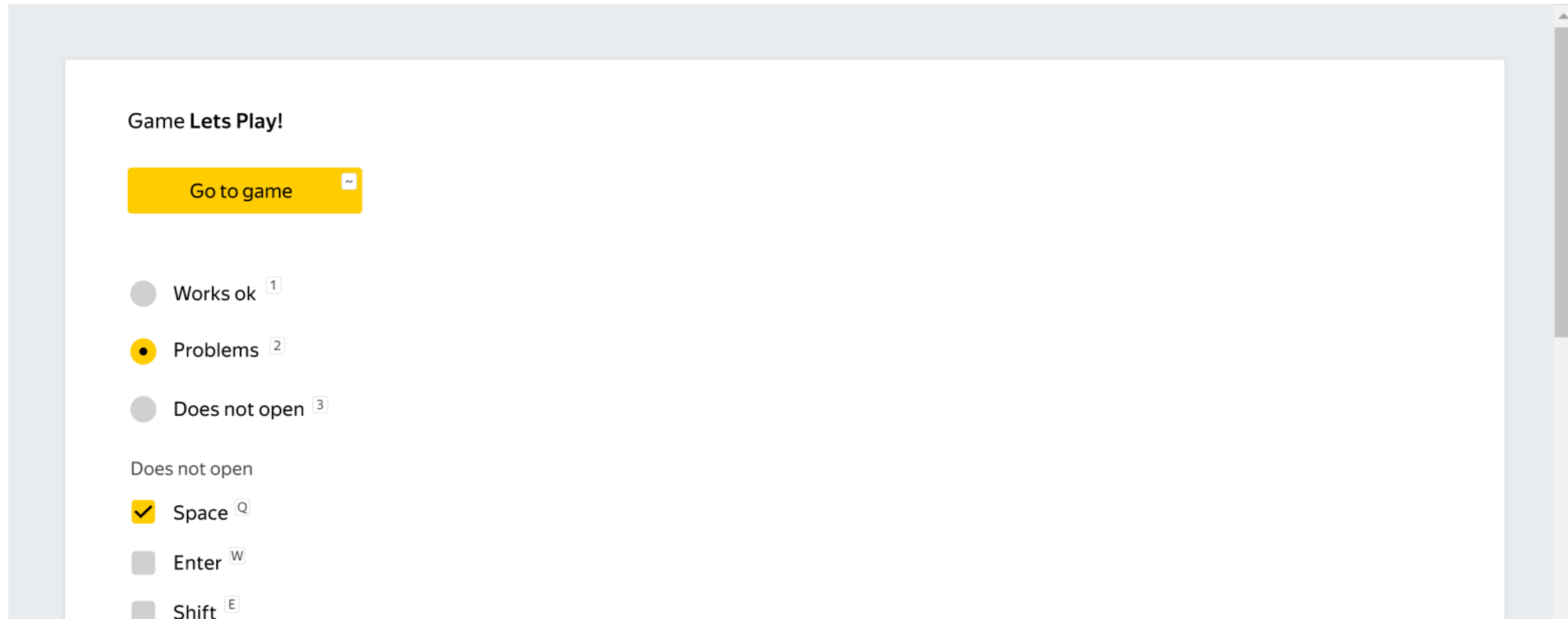
The phrase doesn't match the query ²

Rule #7. Efficient space usage

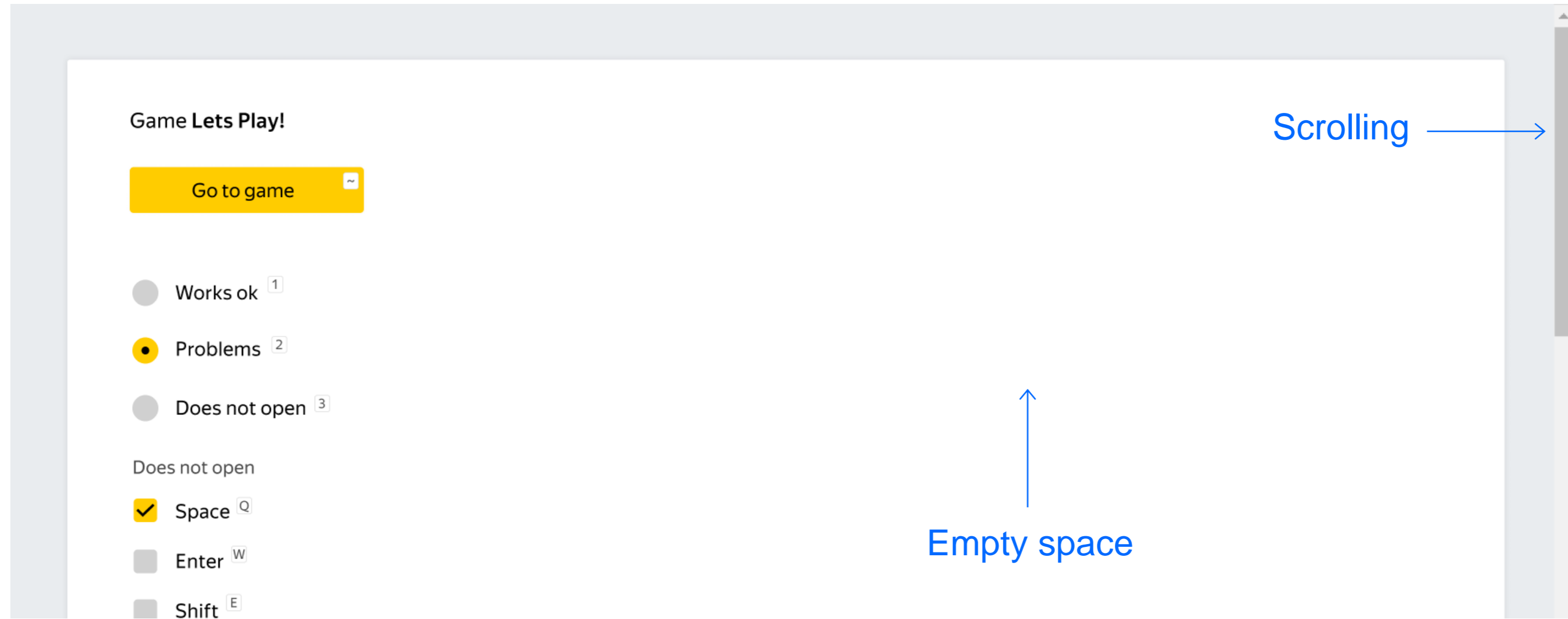
- ▶ Group the elements within your task block
- ▶ Absence of empty spaces
- ▶ Highlight most important information

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

Game Lets Play!

Go to game [~]

Works ok ¹ Problems ² Does not open ³

Keys do not work

Space ^Q Enter ^W Shift ^E

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

Query [borrow a Yota router for a week](#)

Phrase [Yota router](#)

Additionally ?

Ad headline Buy Yota router at a super price!

Text High-quality wi-fi routers! Installation and configuration. Call us!

Does the meaning of the phrase match the query?

Yes ¹ No ²

Query [should I buy an apartment now](#)

Phrase [buying an apartment](#)

Additionally ?

Ad headline Buying an apartment on Move.ru

Text Selling apartments in your city. Prices straight from the owners

Does the meaning of the phrase match the query?

Yes ¹ No ²

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

Phrase где правильно переходить улицу
Translation 1 where can I cross the street correctly
Translation 2 where can I cross the street

Check in online translators

Yandex¹ Google² Bing³ Lingvo⁴ PROMT⁵

First translation is better^Q Second translation is better^W

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

Task: evaluate which translation from Russian to English is better

Phrase где правильно переходить улицу
Translation 1 where can I cross the street correctly
Translation 2 where can I cross the street

Check in online translators



First translation is better^Q Second translation is better^W

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

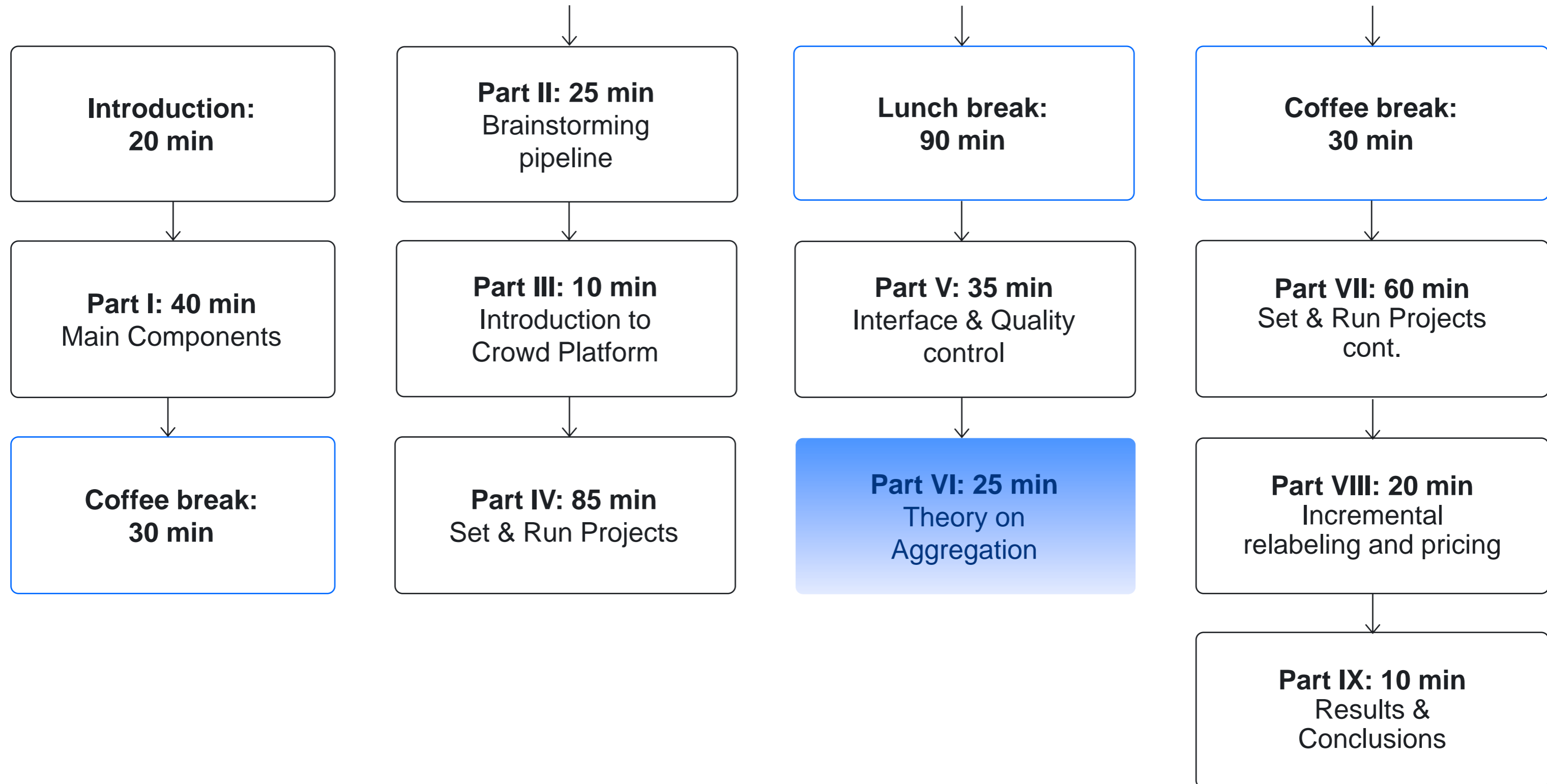
Part VI

Theory on Aggregation

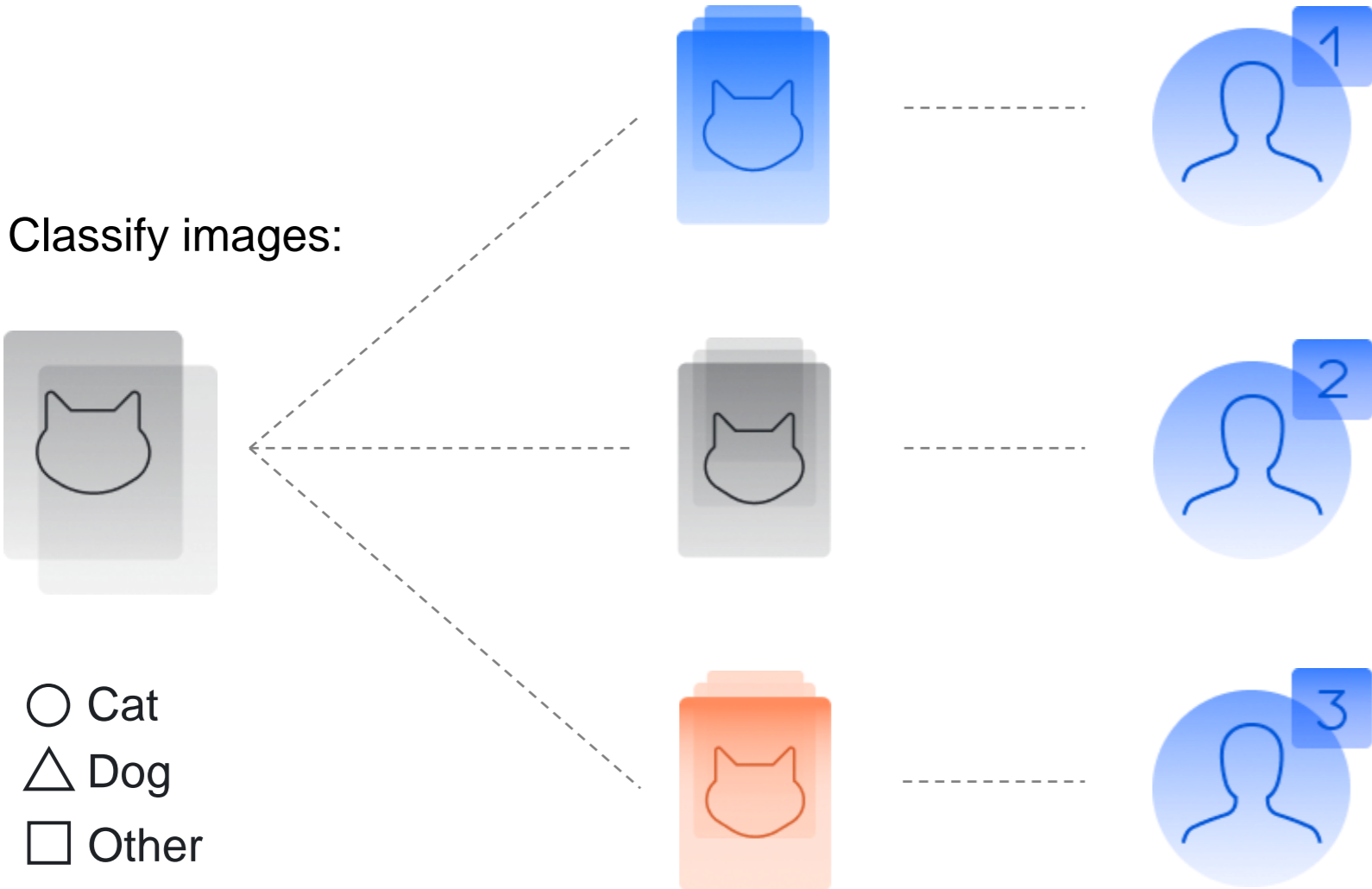
Valentina Fedorova,
Research analyst

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

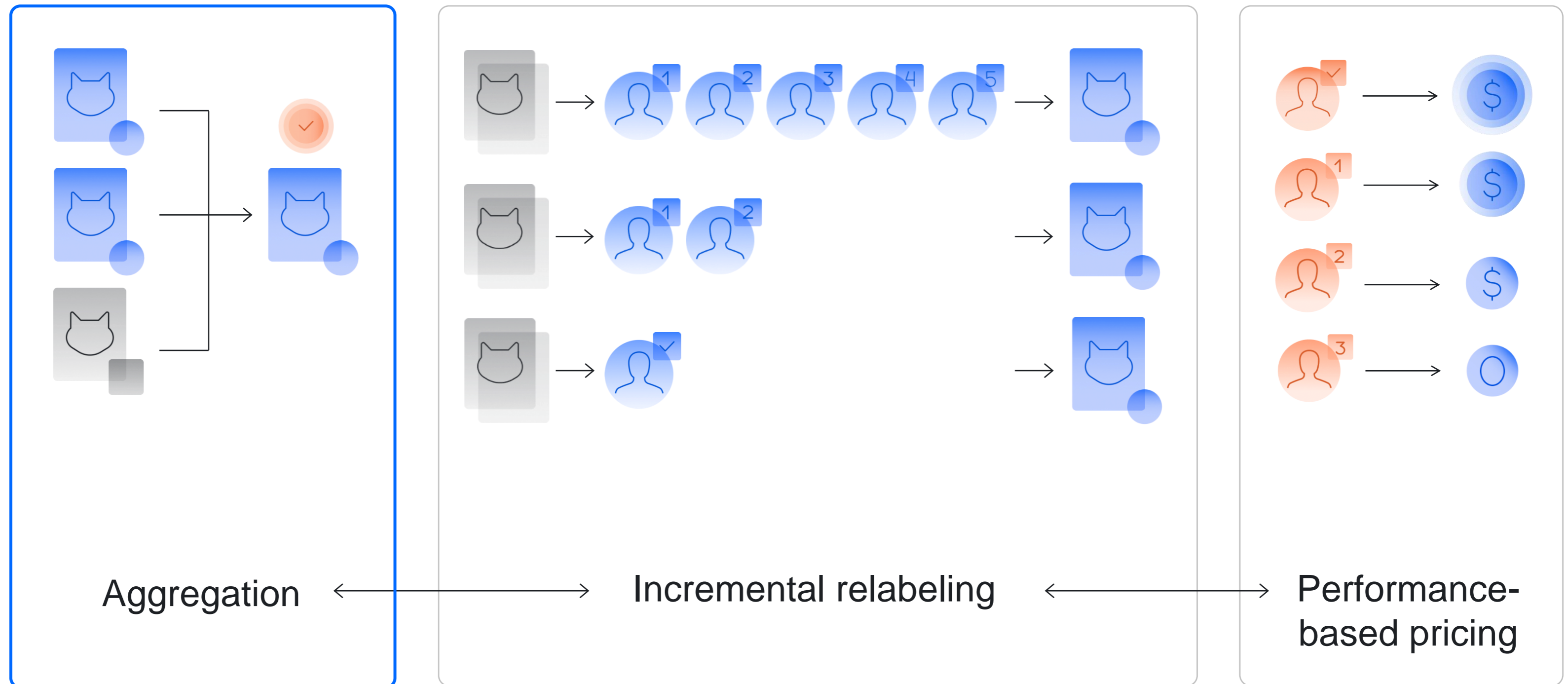


Accuracy

Cost

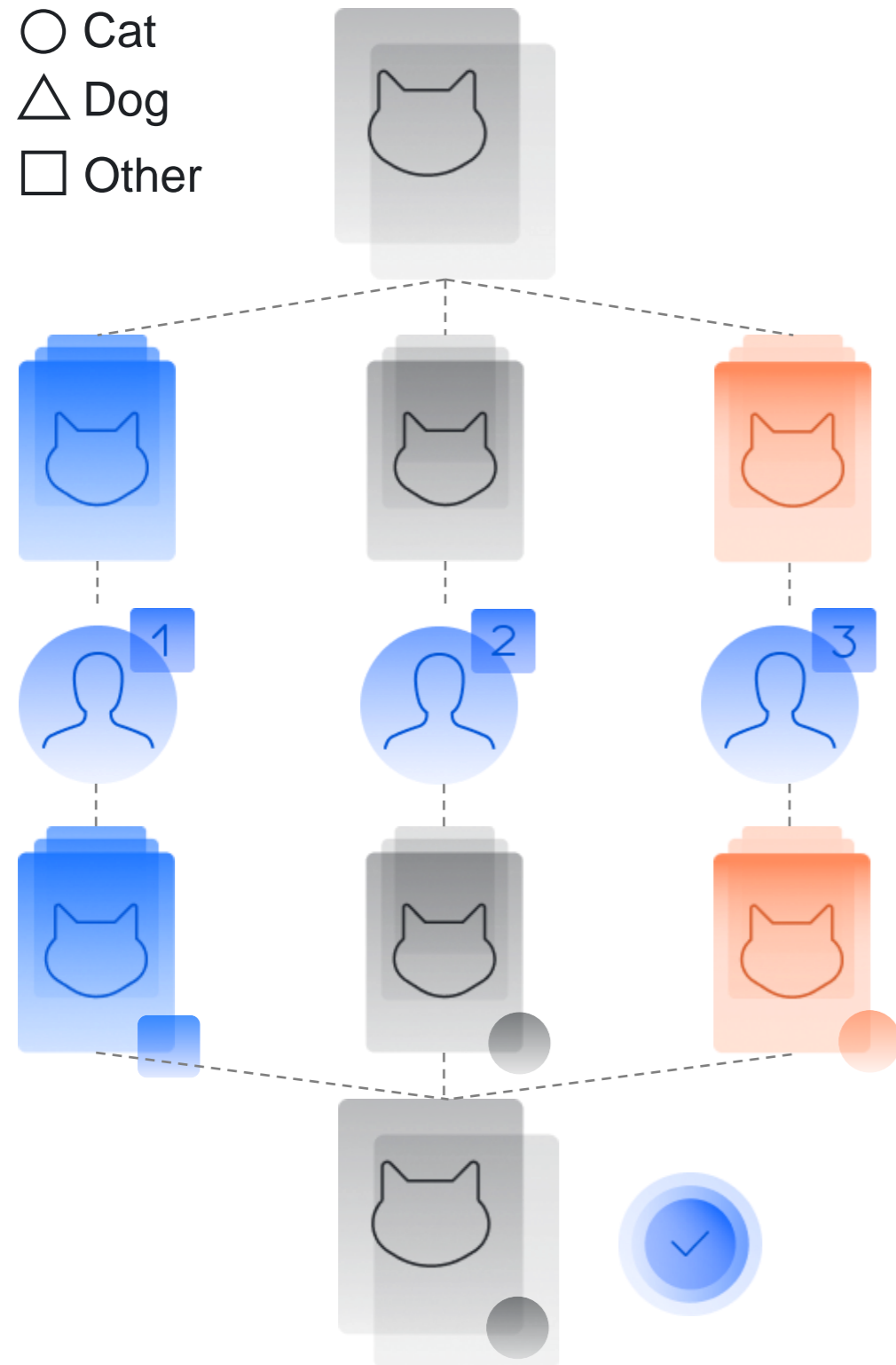
- ▶ Labels with a **maximal level of accuracy** for a **given budget**
or
- ▶ Labels of a **chosen accuracy level** for a **minimal budget**

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




Projects > Does the image contains traffic lights? > pool

 pool — closed ▲

Statistics Download results  ^ Edit  ?

