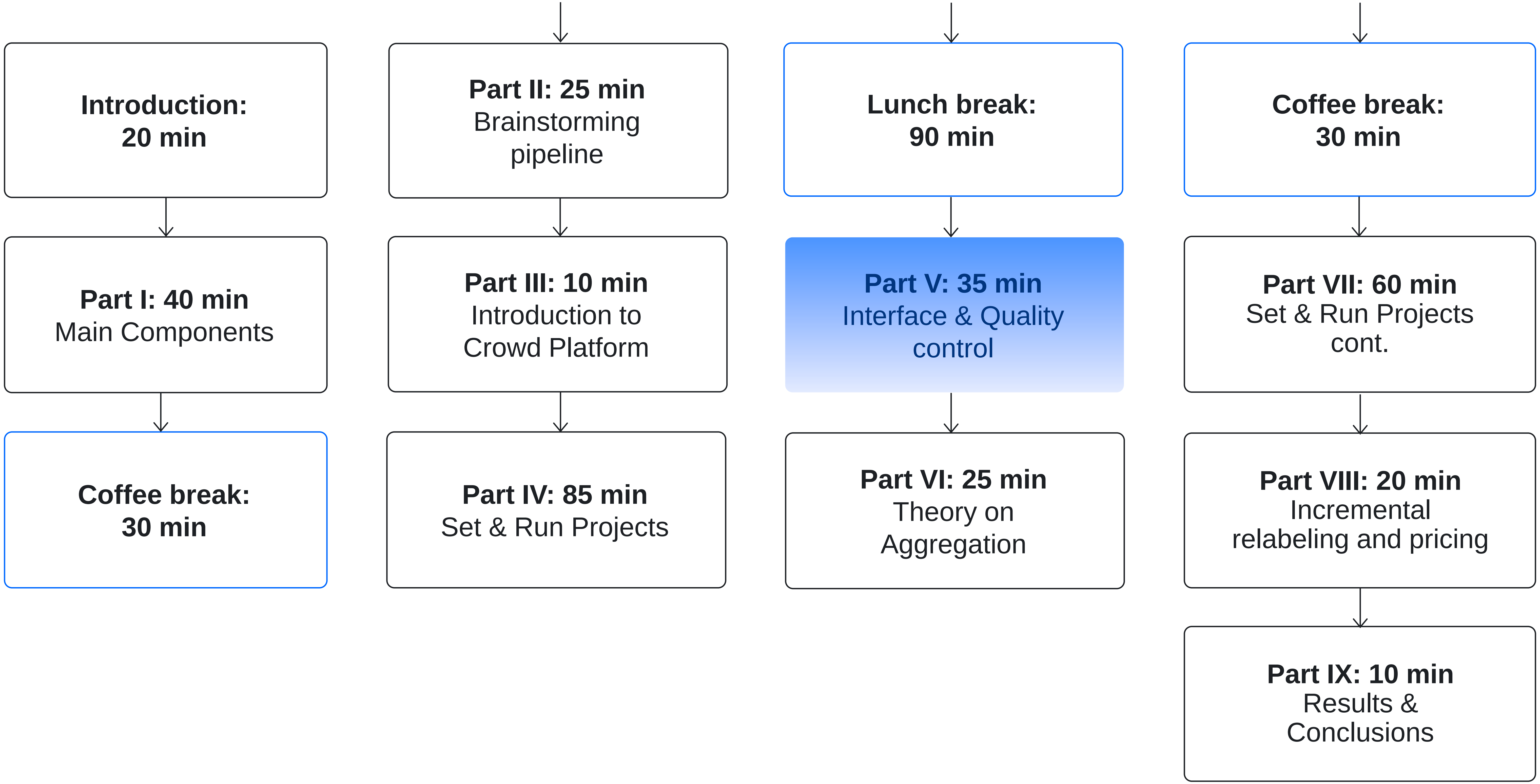
Part V

Effective quality control and task interface: details

Alexey Drutsa,
Head of Efficiency and Growth Division, Toloka

Tutorial schedule



Introduction:
20 min

Part I: 40 min
Main Components

Coffee break:
30 min

Part II: 25 min
Brainstorming pipeline

Part III: 10 min
Introduction to Crowd Platform

Part IV: 85 min
Set & Run Projects

Lunch break:
90 min

Part V: 35 min
Interface & Quality control

Part VI: 25 min
Theory on Aggregation

Coffee break:
30 min

Part VII: 60 min
Set & Run Projects cont.

Part VIII: 20 min
Incremental relabeling and pricing

Part IX: 10 min
Results & Conclusions

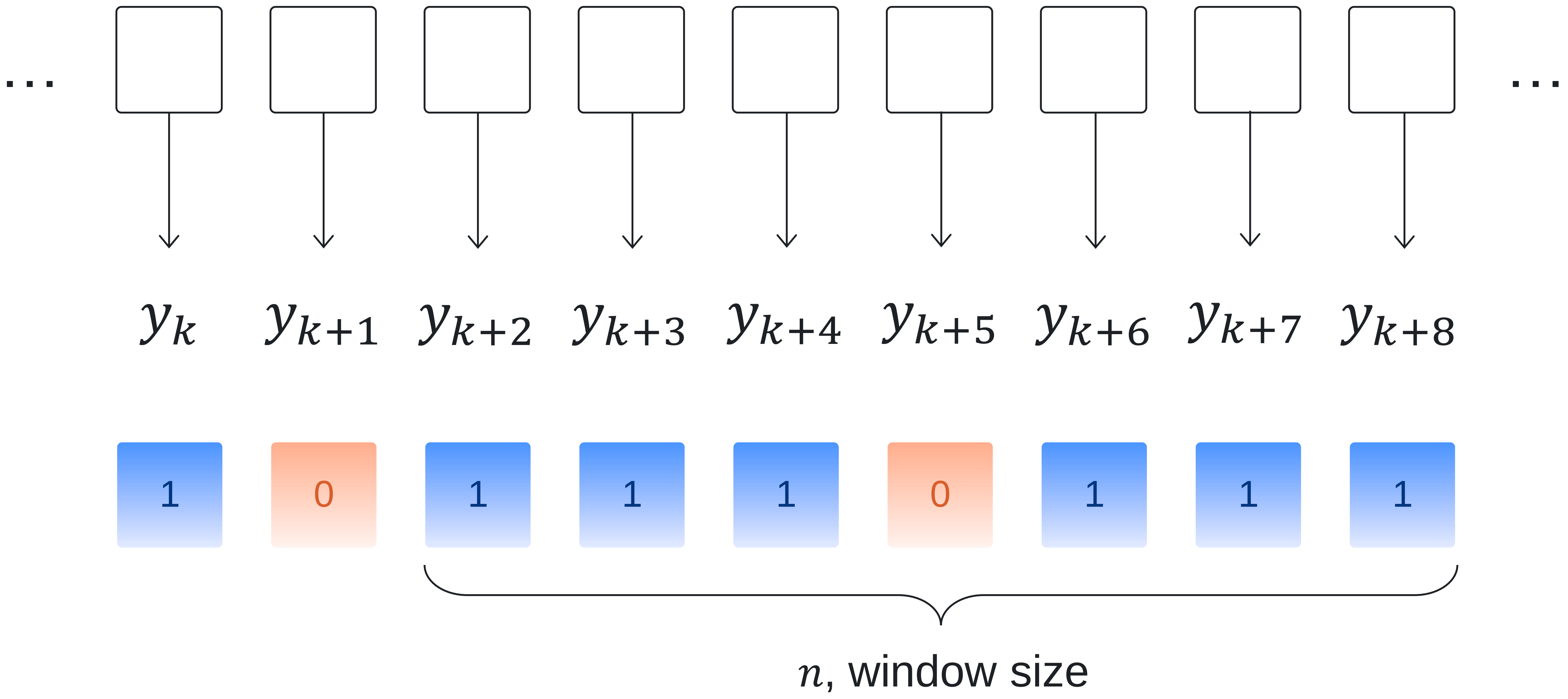
Quality control: the rate of correct answers

Task sequence

Tasks executed by a performer

Signals of answer correctness

For instance, binary, y, ∈ {0,1}



… …

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}

1 0 1 1 1 0 1 1 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 $\boldsymbol{y_i \in [0, 1]}$ 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