View operations
Dawid-Skene aggregation model
Aggregation by skill


POOL TASKS (File example for task uploading (tsv, UTF-8)) ?

 Upload  files Edit  Preview

30 task suites	0 training task
90 tasks	10 control task

100 %
Done 30, accepted 30

View assignments

0  30

Multiclass labels

Project 1: Filter images

Are there shoes
in the picture?

Yes

No



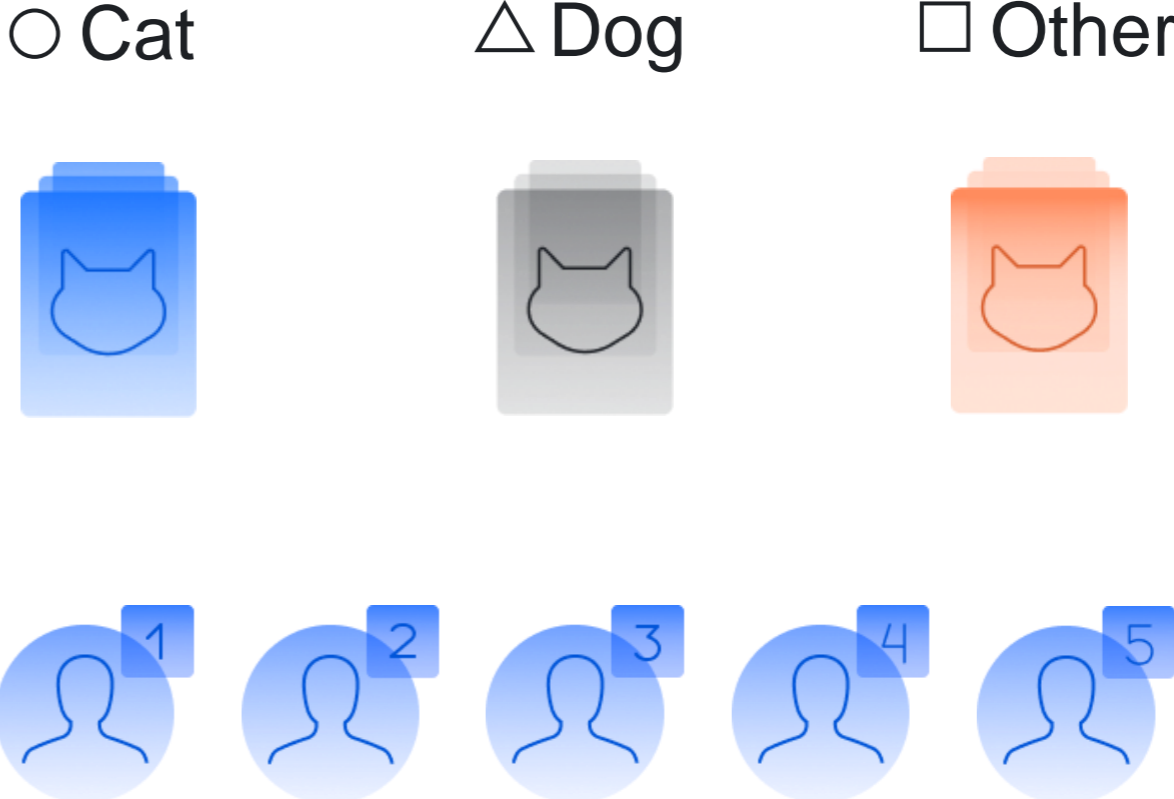
Notation

▶ Categories $k \in \{1, \dots, K\}$. E.g.:

▶ Objects $j \in \{1, \dots, J\}$. E.g.:

▶ Workers: $w \in \{1, \dots, W\}$. E.g.:

- $W_j \subseteq \{1, \dots, W\}$ — workers labeled object j



The simplest aggregation: Majority Vote (MV)

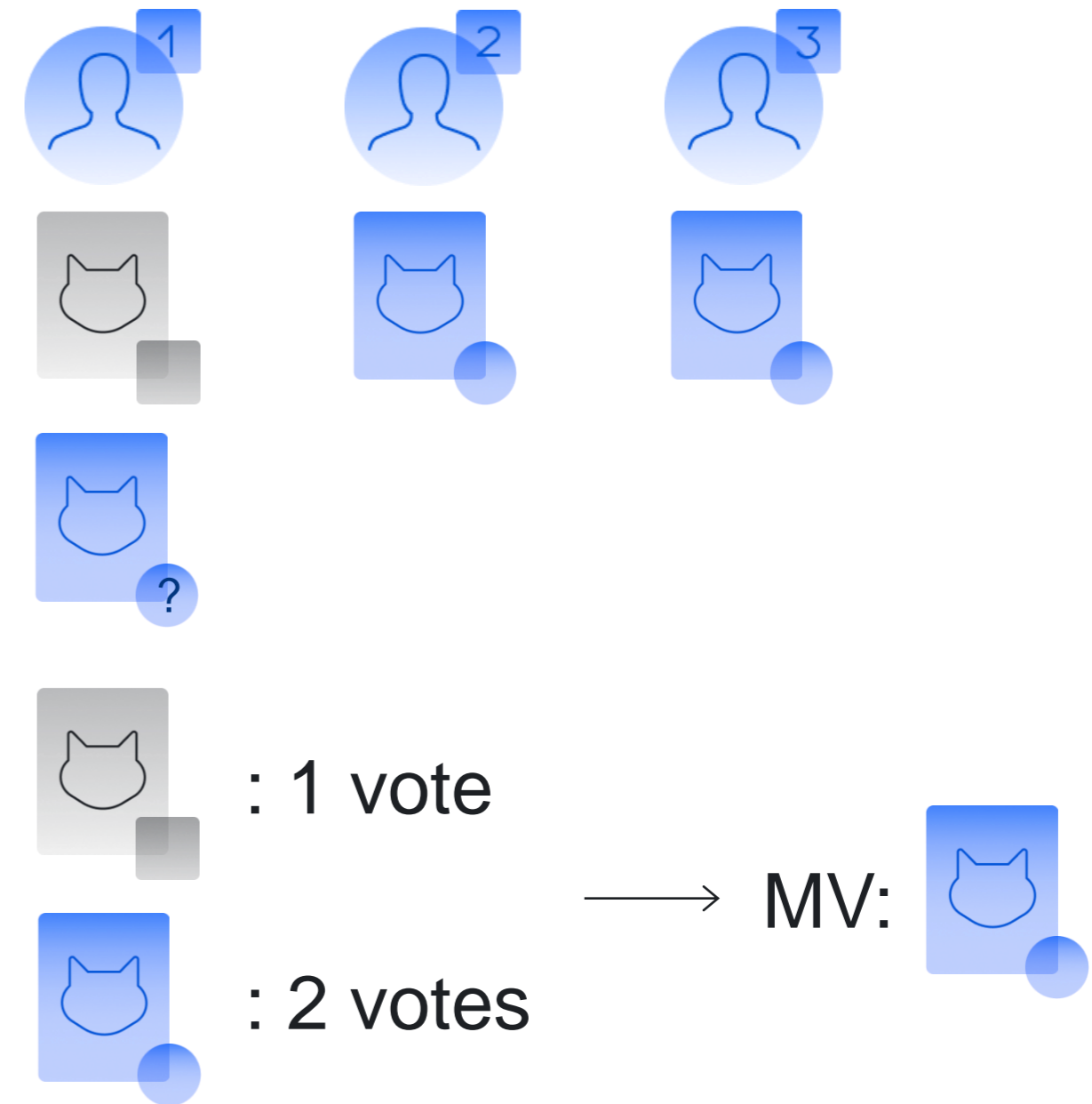
► The problem of aggregation:

- Observe noisy labels

$$y = \{y_j^w \mid j = 1, \dots, J \text{ and } w = 1, \dots, W\}$$

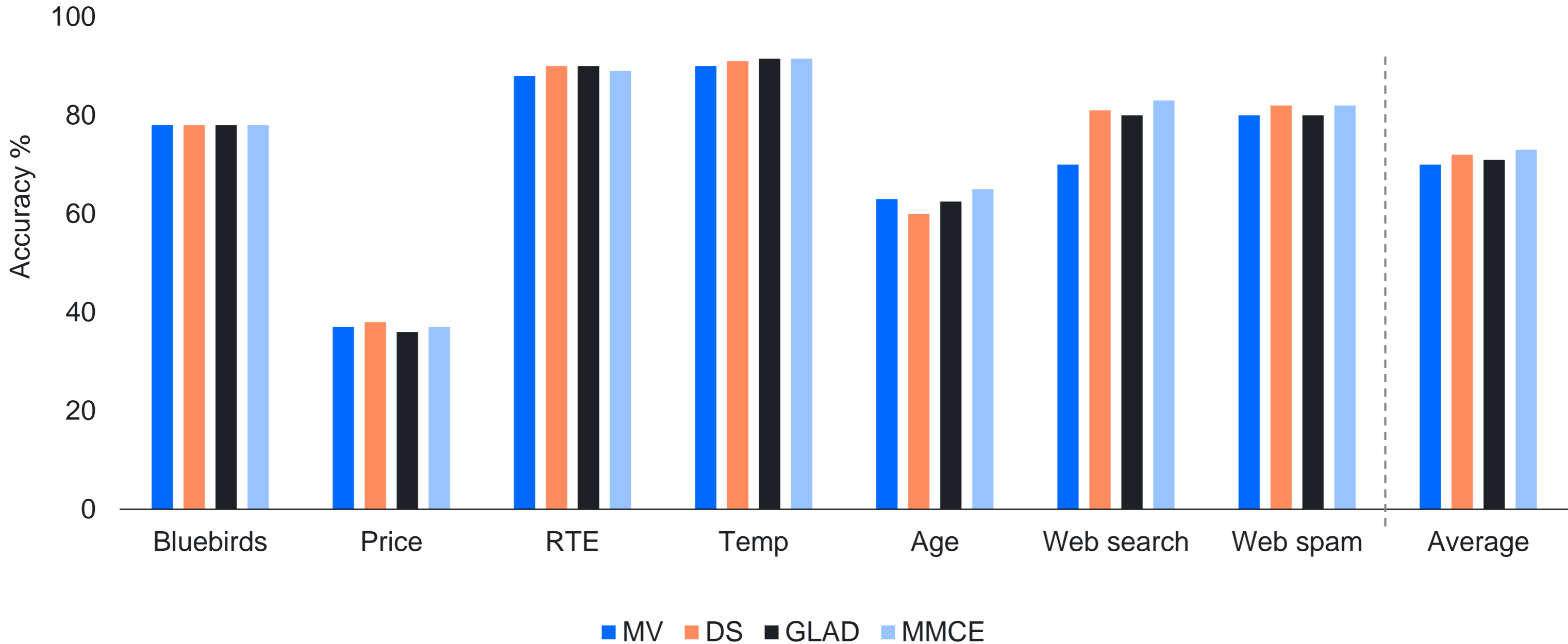
- Recover true labels $z = \{z_j \mid j = 1, \dots, J\}$

► A straightforward solution:



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

Performance of MV vs other methods

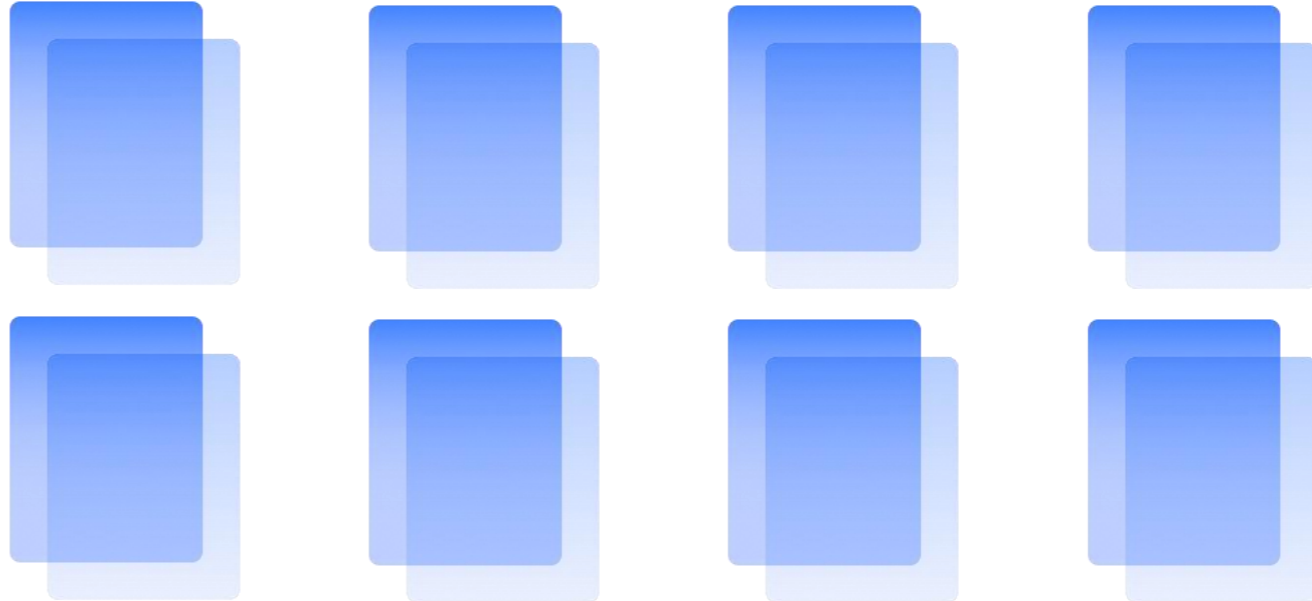


Properties of MV

All workers are treated similarly

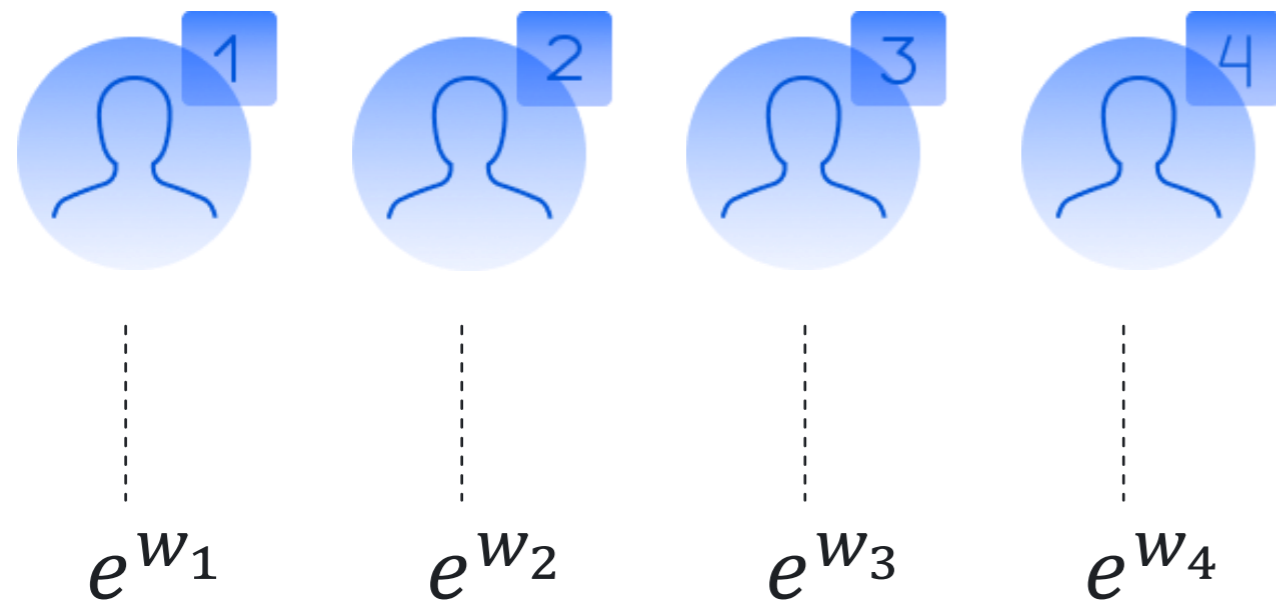


All objects are treated similarly

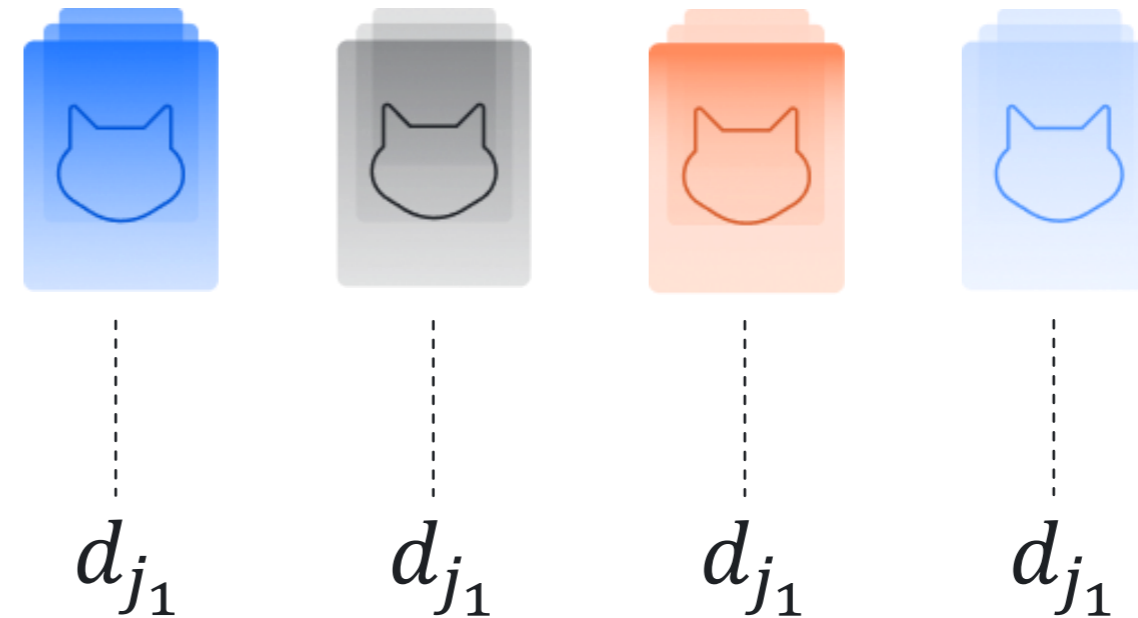


Advanced aggregation: workers and objects

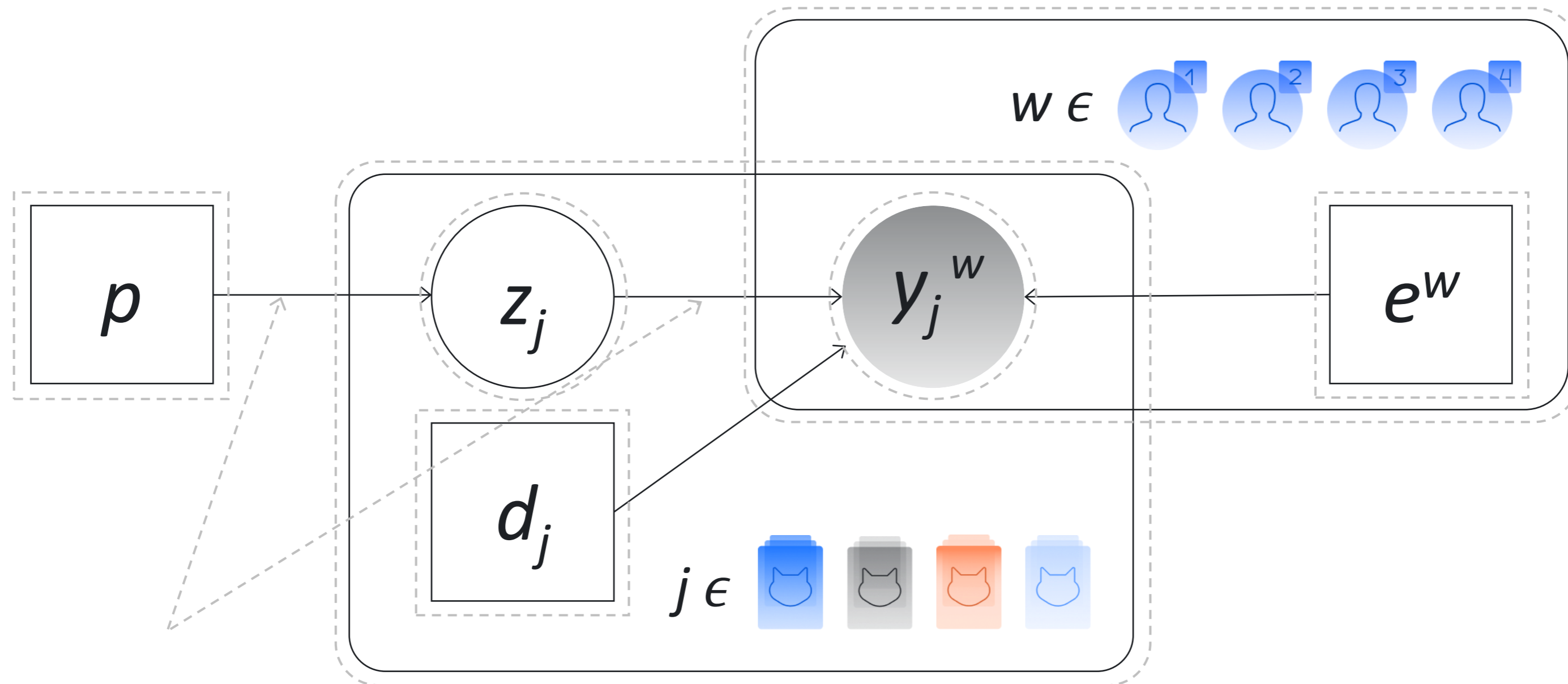
Parameterize expertise of workers by e^w