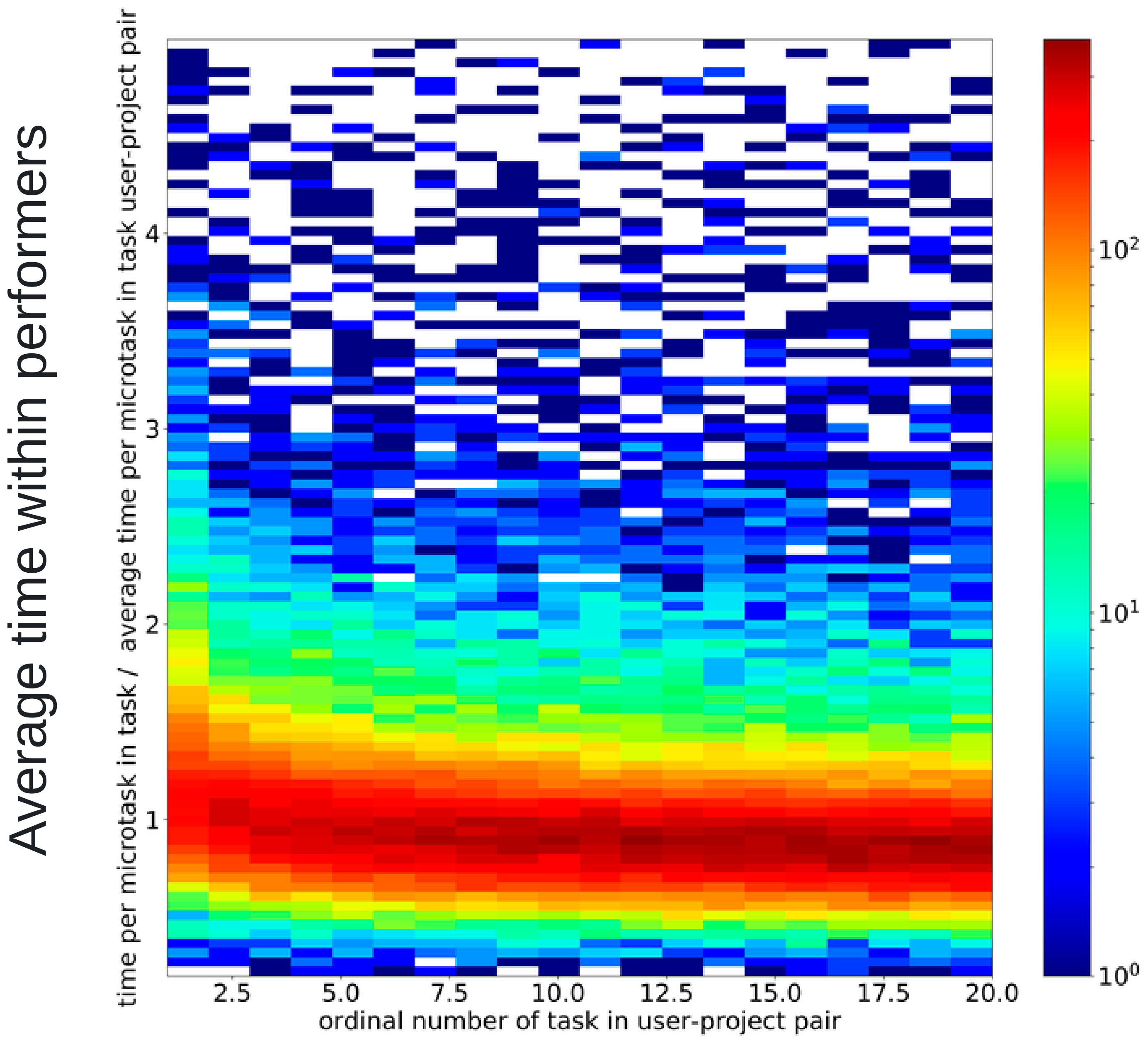
- Access control to certain projects and tasks
- e.g., revoke access to your tasks if a skill becomes too low