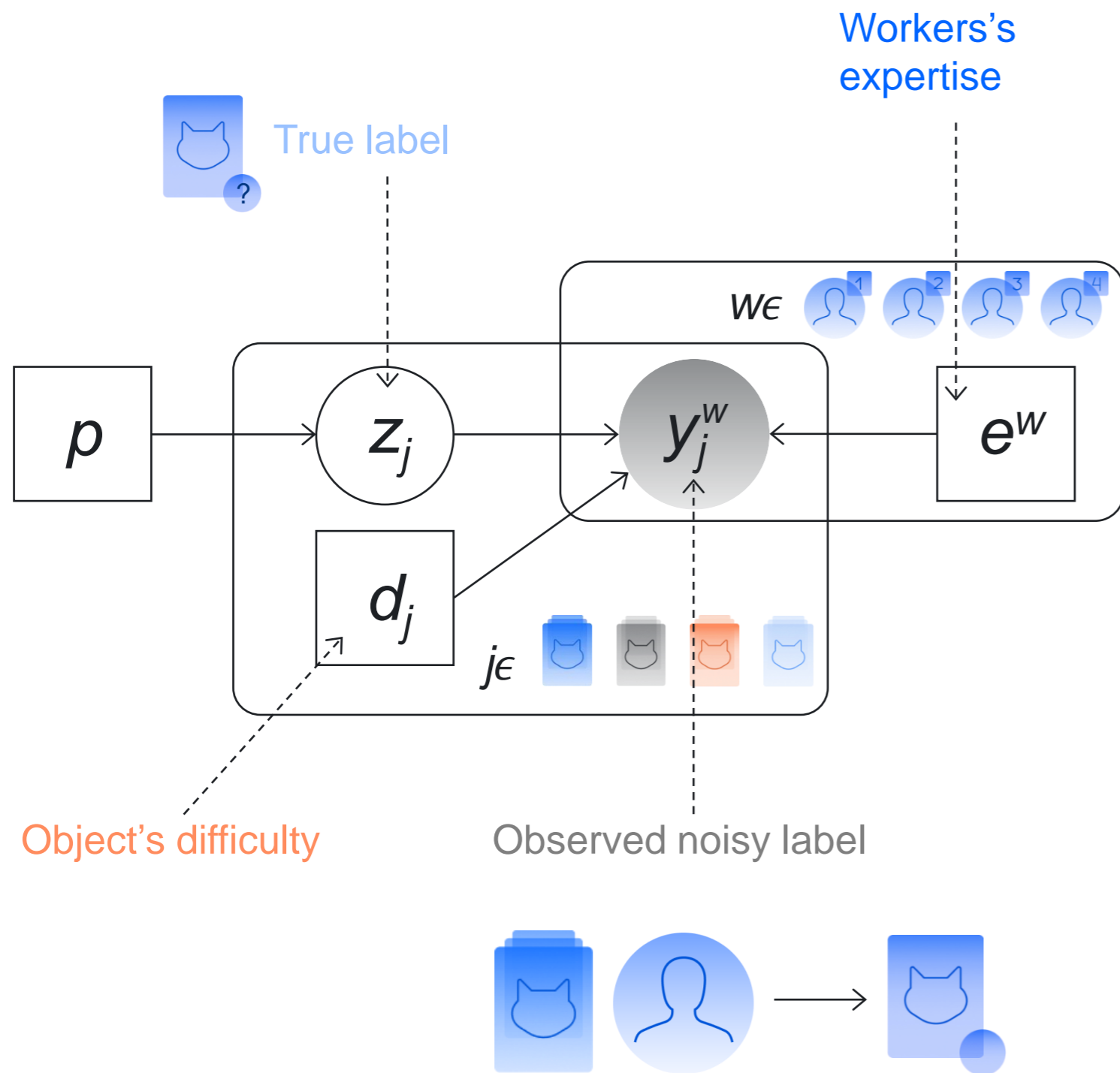
Parameterize difficulty of objects by d_j



Advanced aggregation: latent label models

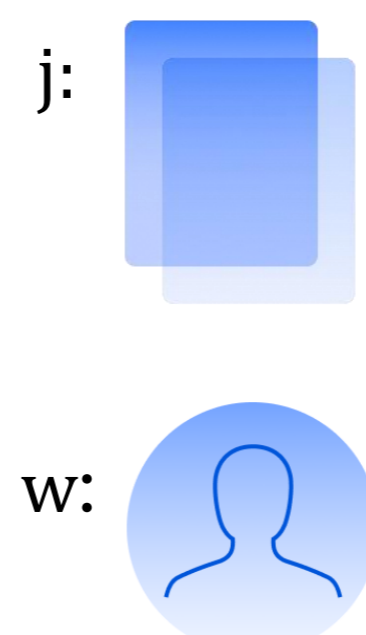


Latent label models: noisy label model



A noisy label model $M_j^w = M(e^w, d_j)$ is a matrix of size $K \times K$ with elements

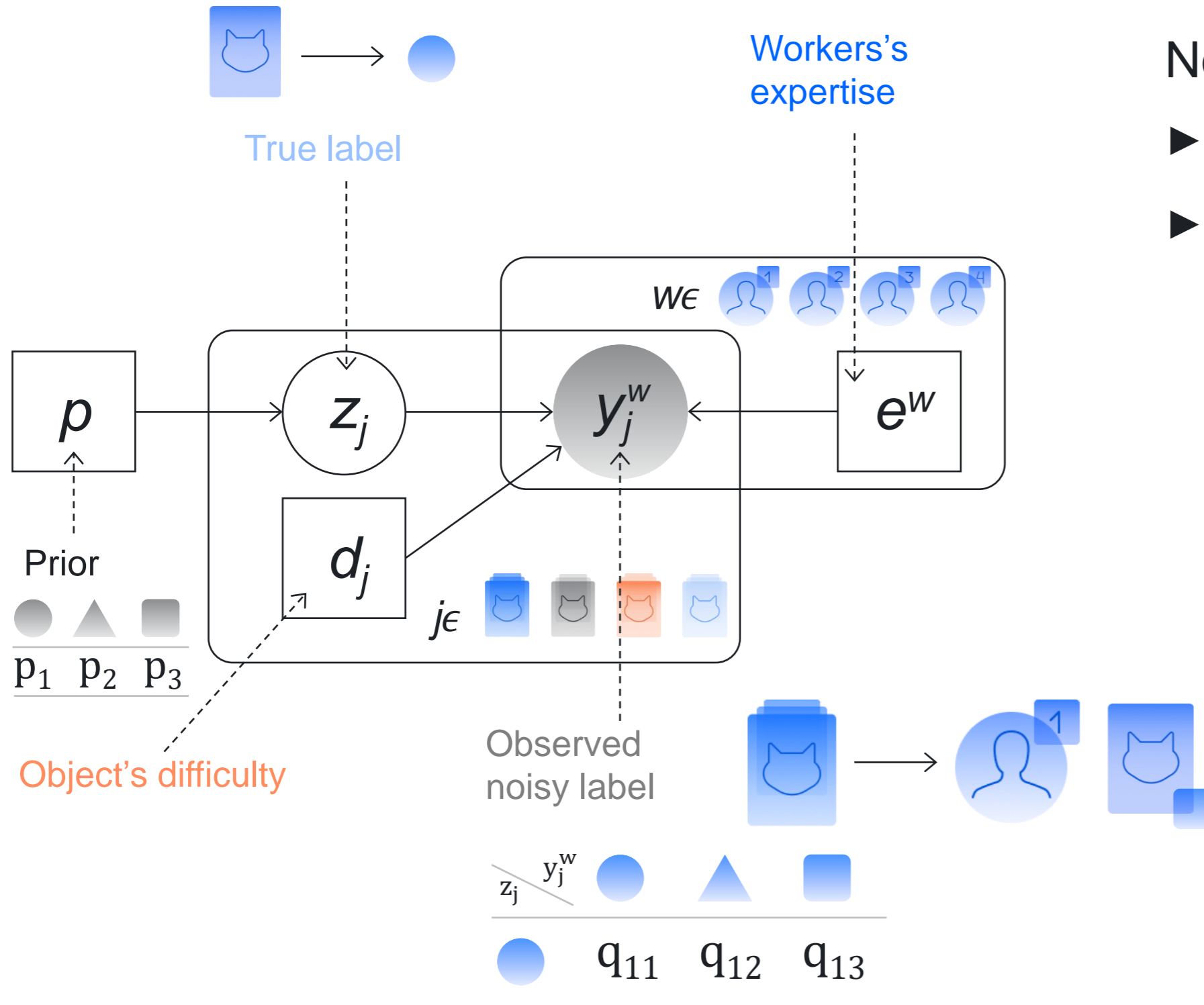
$$M_j^w[c, k] = \Pr(Y_j^w = k | Z_j = c)$$



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

$$q_{c1} + q_{c2} + q_{c3} = 1 \text{ for each } c$$

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 $\text{Mult}(\cdot)$

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:

$$L\left(\underbrace{\{z_j\}_{j=1}^J}_{\text{Latent true label}}, p, \underbrace{\{d_j\}_{j=1}^J, \{e^w\}_{w=1}^W}_{\text{Latent parameters}}\right) = \underbrace{\prod_{j \in J} \sum_{z_j \in \{1, \dots, K\}} \Pr(z_j | p) \prod_{w \in W_j} \Pr(y_j^w | z_j, d_j, e^w)}_{\text{Likelihood of noisy and true labels for object } j}$$

Observed noisy label

- ▶ Estimate parameters and true labels by maximizing $L(\dots)$

Latent label models: EM algorithm

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

$$\mathbb{E}_{\mathbf{z}} \log \Pr(\mathbf{y}, \mathbf{z}) = \sum_{j \in J} \sum_{z_j \in \{1, \dots, K\}} \Pr(z_j | p) \log \prod_{w \in W_j} \Pr(z_j | p) \Pr(y_j^w | z_j, \mathbf{d}_j, \mathbf{e}^w)$$

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

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

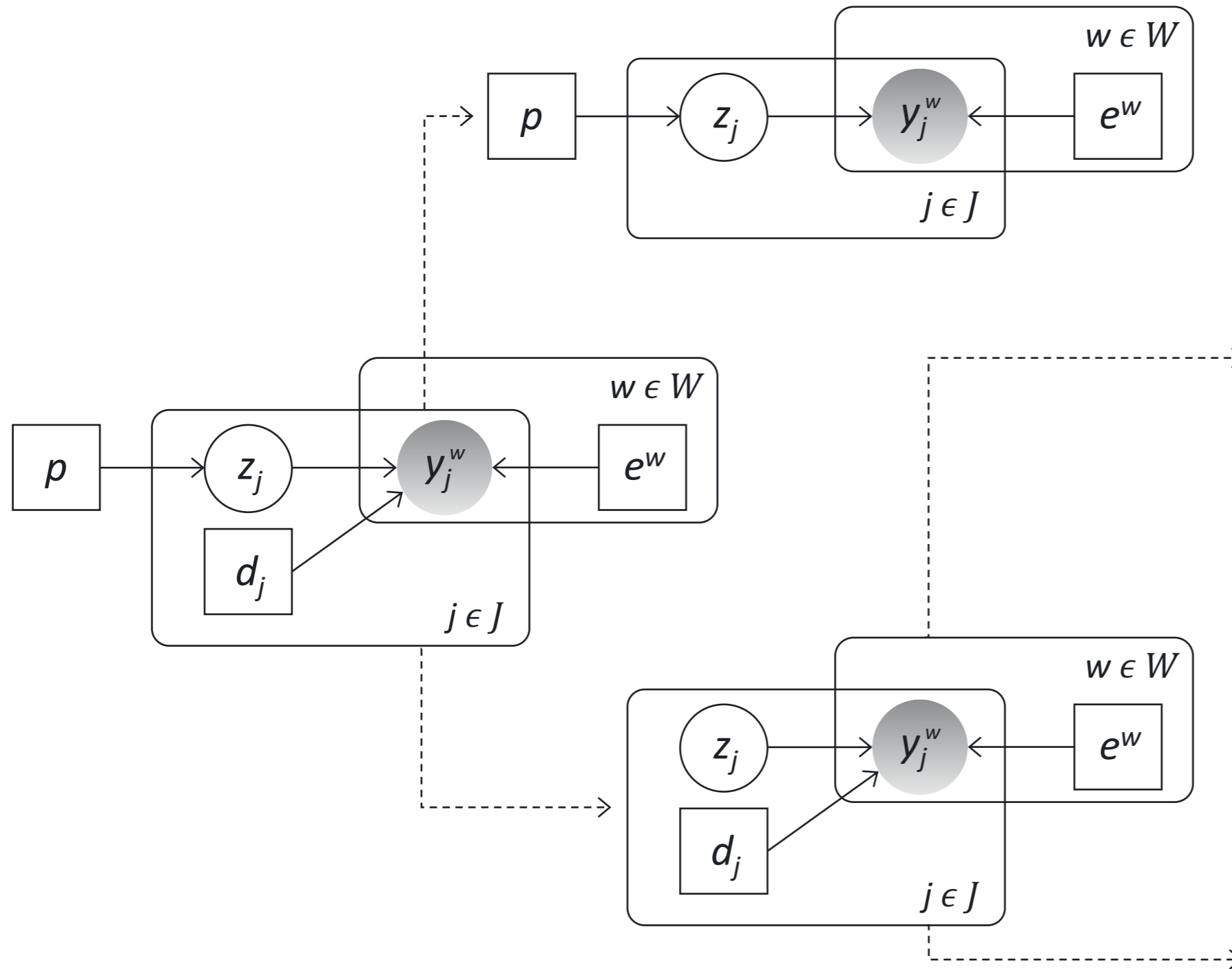
- ▶ **M-step:** Maximize the expectation of LL with respect to the posterior distribution of \hat{z} :

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

- Analytical solutions
- Gradient descent

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

Latent label model (LLM): special cases

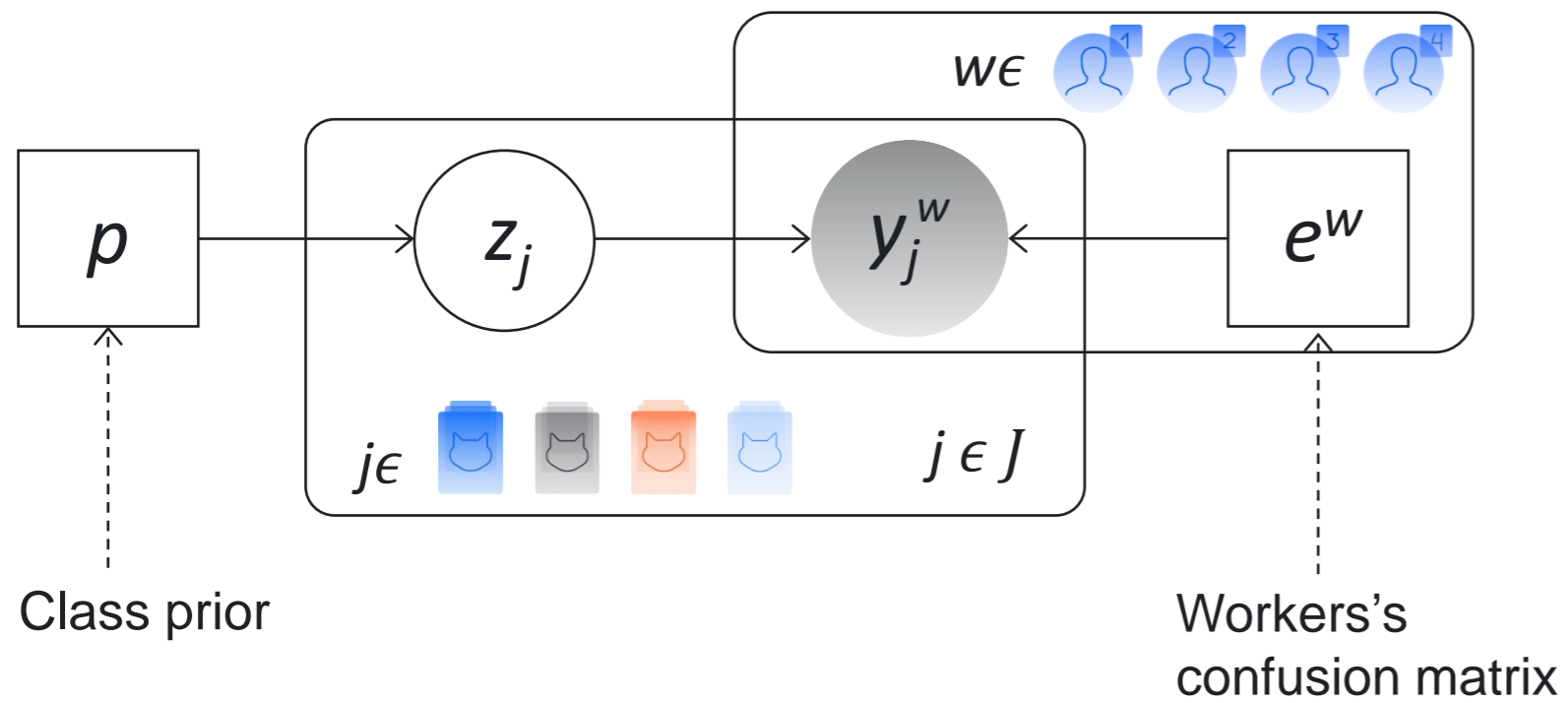


- ▶ Dawid and Skene model (DS):
 - Categories are **different**
 - Objects are **similar**
 - Workers are **different**

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

$z \backslash y_w$	●	▲	■
●	■	■	■
▲	■	■	■
■	■	■	■

DS: parameters optimization

► **E-step:**

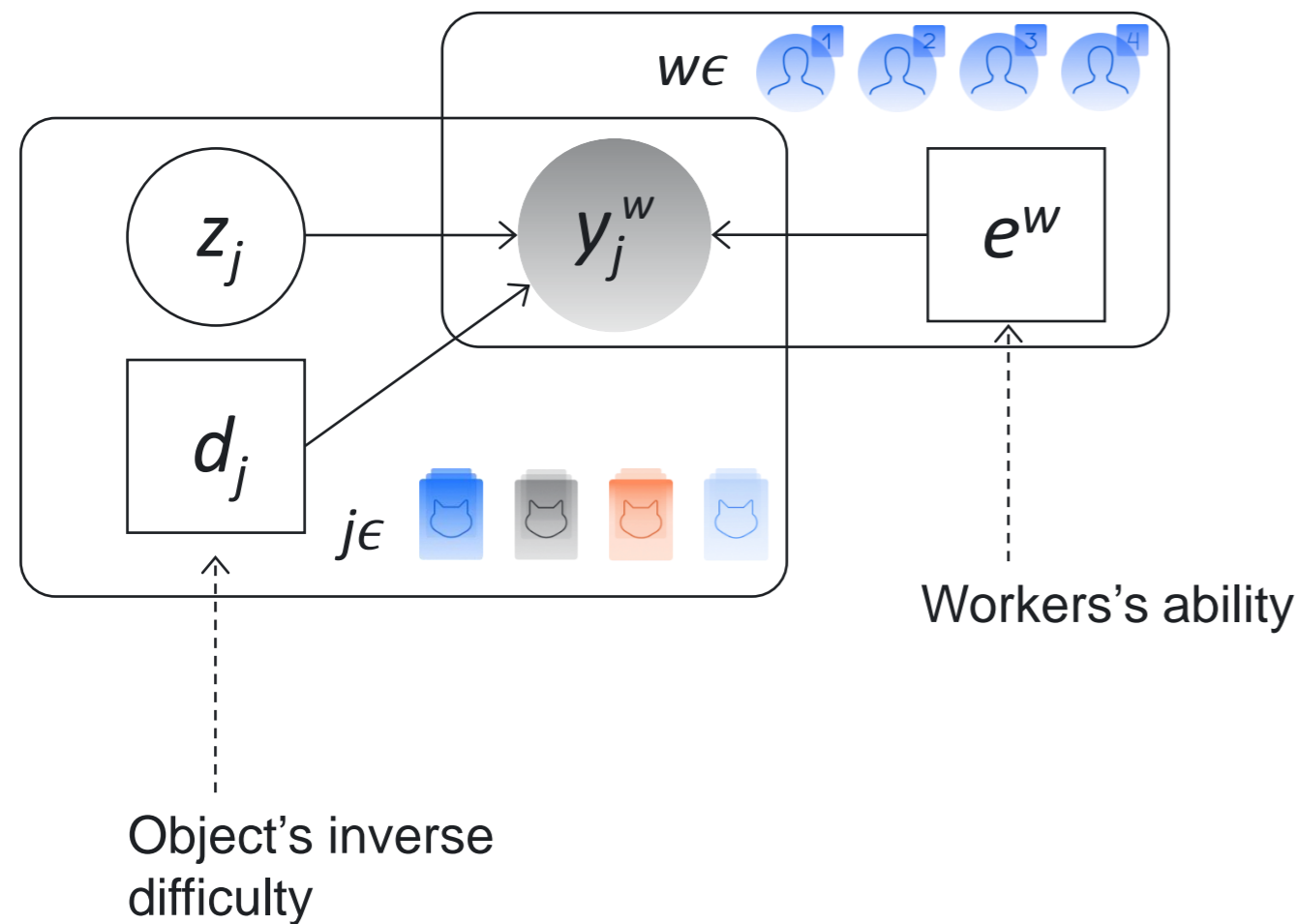
$$\hat{z}_j[c] = \frac{p[c] \prod_{w \in W_j} e^w[c, y_j^w]}{\sum_k p[k] \prod_{w \in W_j} e^w[k, y_j^w]}, \quad c = 1, \dots, K$$

► **M-step:** Analytical solution

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

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

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



LLM with parameters:

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

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

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

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., $P(z_j) = 1/K$)

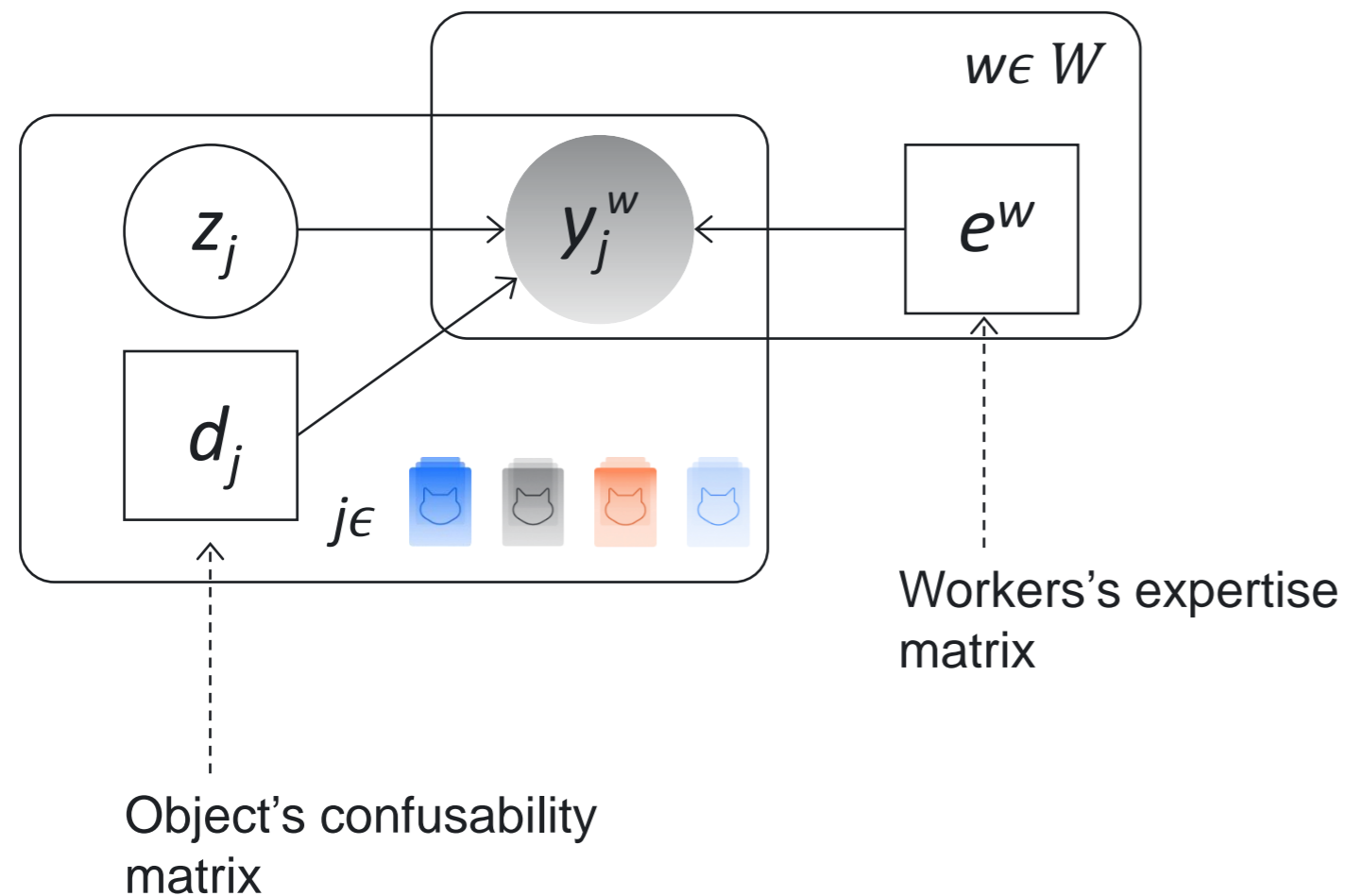
► **E-step:**

$$\hat{z}_j [c] \propto P(Z_j = c) \prod_{w \in W_j} a(w, j)^{\delta(y_j^w = c)} \left(\frac{1 - a(w, j)}{K - 1} \right)^{\delta(y_j^w \neq c)}, \quad c = 1, \dots, K$$

► **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)


























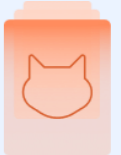












- Find parameters that minimize the maximum conditional entropy of observed labels:

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

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

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

Summary of aggregation methods

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

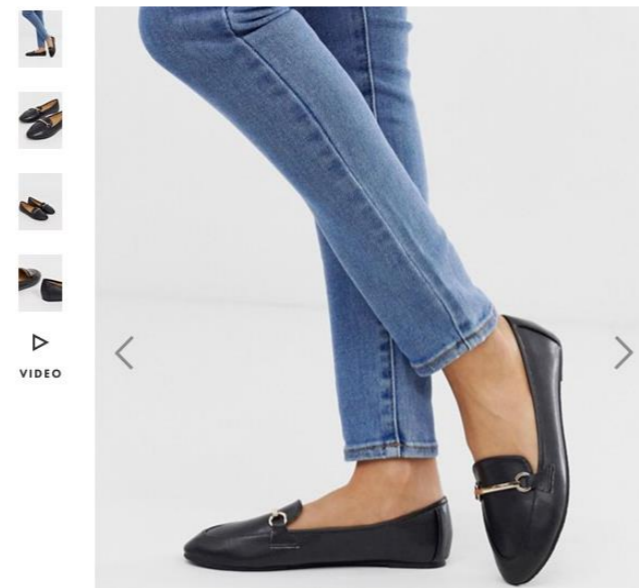
Pairwise comparisons

Project 4: Compare items

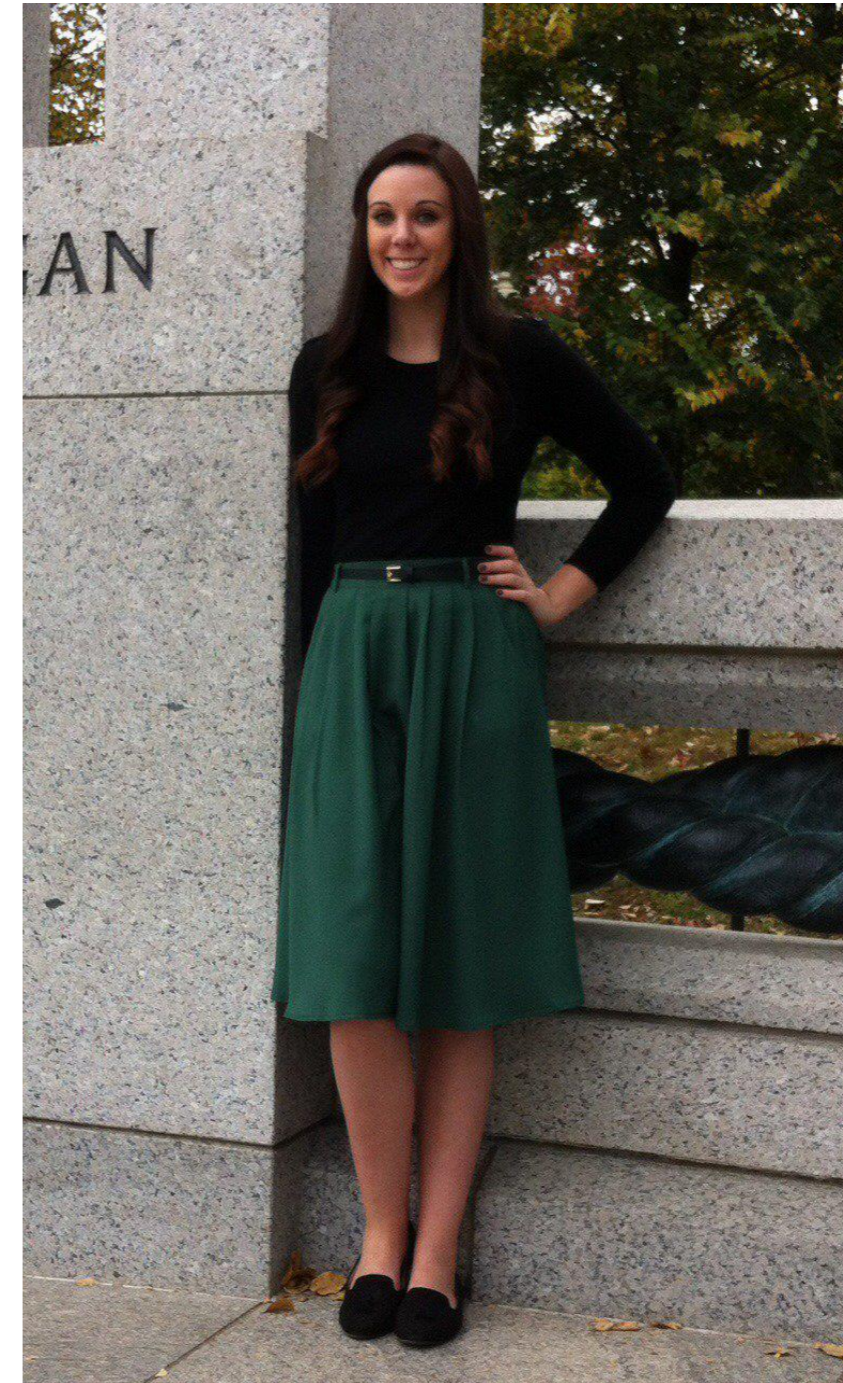
Which shoes look more similar to the one in the picture?



Left



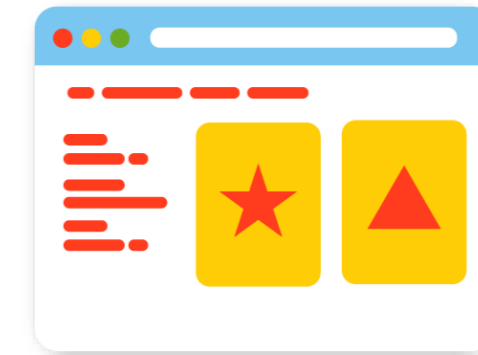
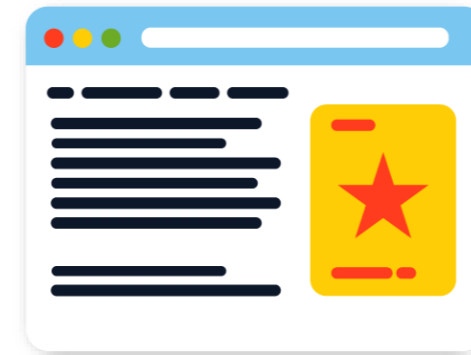
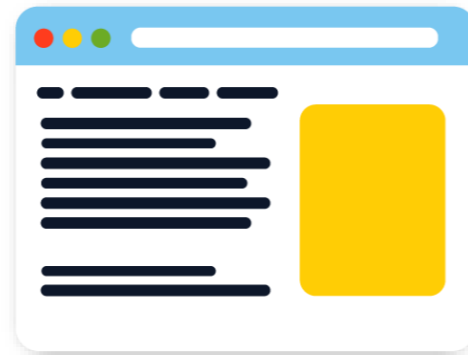
Right



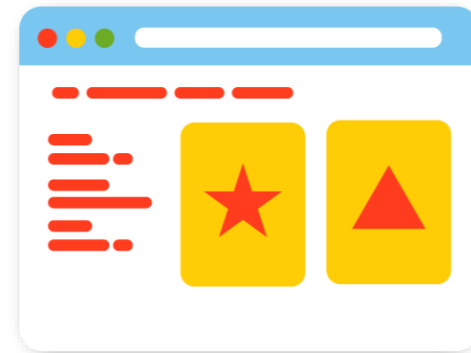
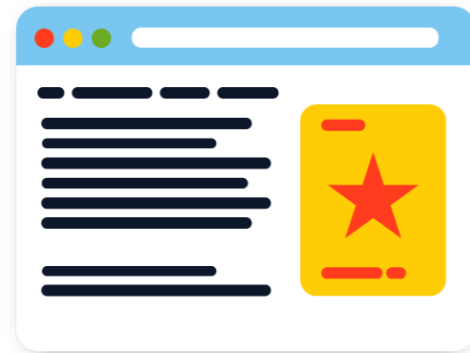
Notation

▶ Answers: **Left** or **Right**

▶ Items $d_j \in \{1, \dots, N\}$ E.g.:



▶ Tasks:



Choose a better item:

Left
Right

▶ Workers $w \in \{1, \dots, W\}$ E.g.:



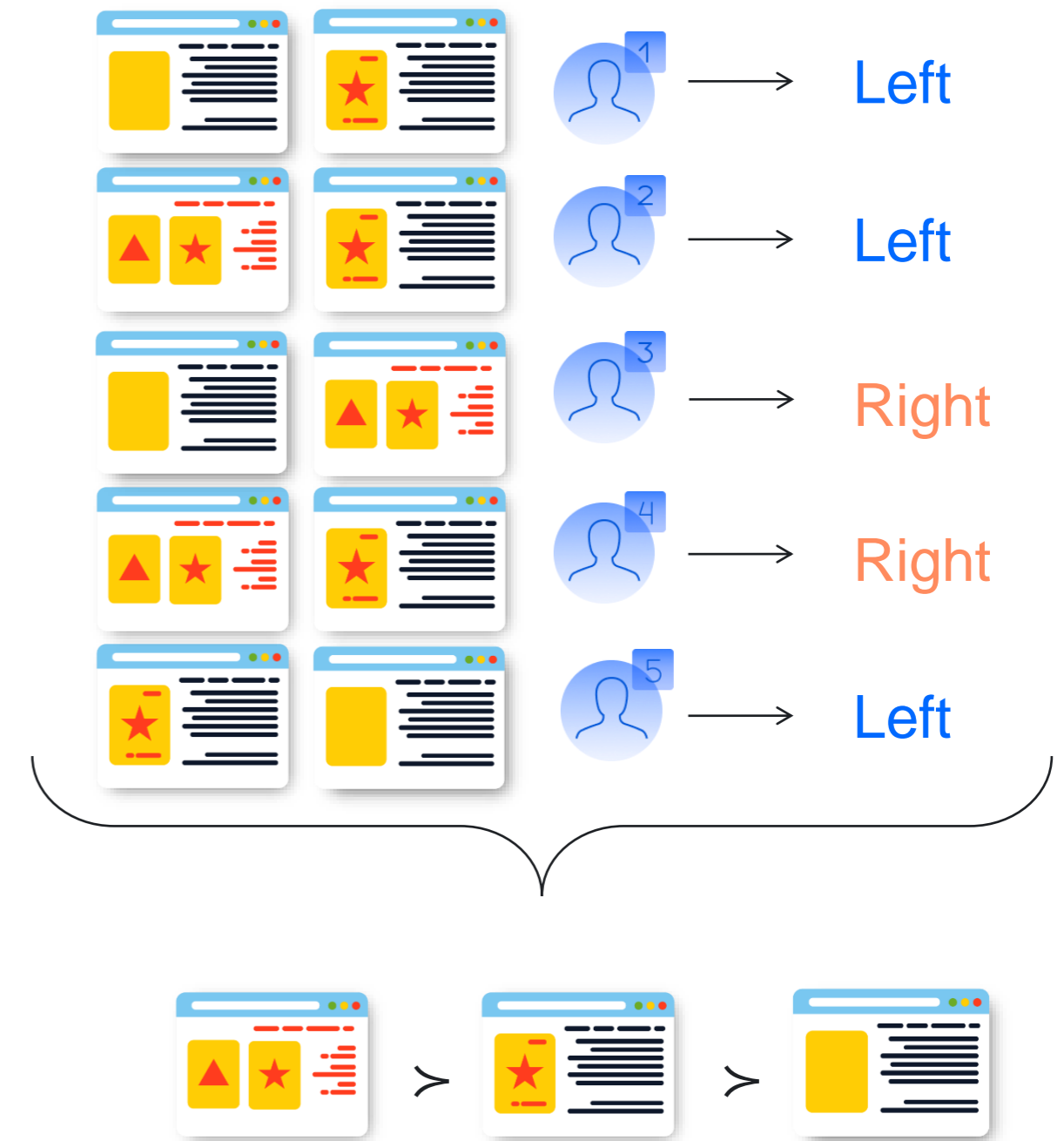
Formalization

Ranking from pairwise comparisons:

- ▶ Given pairwise comparisons for items in D :

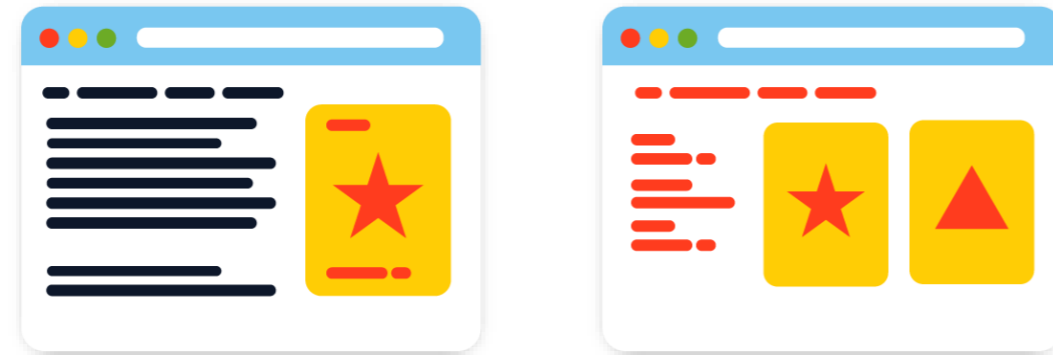
$$P = \{(w_k, d_i, d_j) : i \succ_k j\}$$

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



Difference from multiclassification

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

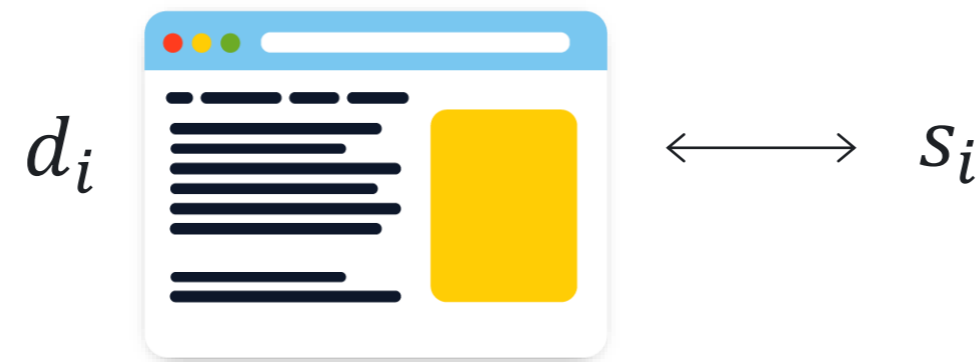


- ▶ Different tasks may contain common items



Bradley and Terry model (BT)

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



- ▶ 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

NoisyBT model: parameterization of workers

w_k  \longleftrightarrow “reliability” γ_k and “bias” q_k

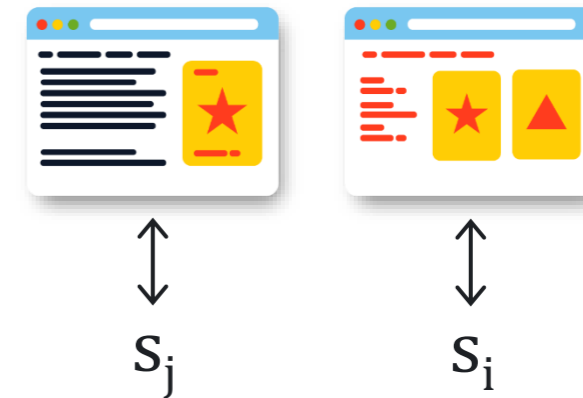
- ▶ The probability that w reads task is

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

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

$$(f(s_i - s_j), f(s_j - s_i))$$

Probability to choose **Left** if compares items



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

$$(f(q_k), f(-q_k))$$

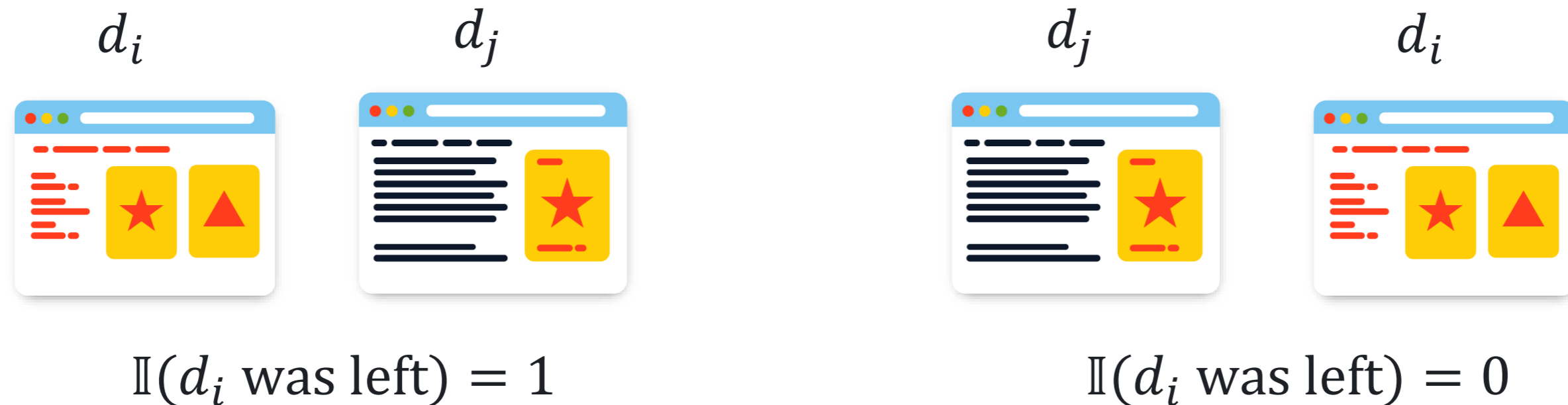
Probability to choose **Left** if answers randomly

NoisyBT: likelihood of workers' answers

The likelihood of $i \succ_k j$ is

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

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



NoisyBT: parameters optimization

Likelihood of observed comparisons:

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

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

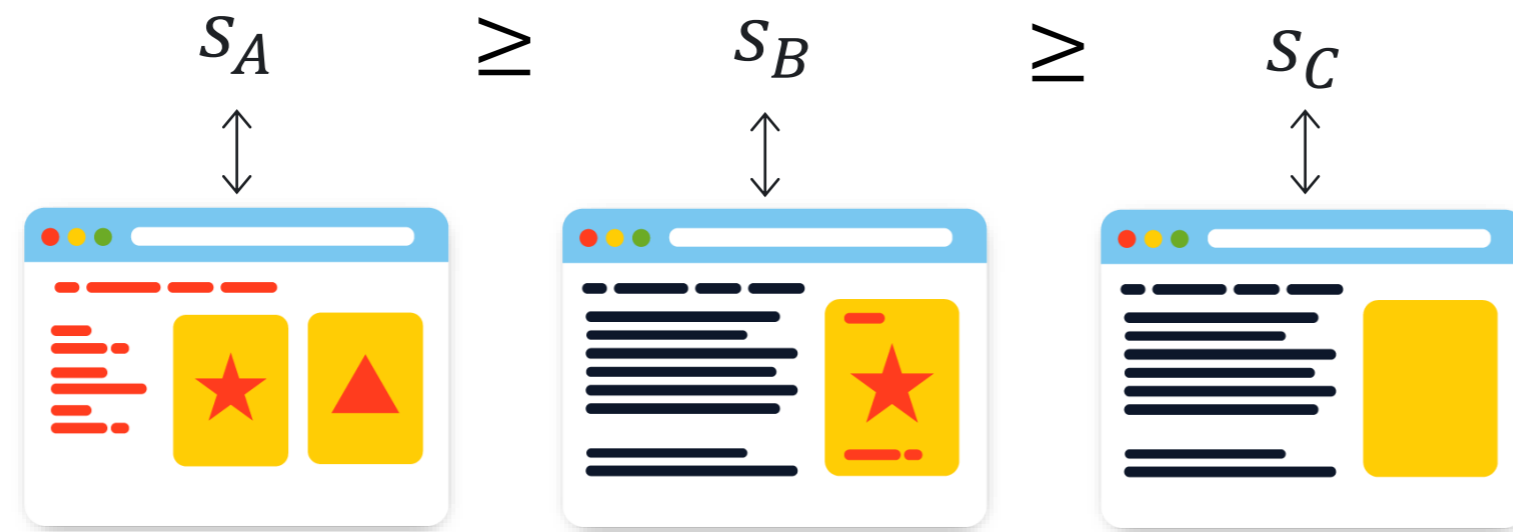
- ▶ $\{s_i\}_{i=1, \dots, N}$ and $\{\gamma_k, q_k\}_{k=1, \dots, W}$ are inferred by maximizing the log-likelihood:

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

- ▶ 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



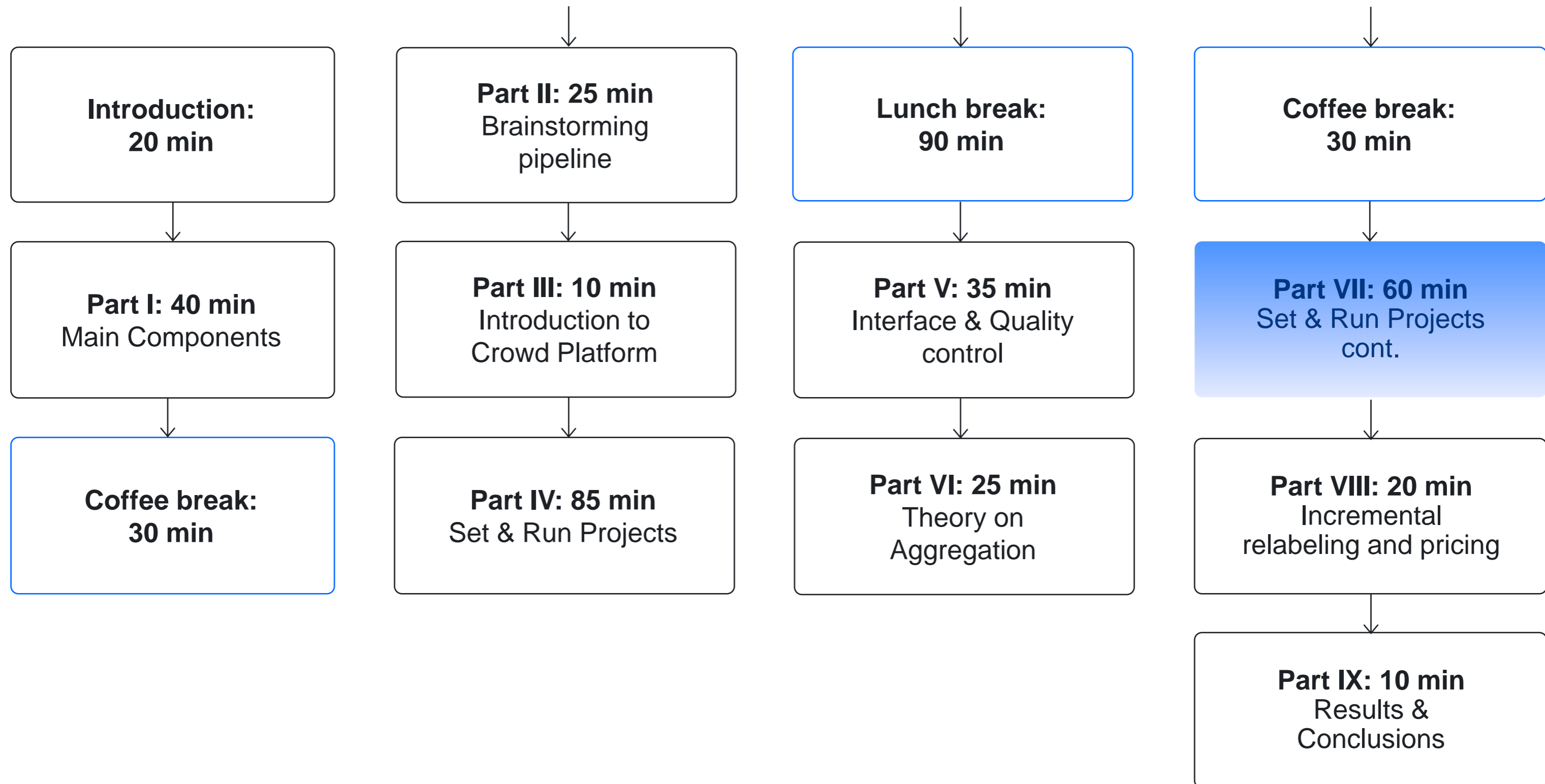
Part VII

Setting up and running label collection projects cont.