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

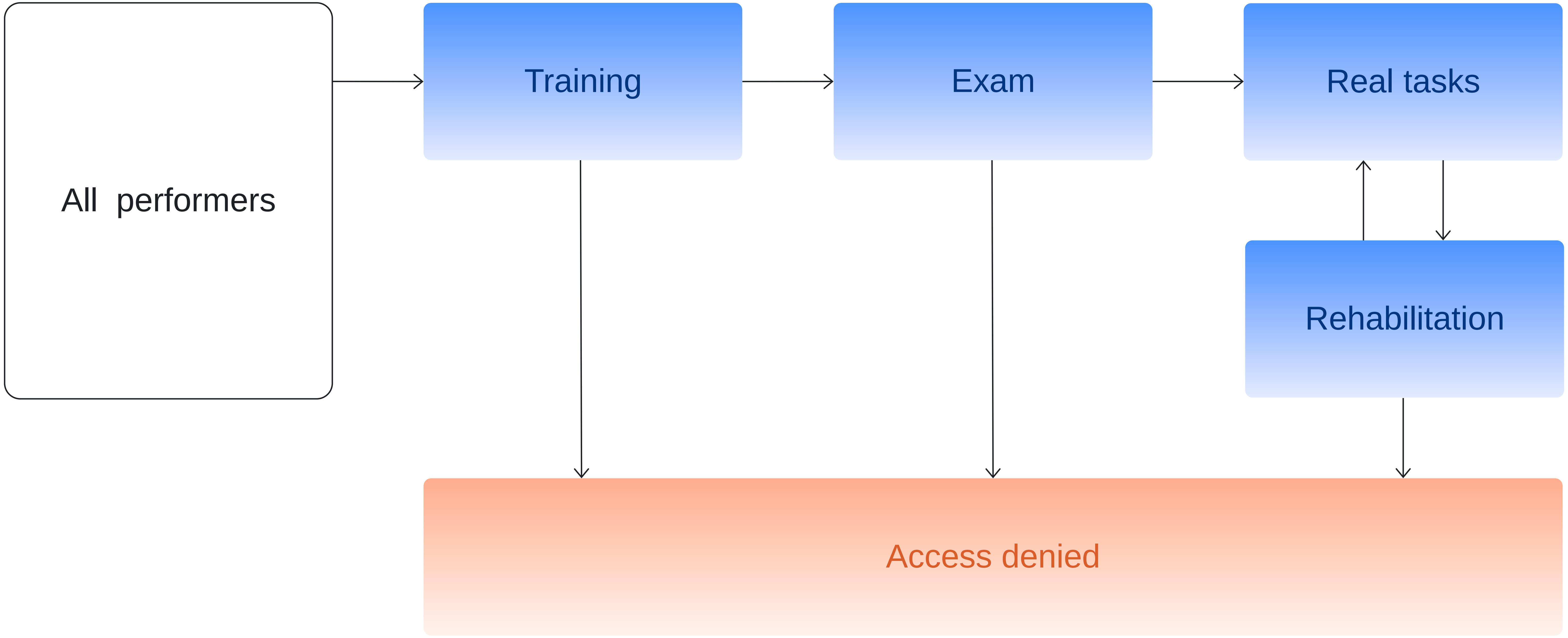
- All tasks are control ones
- There are hints that explain incorrect answers

Exam task