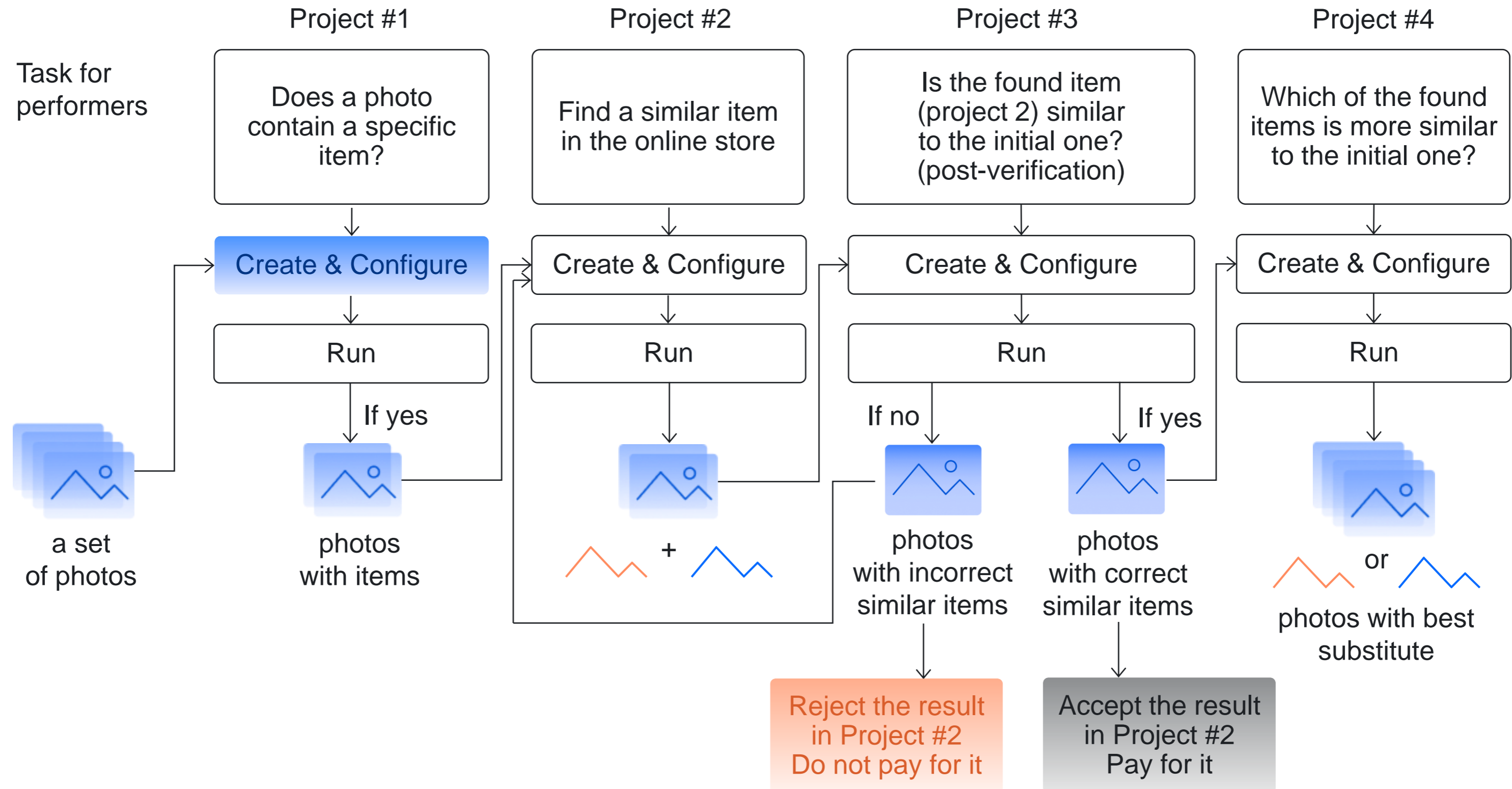
Daria Baidakova,
Project Manager

Toloka

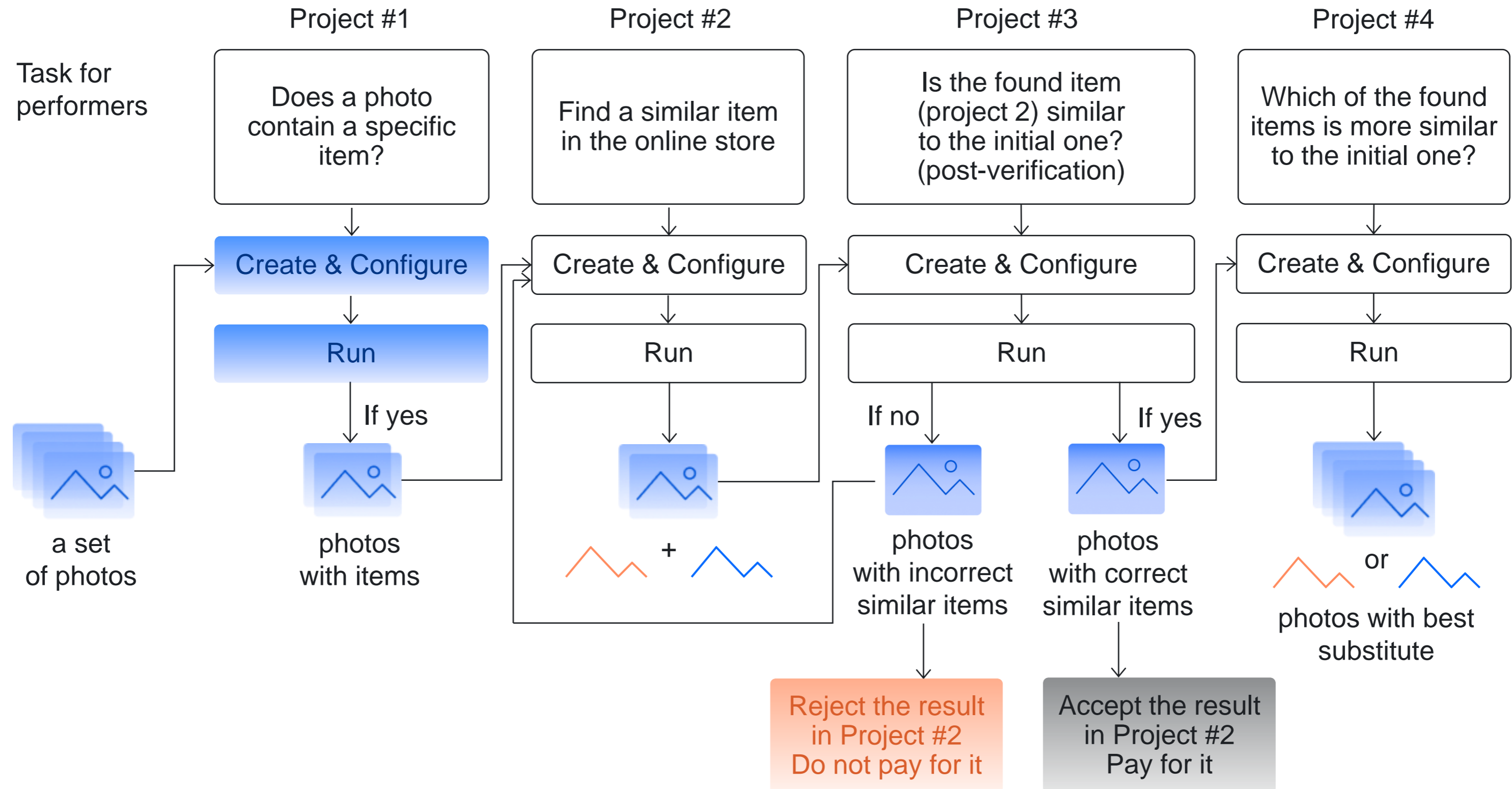
Tutorial outline



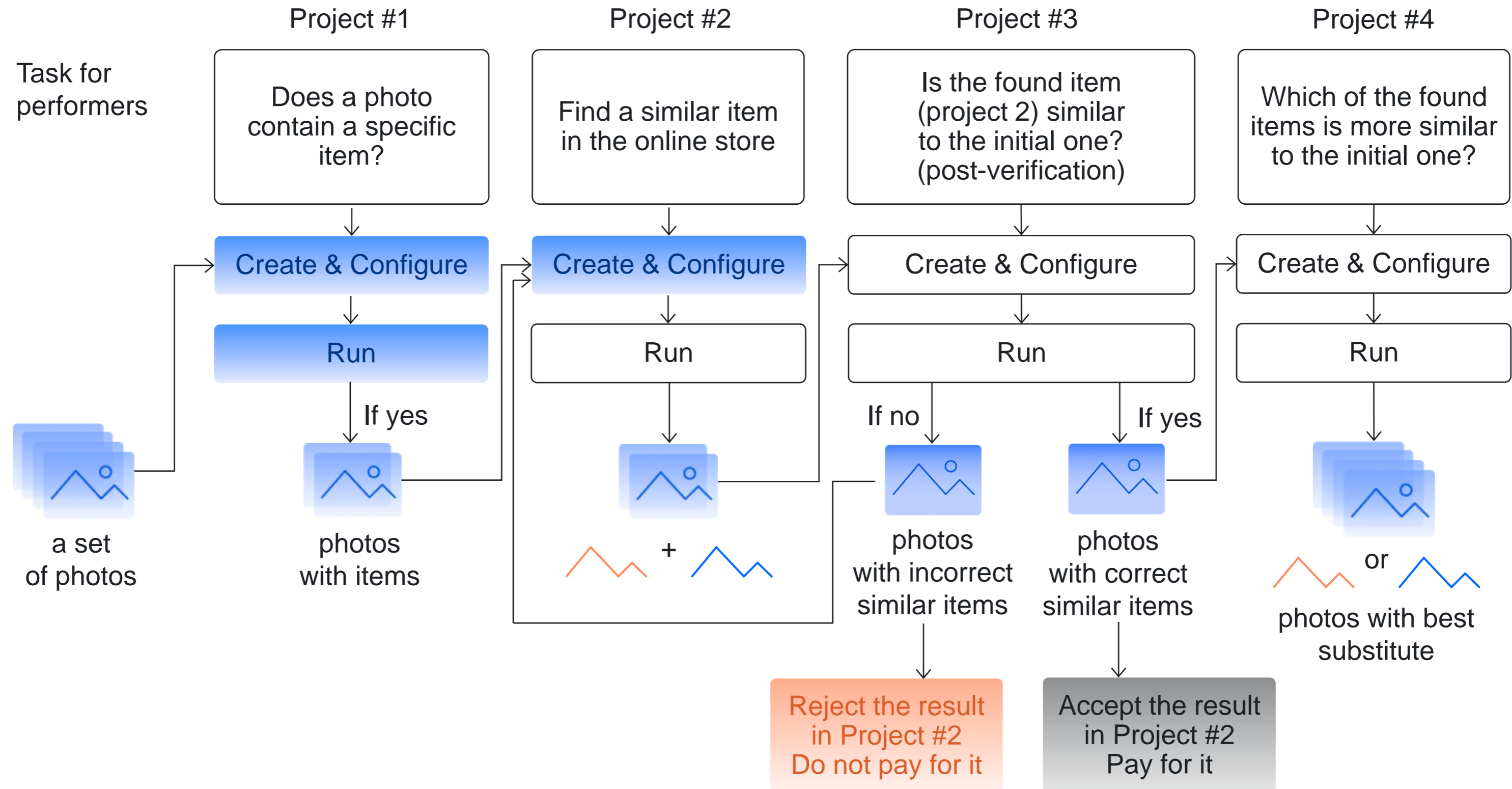
Most of us are at this step



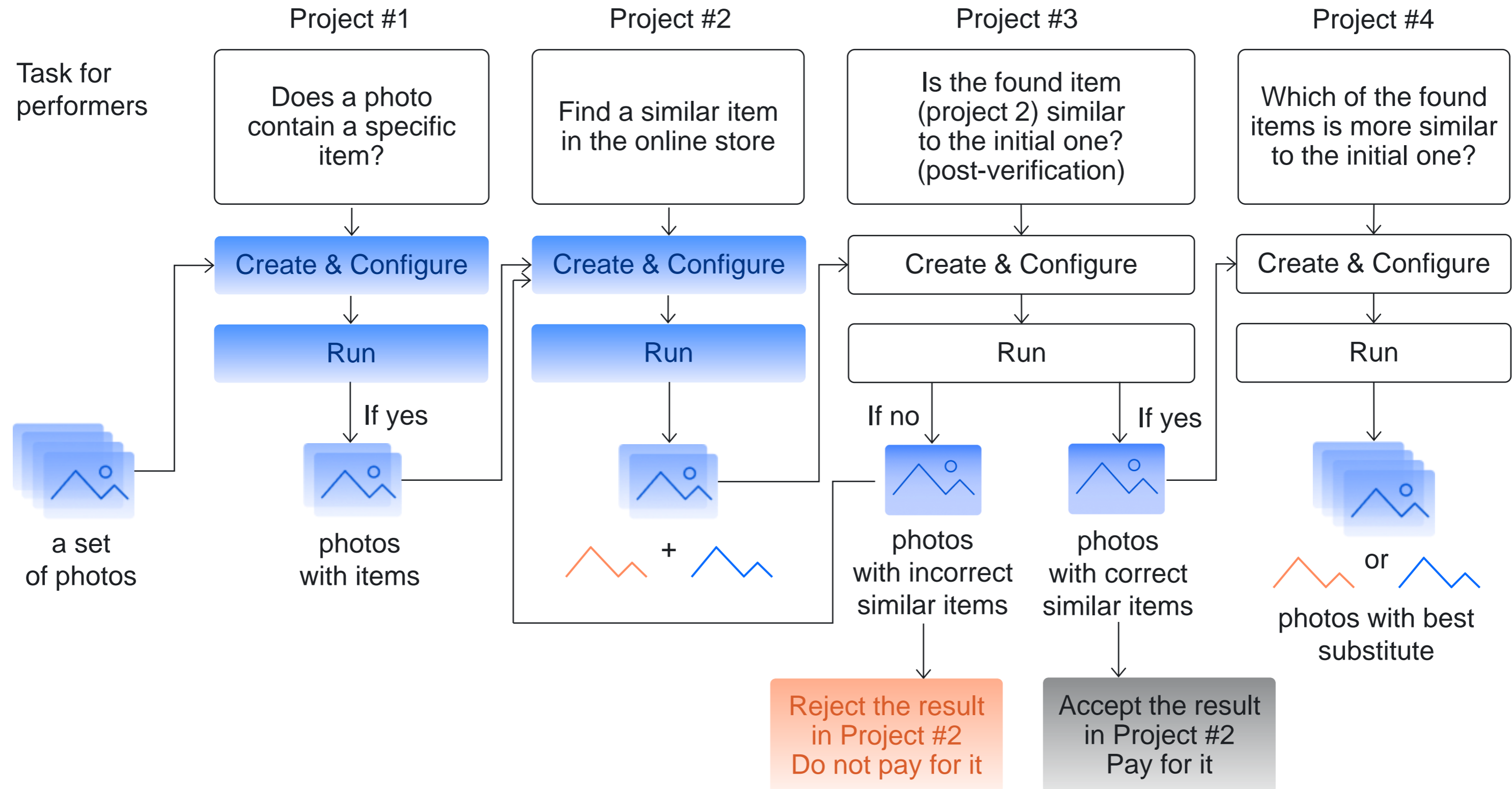
Most of us are at this step



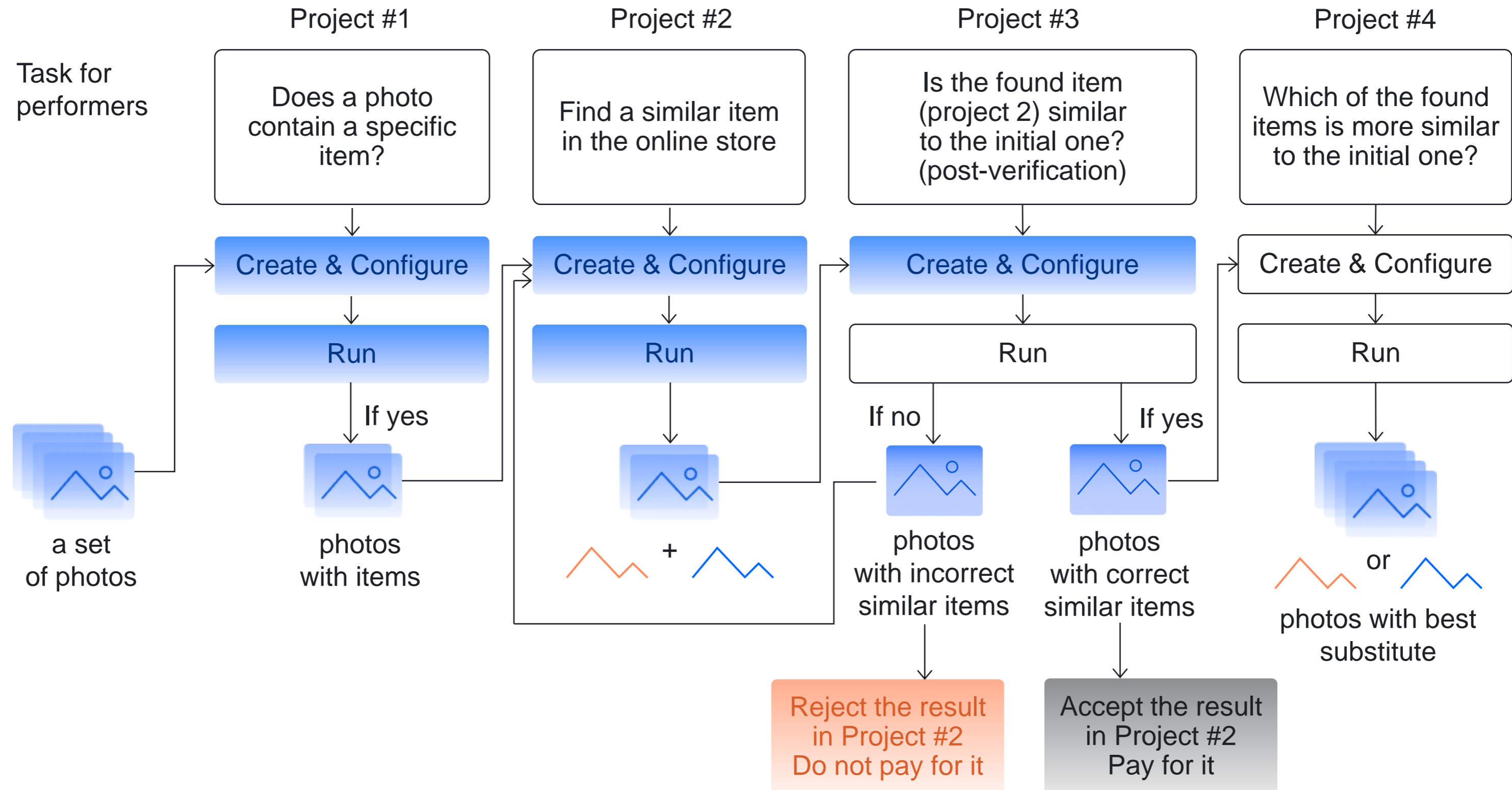
Most of us are at this step



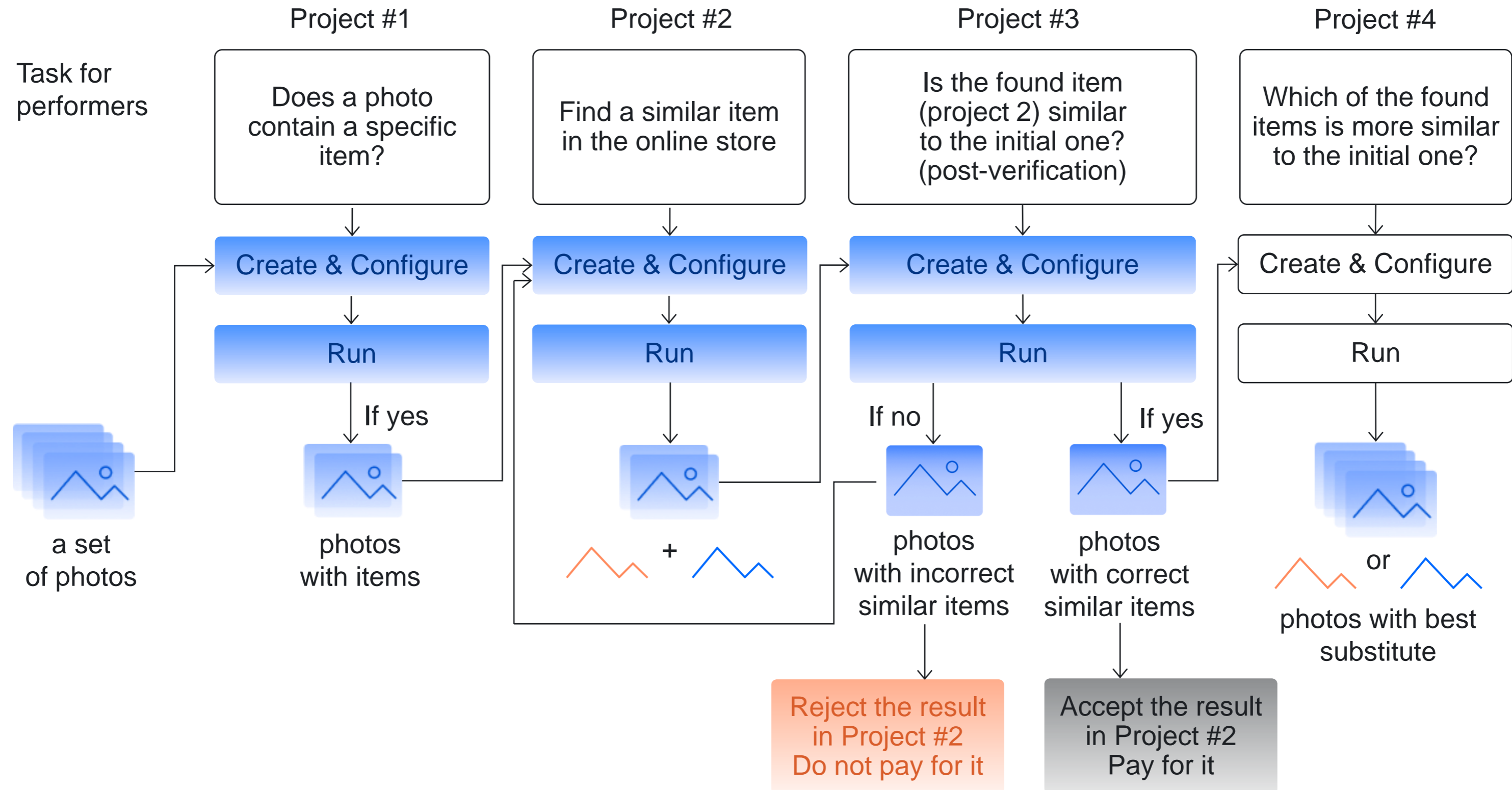
Most of us are at this step



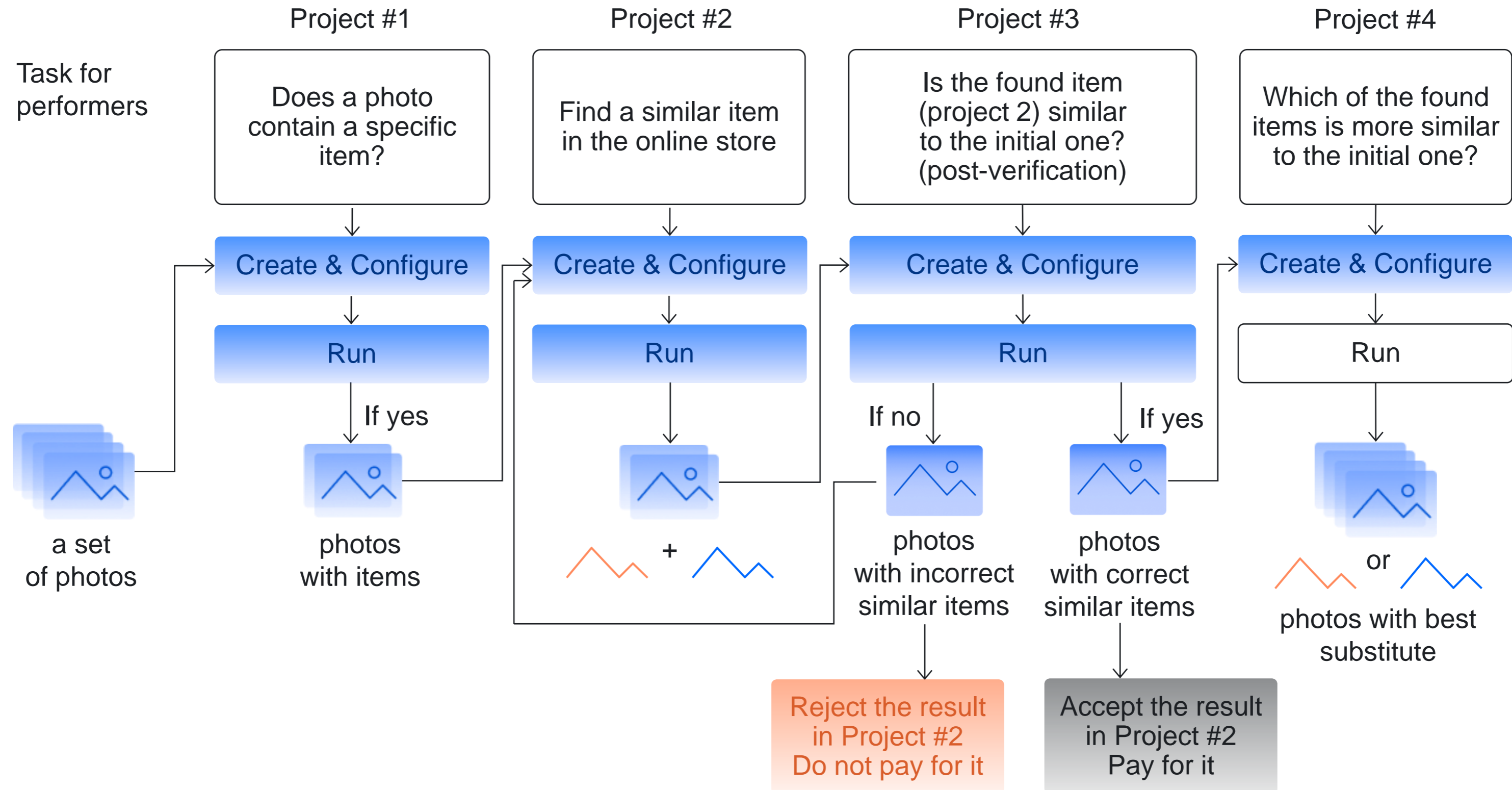
Most of us are at this step



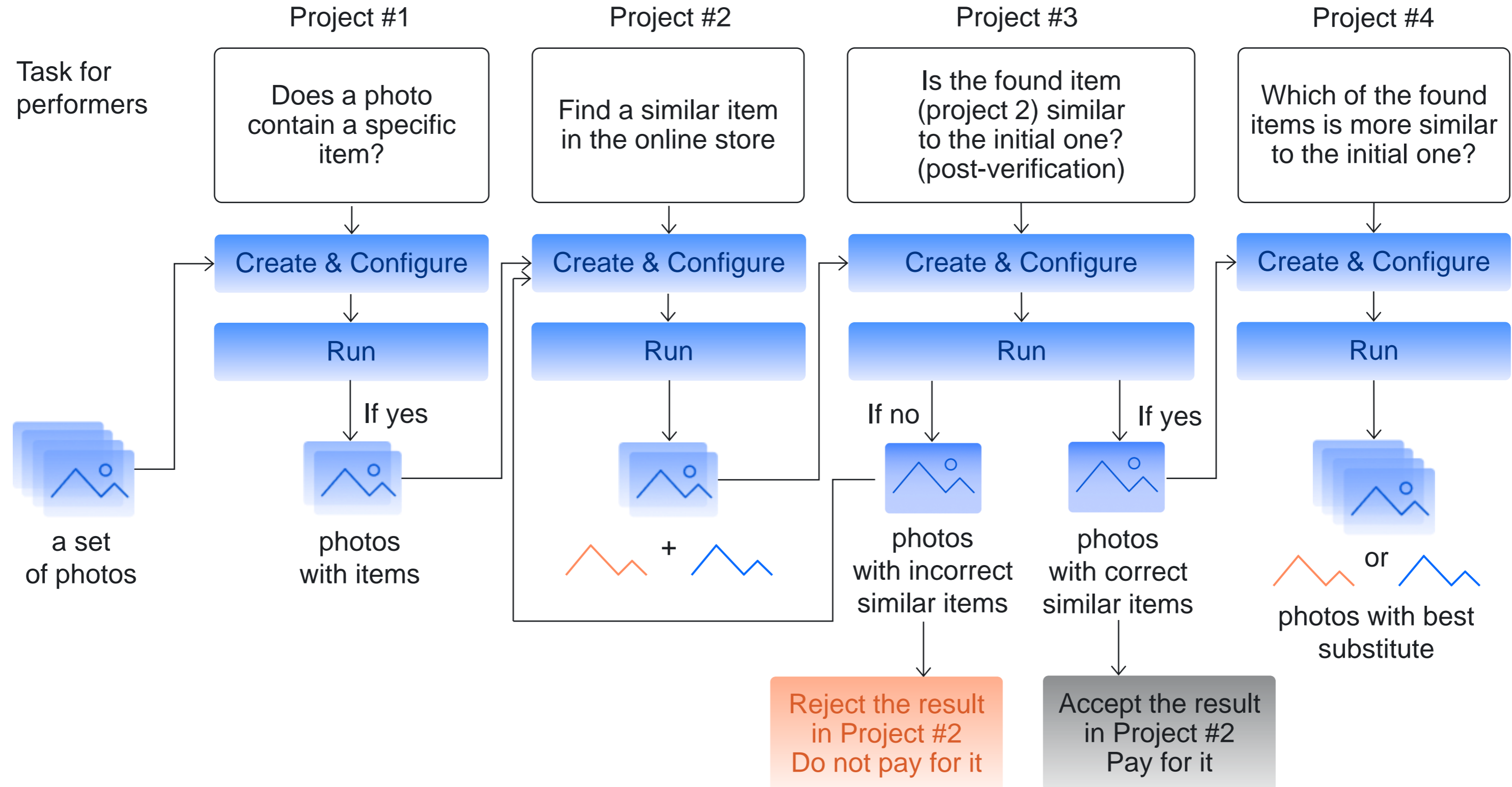
Most of us are at this step



Most of us are at this step



Most of us are at this step



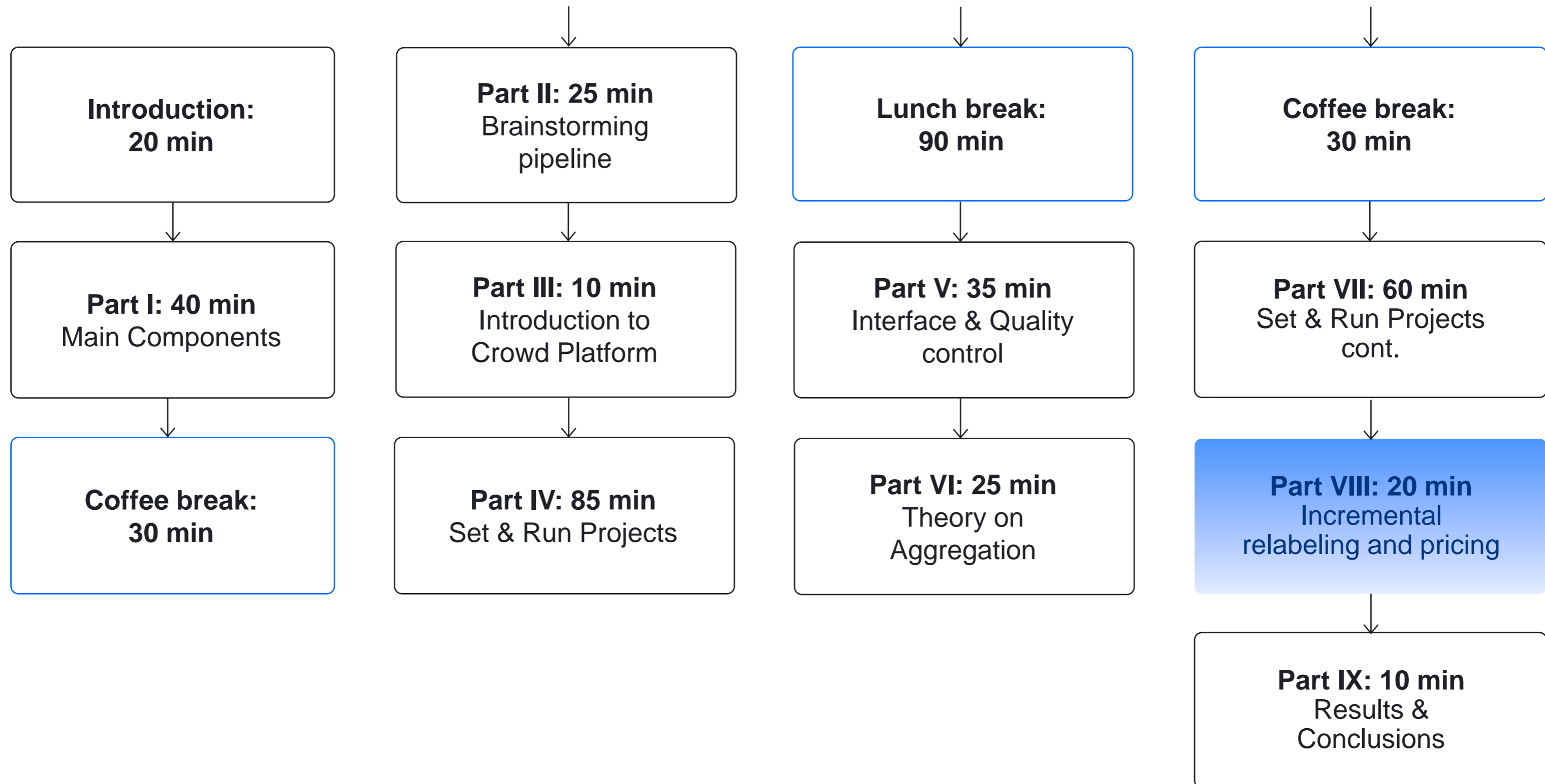
Part VIII

Theory on incremental relabelling and pricing

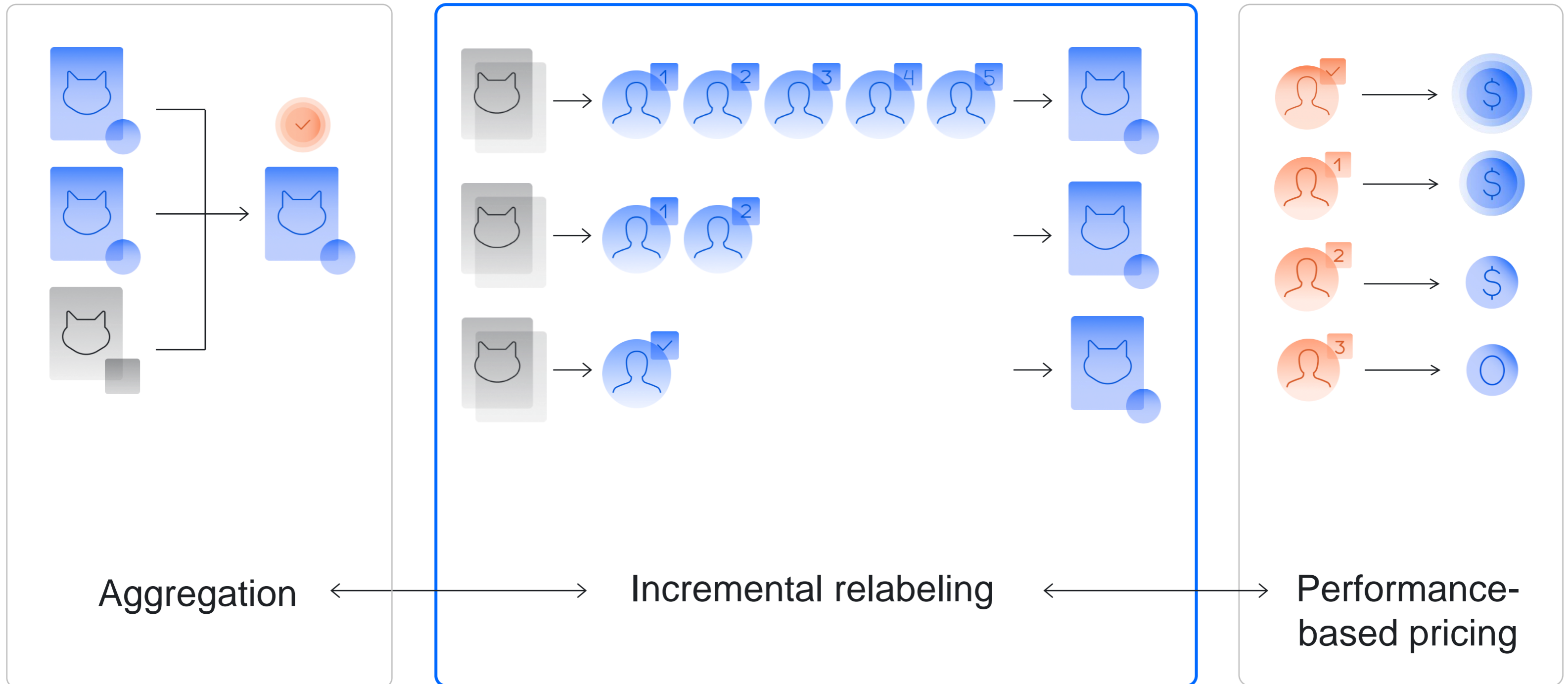
Valentina Fedorova,
Research analyst

Toloka

Tutorial schedule



Key components of labeling with crowds



Incremental relabeling
aka dynamic overlap

Pool settings: dynamic overlap

Quality control

Add rules to get more accurate responses.
All rules work independently.

NON-AUTOMATIC ACCEPTANCE No REVIEW PERIOD IN DAYS

CAPTCHA FREQUENCY

[+](#) Add Quality Control Rule

Overlap

Specify how many performers you want to complete each task in the pool.

OVERLAP

DYNAMIC OVERLAP Off

Speed/quality ratio

Specify additional conditions for selecting performers by their rating in Toloka. This will improve quality, but may reduce the speed of task completion because there will be fewer performers available for completing tasks. [Learn more](#)

Top % Online Time

Specify the percentage of top-rated active users who can access tasks in the pool.

Incremental relabeling problem

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



Incremental relabeling scheme (IRL)

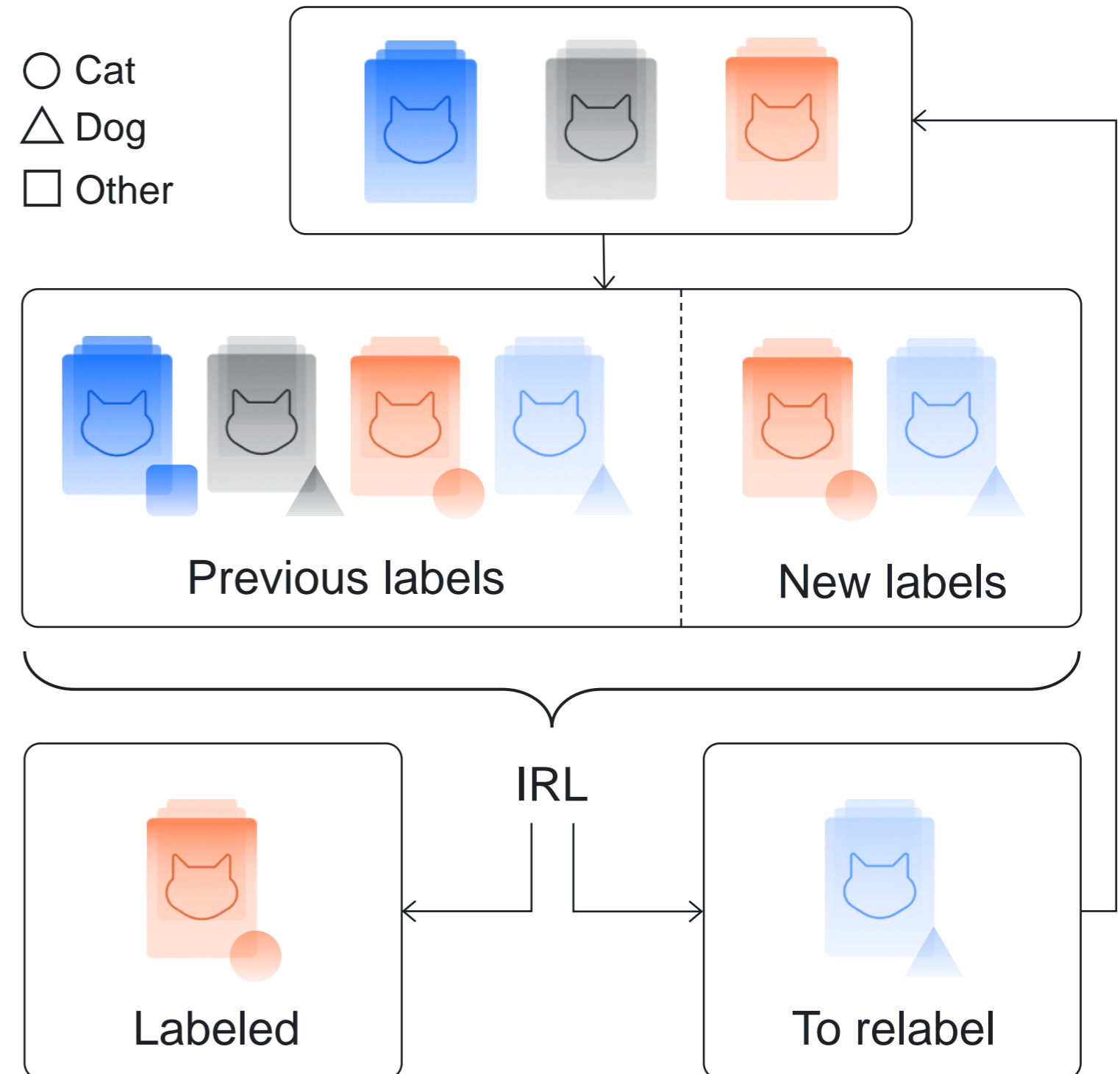
Request 1 label for each object

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

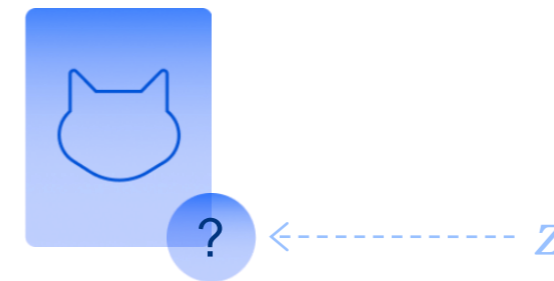


Notations

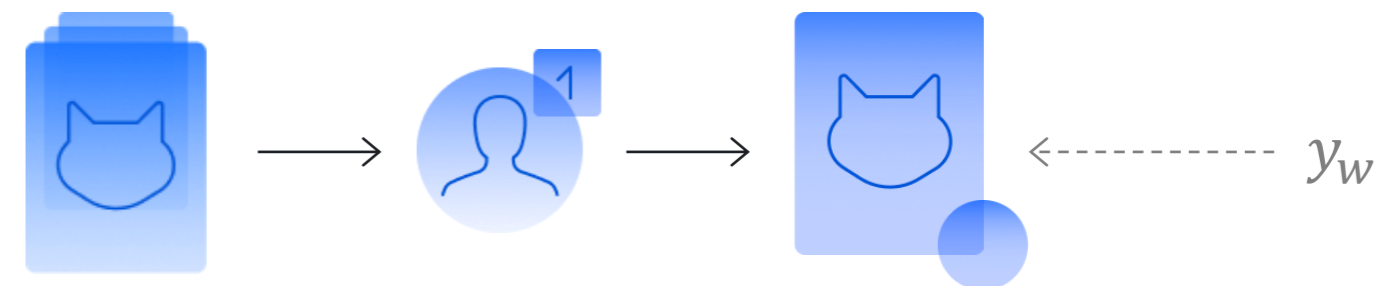
- ▶ Consider one object



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



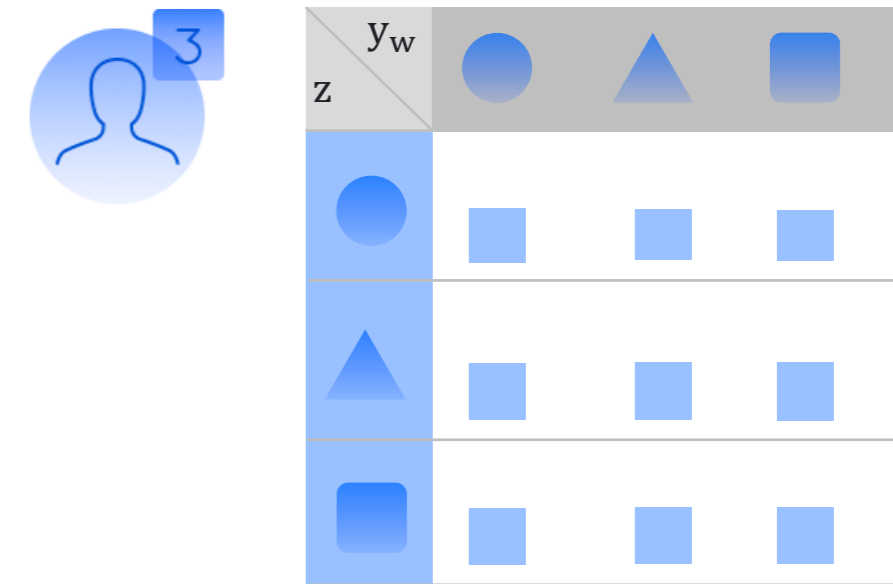
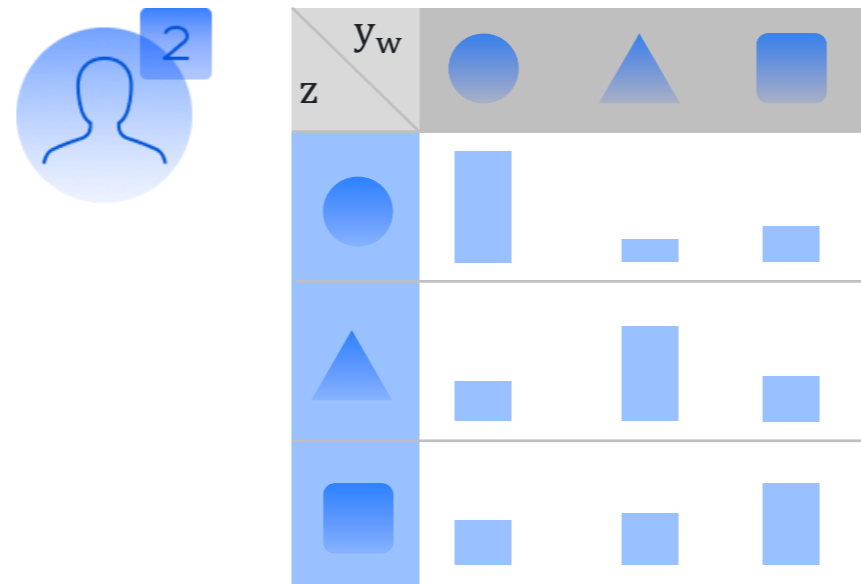
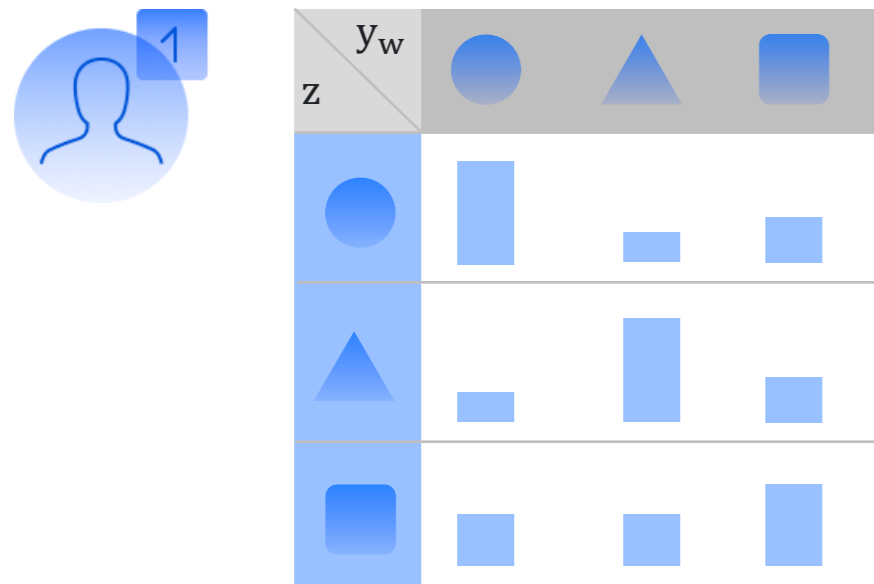
- ▶ $y_w \in \{1, \dots, K\}$ — observed noisy label from worker w :



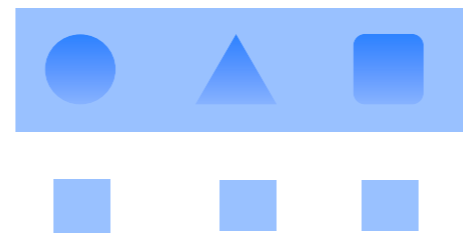
Notations

- ▶ Noisy label model for worker w :

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



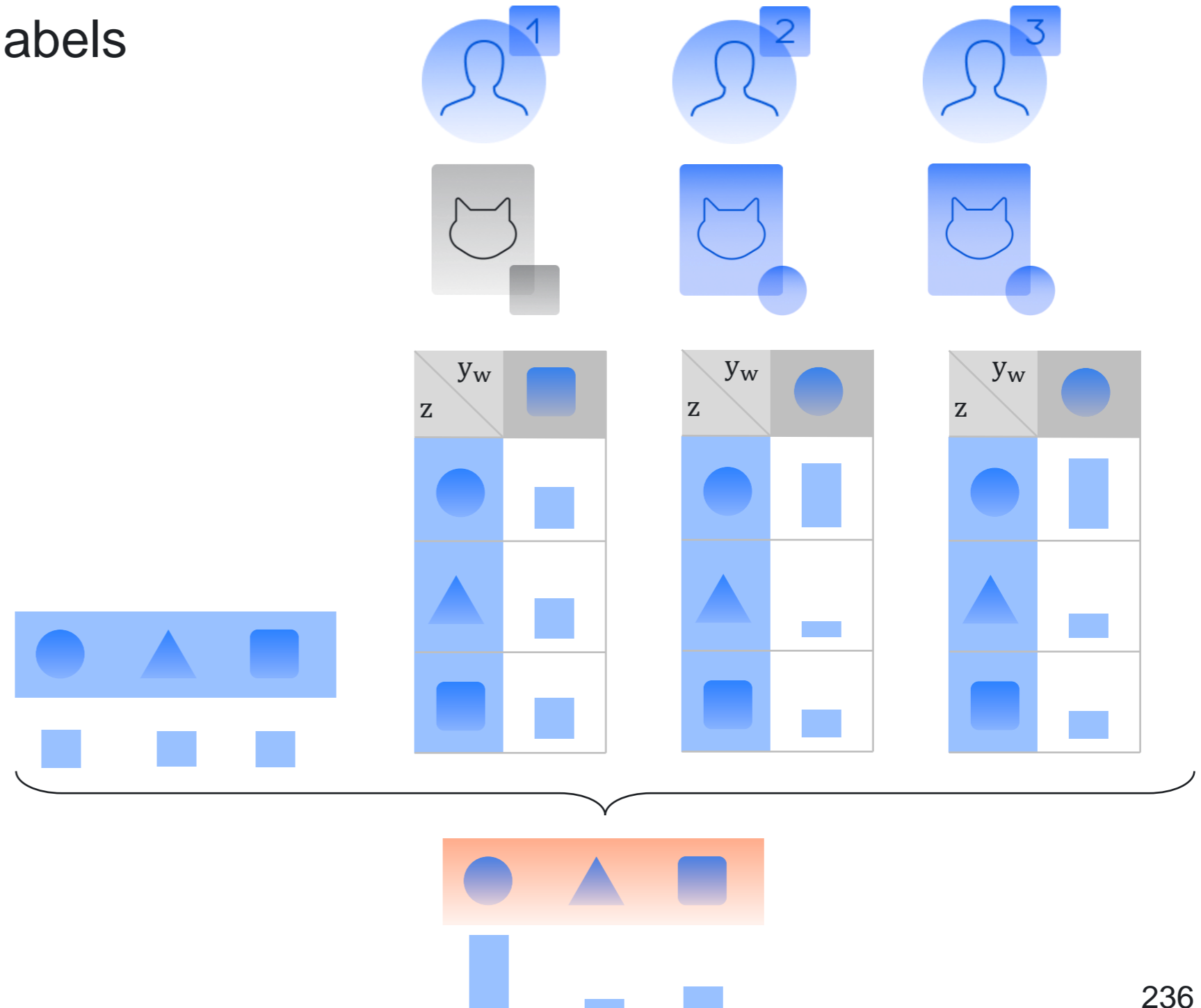
- ▶ Prior distribution: $\Pr(Z = k) = p_k$



Posterior distribution

- ▶ $\{y_{w_1}, \dots, y_{w_n}\}$ — accumulated noisy labels for the object
- ▶ Using Bayes rule:

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



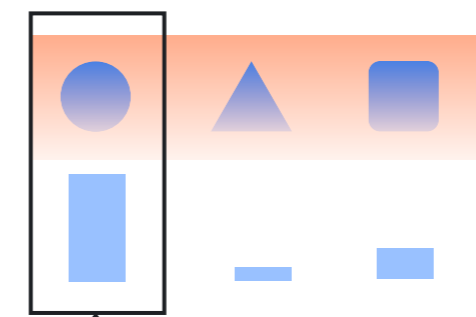
Expected accuracy of aggregated labels

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

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

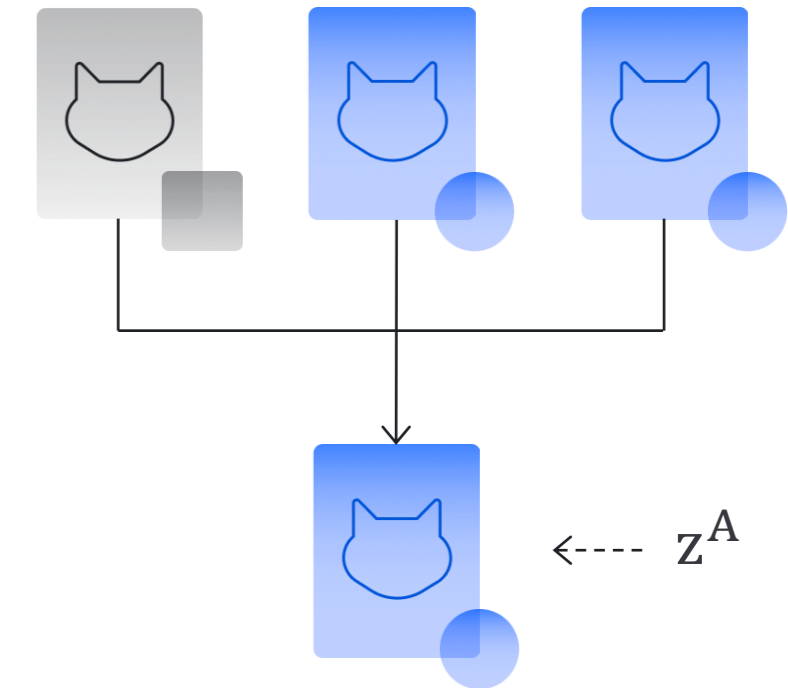
- ▶ Stop labeling if $E(\delta(z = z^A) | \{y_{w_1}, \dots, y_{w_n}\}) \geq C$

Parameter



Expected accuracy of z^A

Posterior



Incremental relabeling algorithm

Input: $U_{t=1}^T$ 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

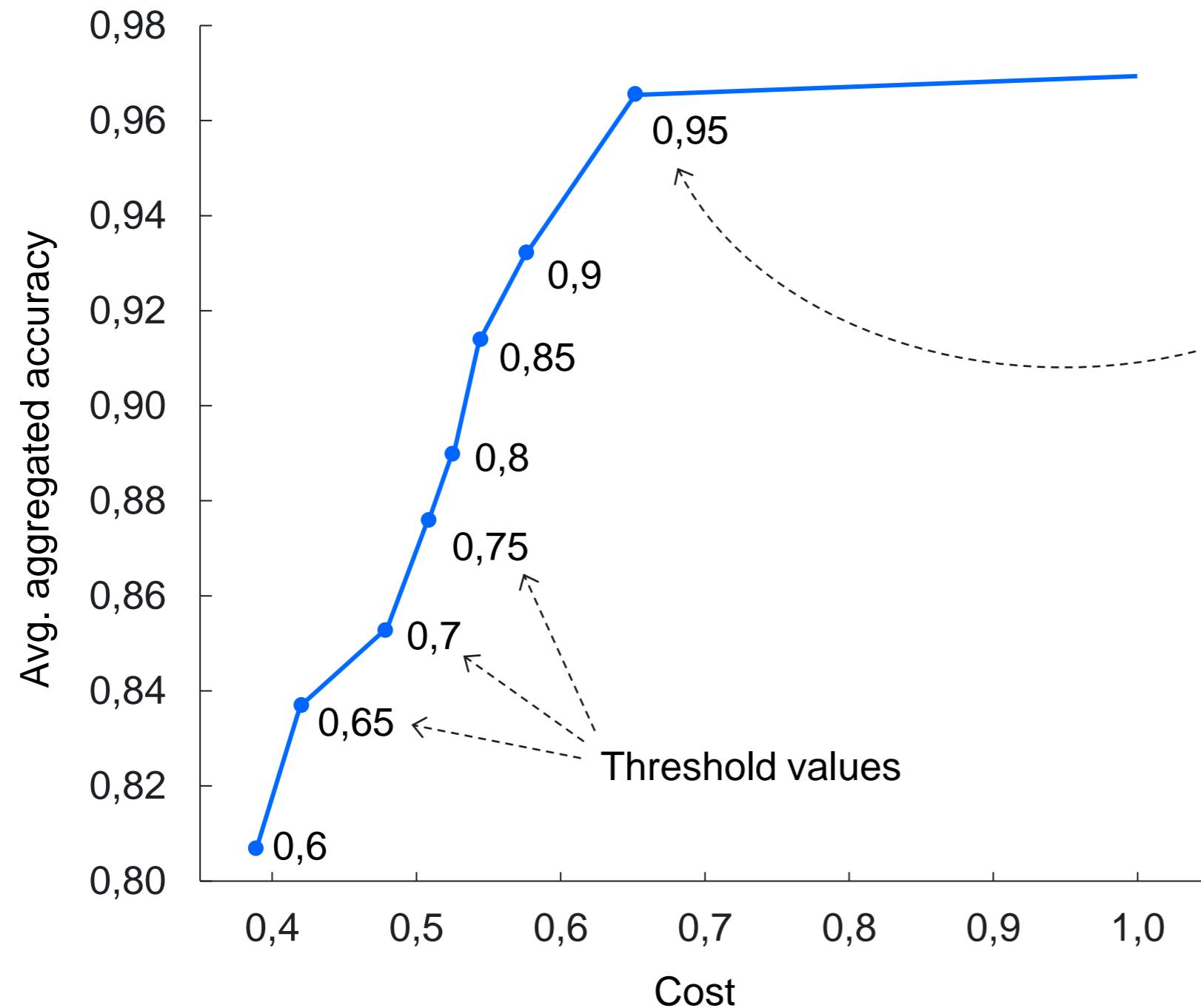
$z_j^M = M(U_{t=1}^T Y^t)$ ← Current aggregated label

$c_j = E(z_j = z_j^M | U_{t=1}^T Y^t)$ ← 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 $\approx 35\%$ of noisy labels

How to obtain a cost-accuracy plot

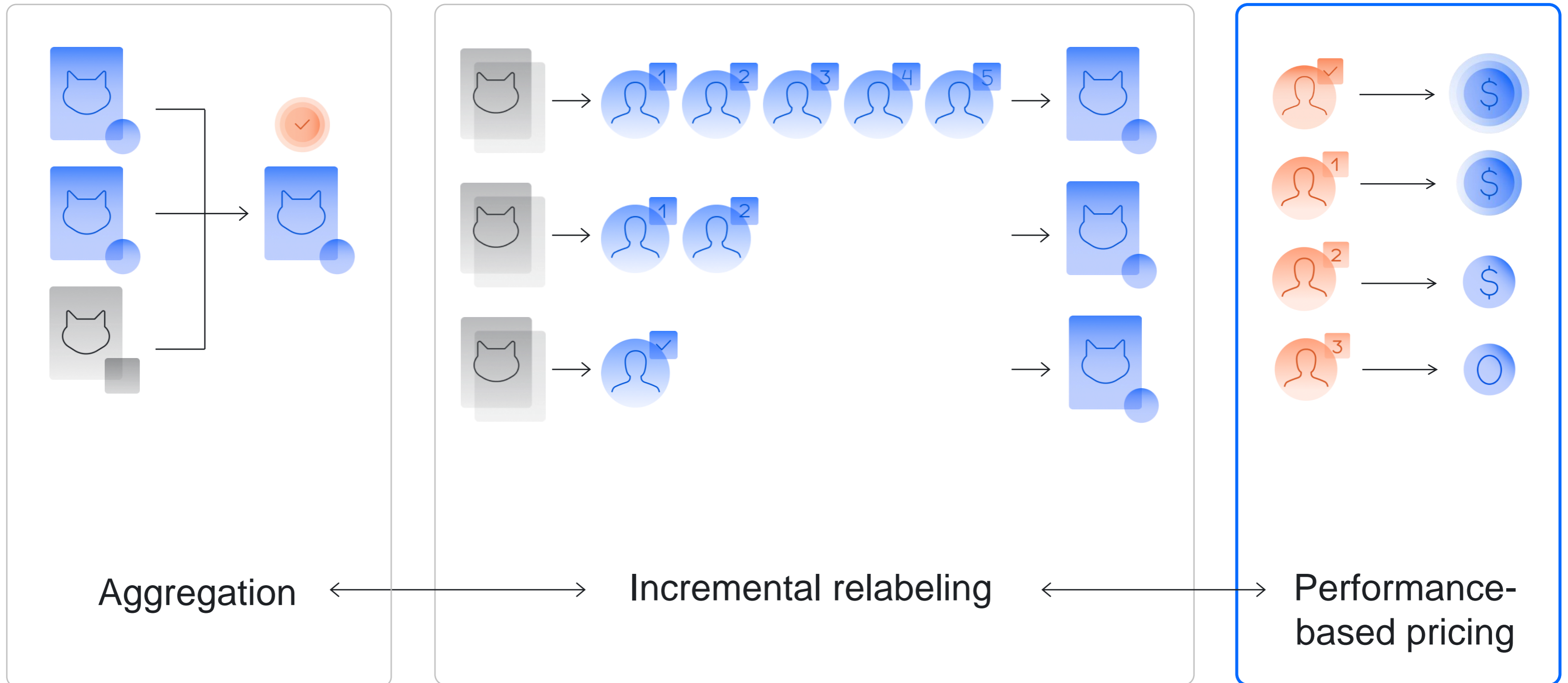
Data for the plot:

- ▶ 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 $0 < c_0 < \dots < c_m \leq 1$
- ▶ 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

POOL NAME (VISIBLE ONLY TO YOU) ?

Use project description

PUBLIC DESCRIPTION ?

Add a private description

Price per task suite

You can add one or more tasks to the page. Enter the total price for all tasks on the page.

PRICE IN US DOLLARS ? FEE ?