Control the results of training

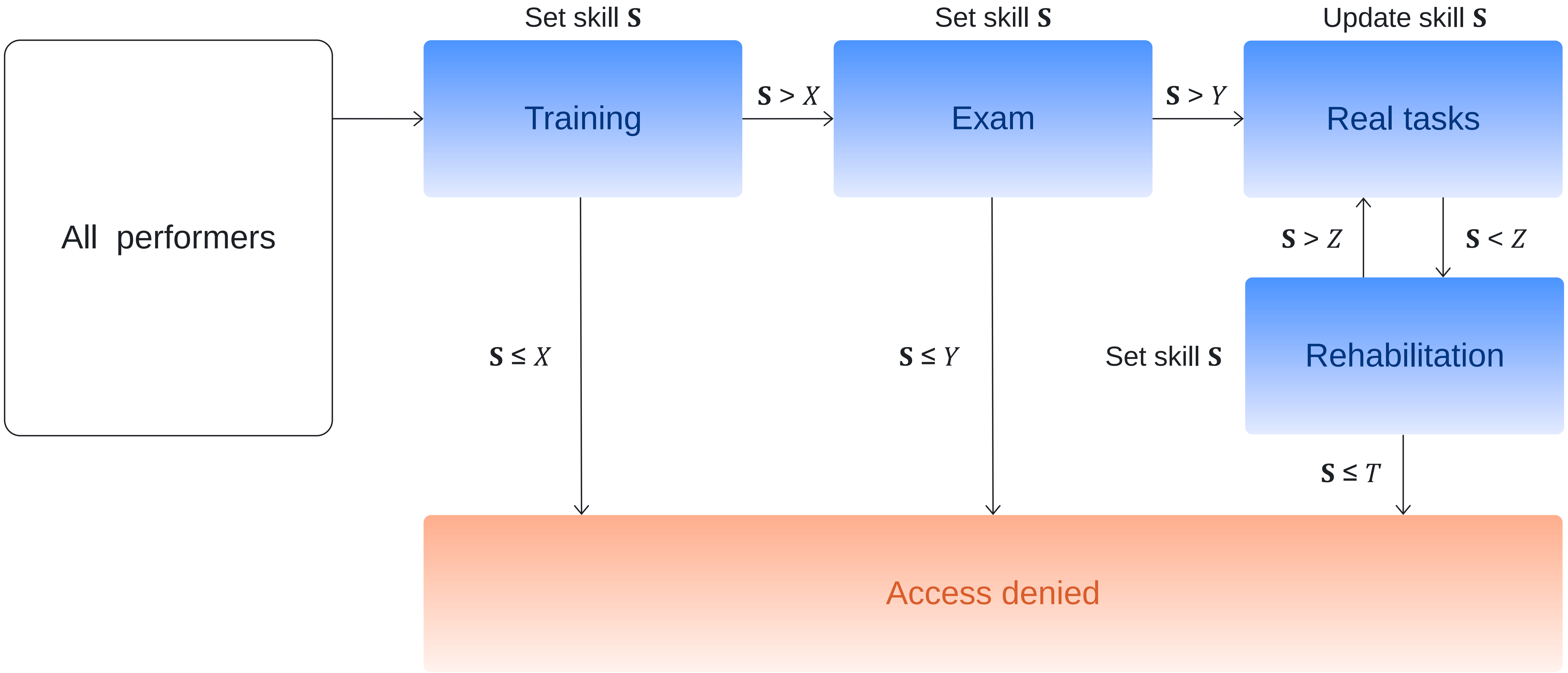
- All tasks are control ones
- No hints and explanations
- A good exam should be:
  - Passable
  - Regularly updated
  - Small