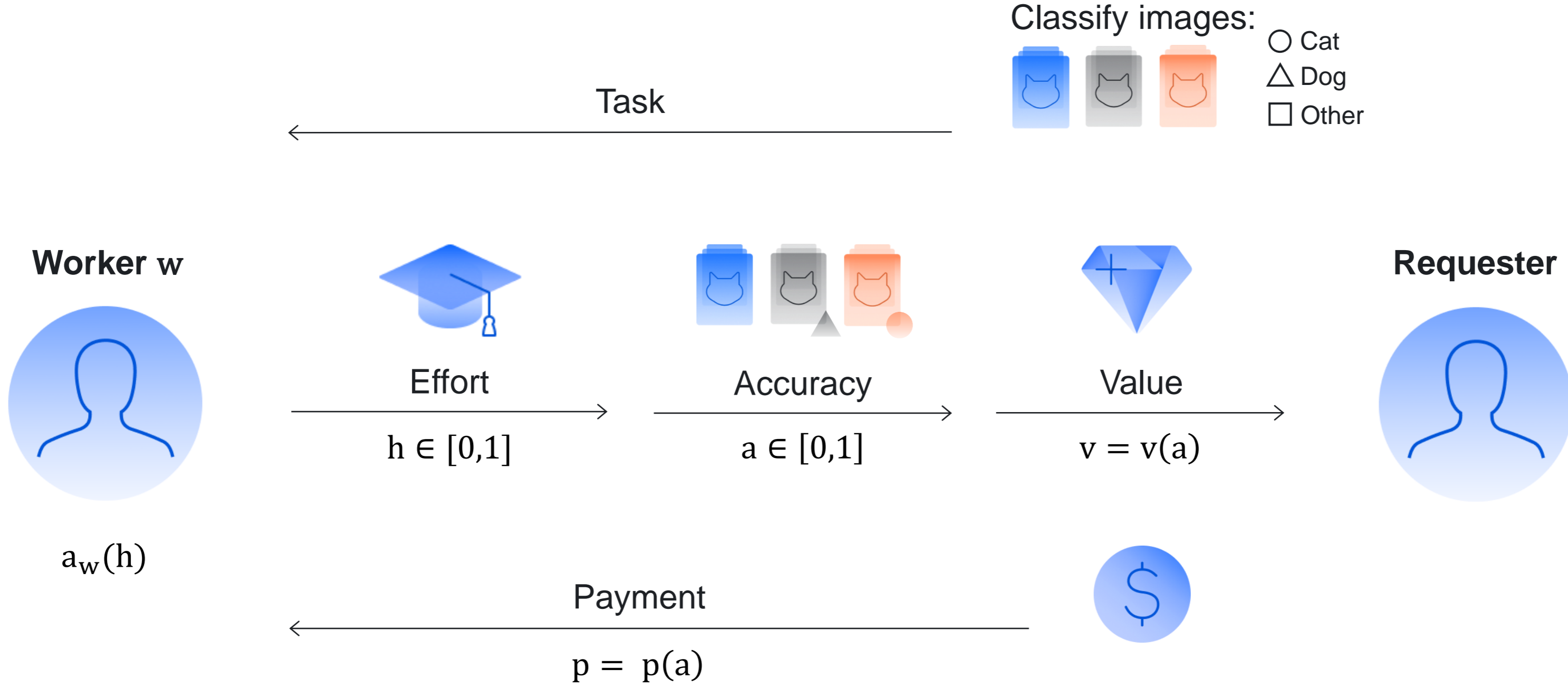
Performers

[Copy settings from...](#)

Filter performers who can access the task.
Toloka has users from different countries, so don't forget to filter by language and region. [Learn more](#)

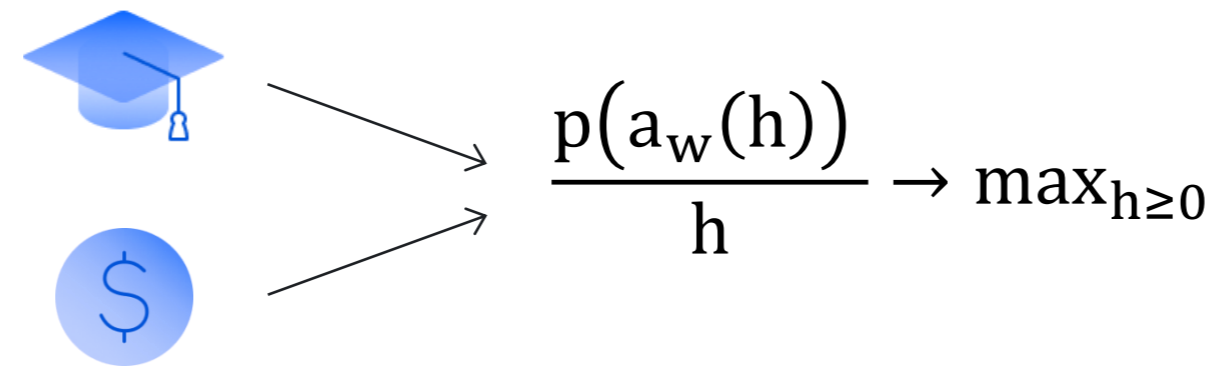
ADULT CONTENT ? Yes

Labeling as a game: notation

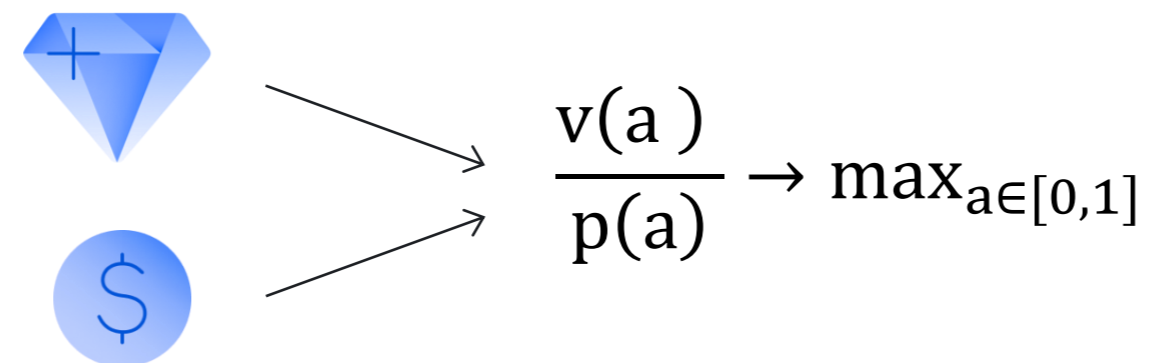


Labeling as a game: formalization

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




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



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.:



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



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



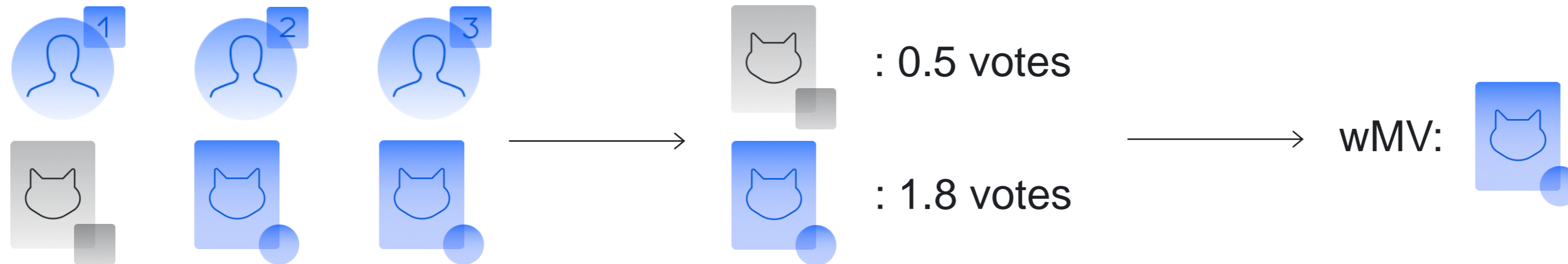
16 correct GS
among 20
 $\hat{q}_w = 0.8$



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

Performance-based pricing in practice: settings

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



- ▶ 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 :

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

Expected accuracy for MV

The price is p/n

$$a_0 = 0.99$$



$$\hat{q}_w = 1$$



$$0.3\$$$



$$\hat{q}_w = 0.8 \\ \Rightarrow n = 15$$



$$0.02\$$$

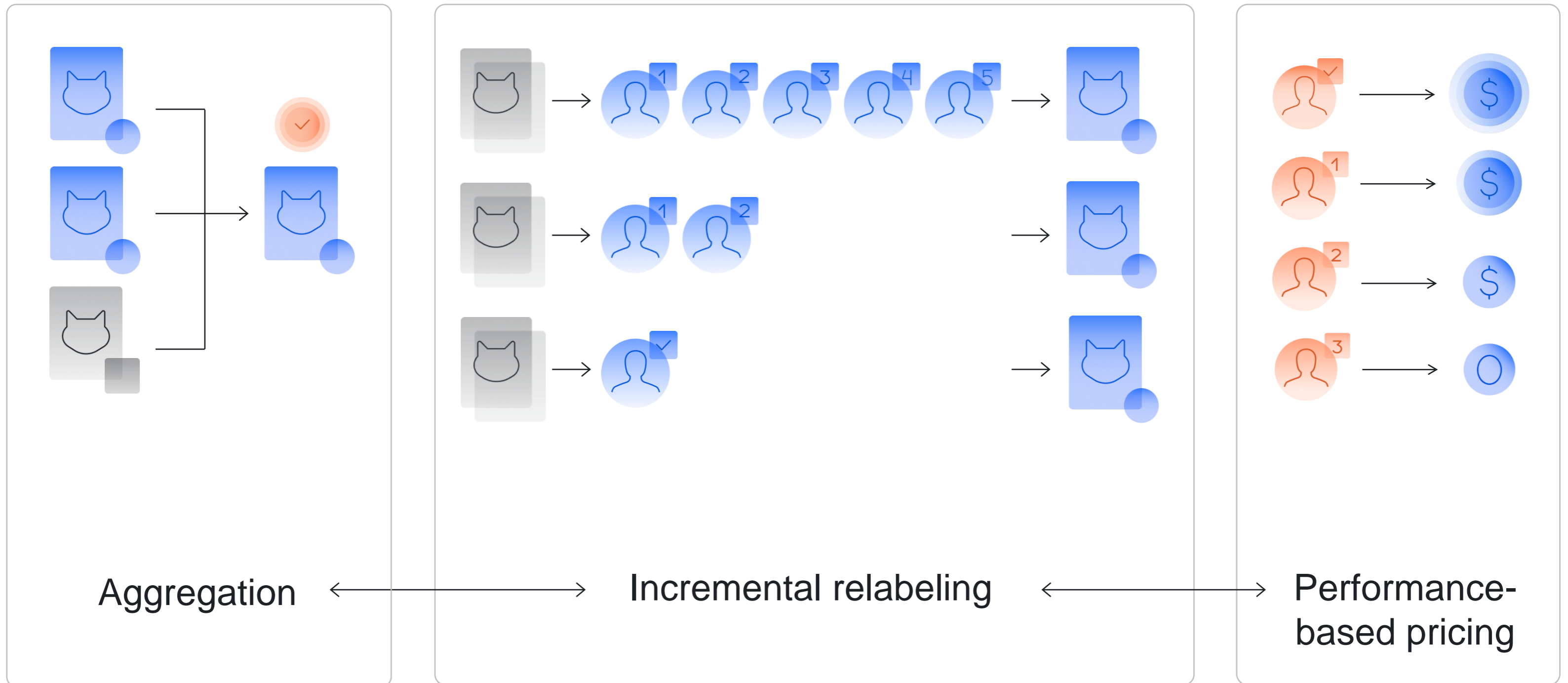


$$\hat{q}_w = 0.5 \\ \Rightarrow n = \infty$$



$$0\$$$

Key components of labeling with crowds

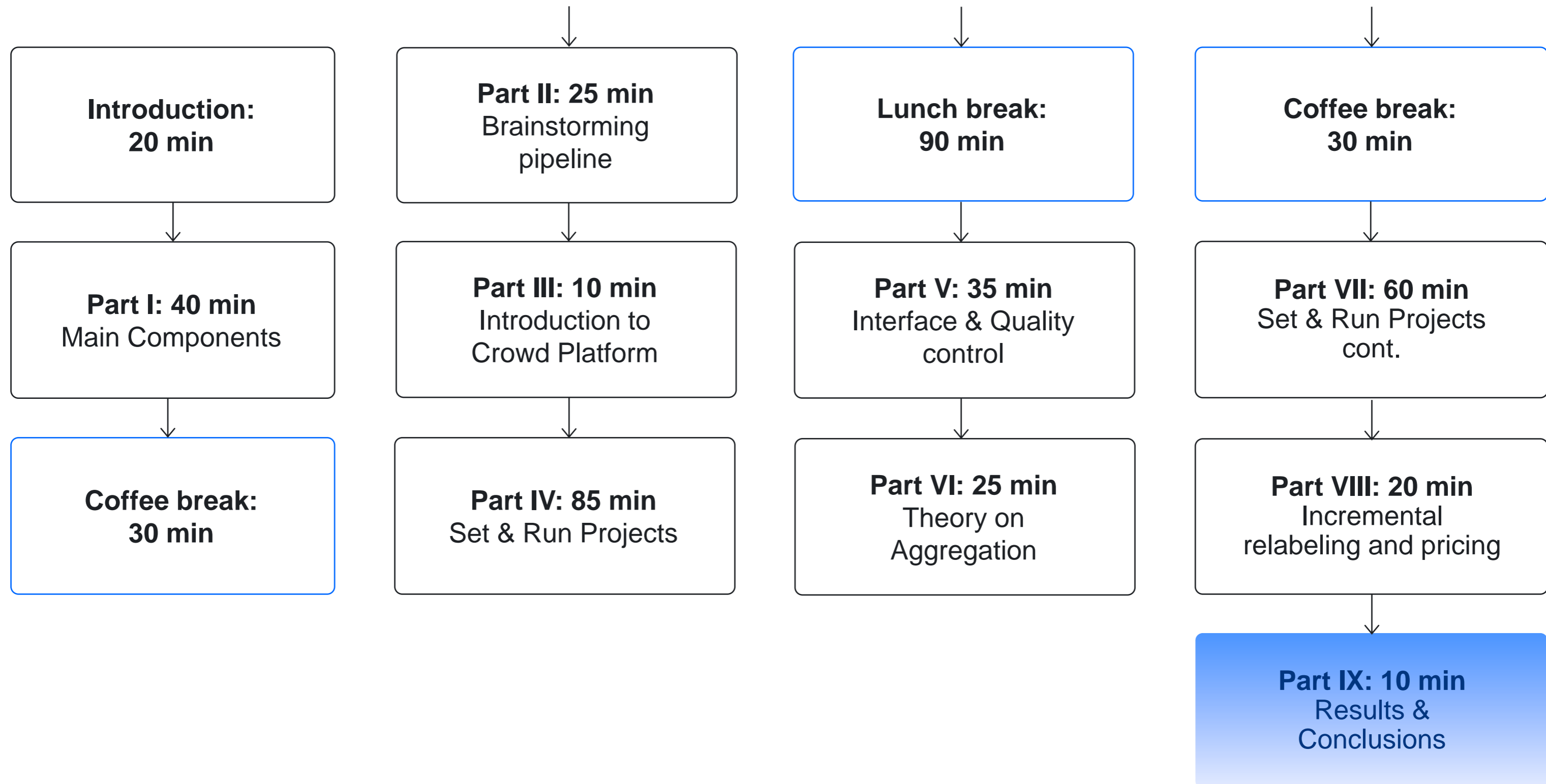


Part IX

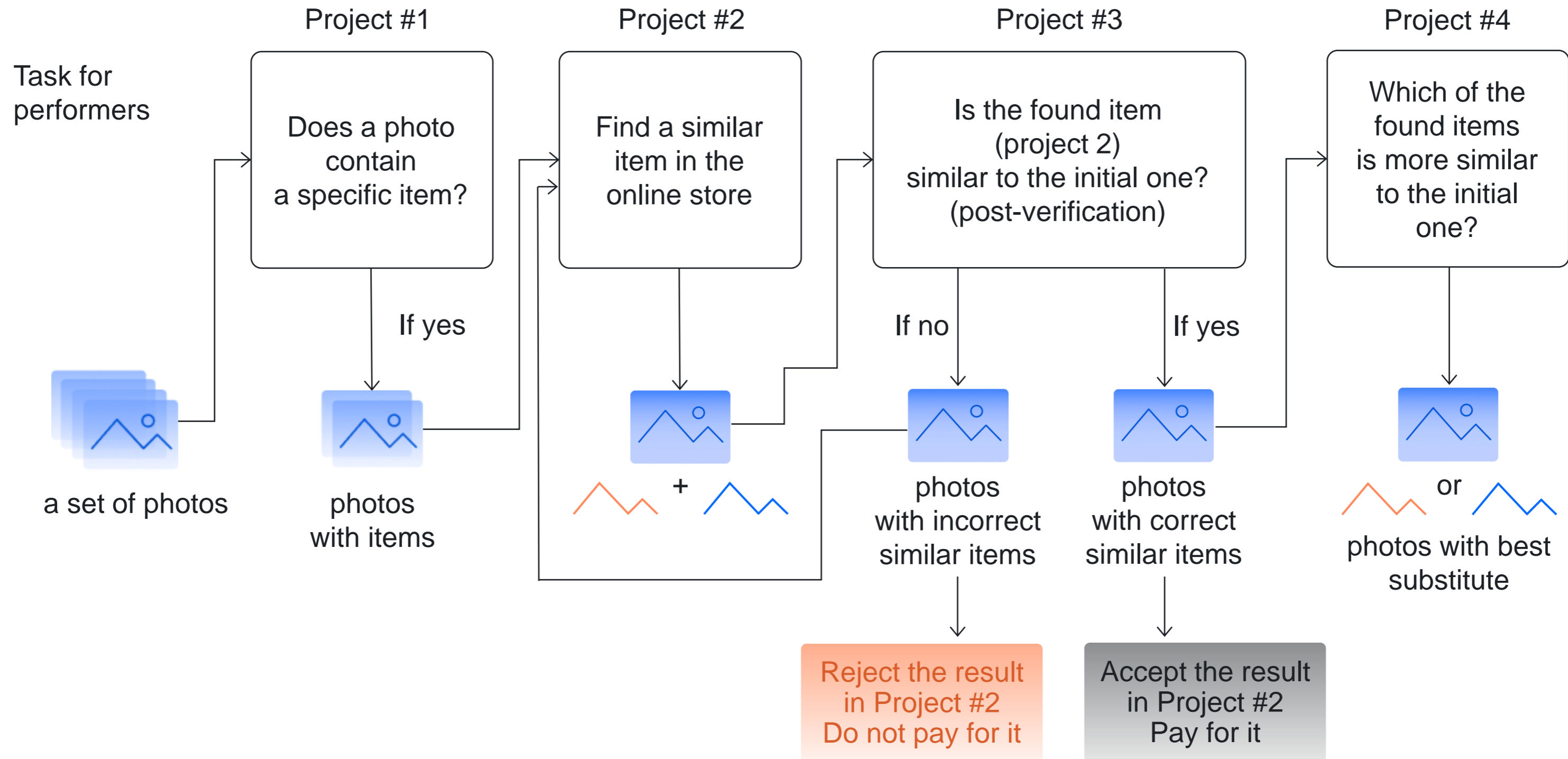
Discussion of the projects' results. Conclusions

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?

Our results

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



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

Our results

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



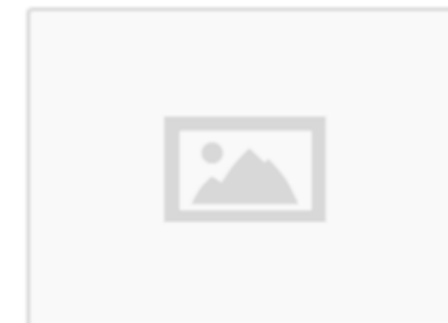
Find the same **shoes** on ASOS

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



Check that the uploaded image matches the product in the store.

[Check the item](#)

Are these **shoes** similar to each other?

Shoes must be the same color and the same style.

Yes No

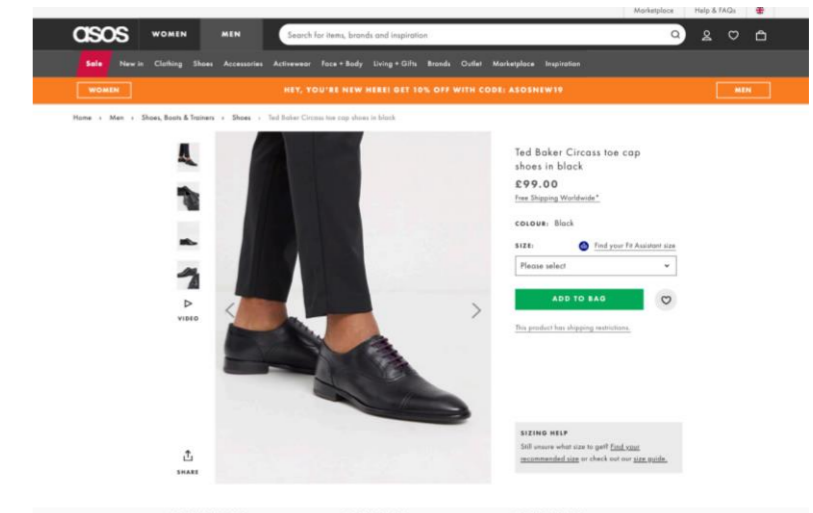
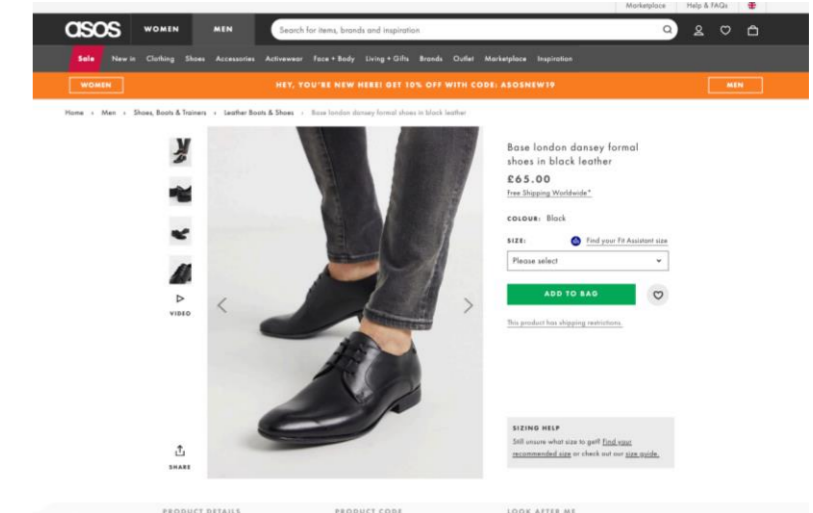
Project #4: Decide which substitute works best

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