Recommended life cycle of performers



Recommended life cycle of performers

Let quality be controlled by means of a skill $\boldsymbol{S}$



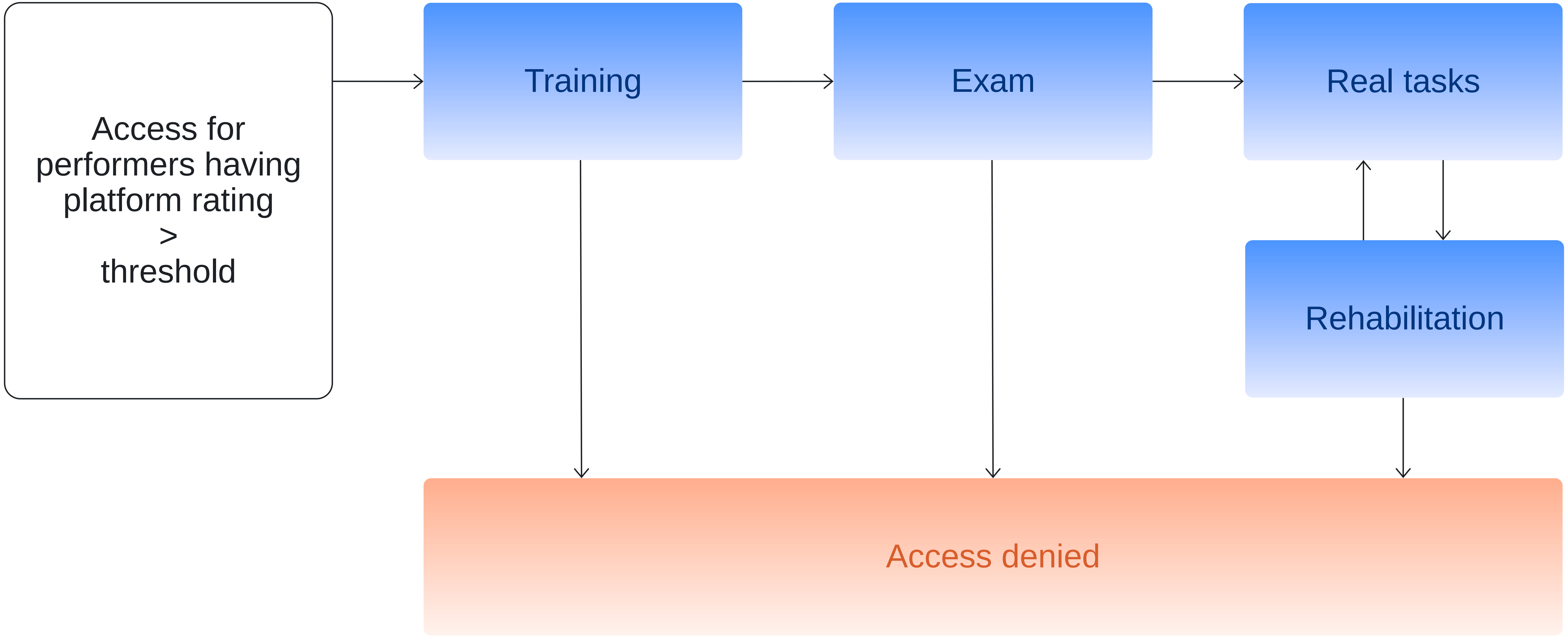
Rehabilitation task

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

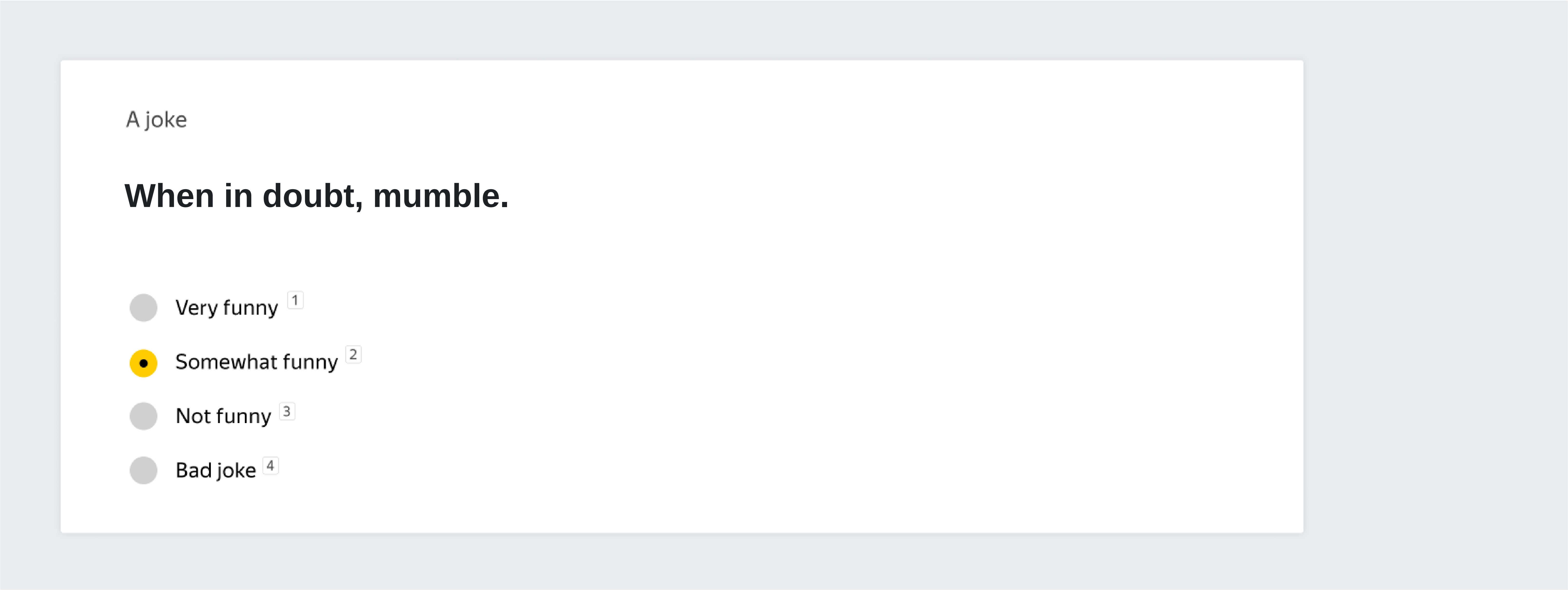
Interface.
Introduction

Task in the eyes of the performers

Web-page with specific features

- Long run time
- Repetitive actions
- Concentration
- Speed

Structure of a task interface



A joke

When in doubt, mumble.

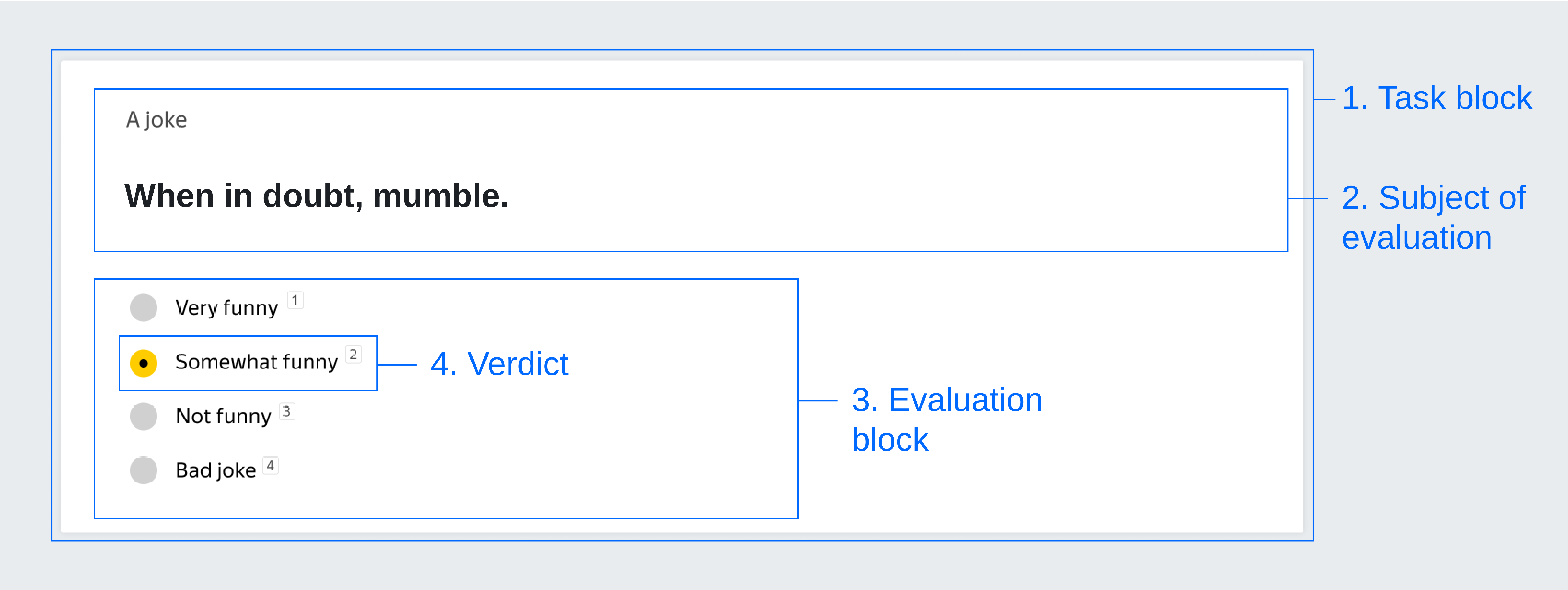
○ Very funny 1

● Somewhat funny 2

○ Not funny 3

○ Bad joke 4

Structure of a task interface



A joke

When in doubt, mumble.

○ Very funny 1

● Somewhat funny 2

○ Not funny 3

○ Bad joke 4

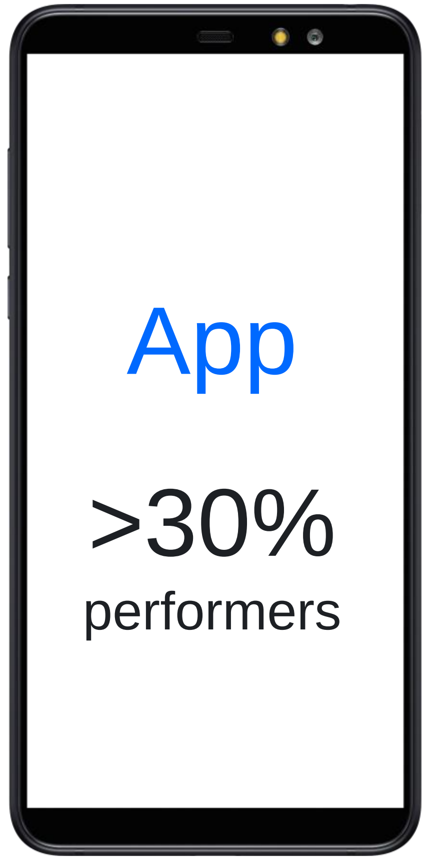
9 golden rules of interface structure

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

Web version



App

>30%

performers

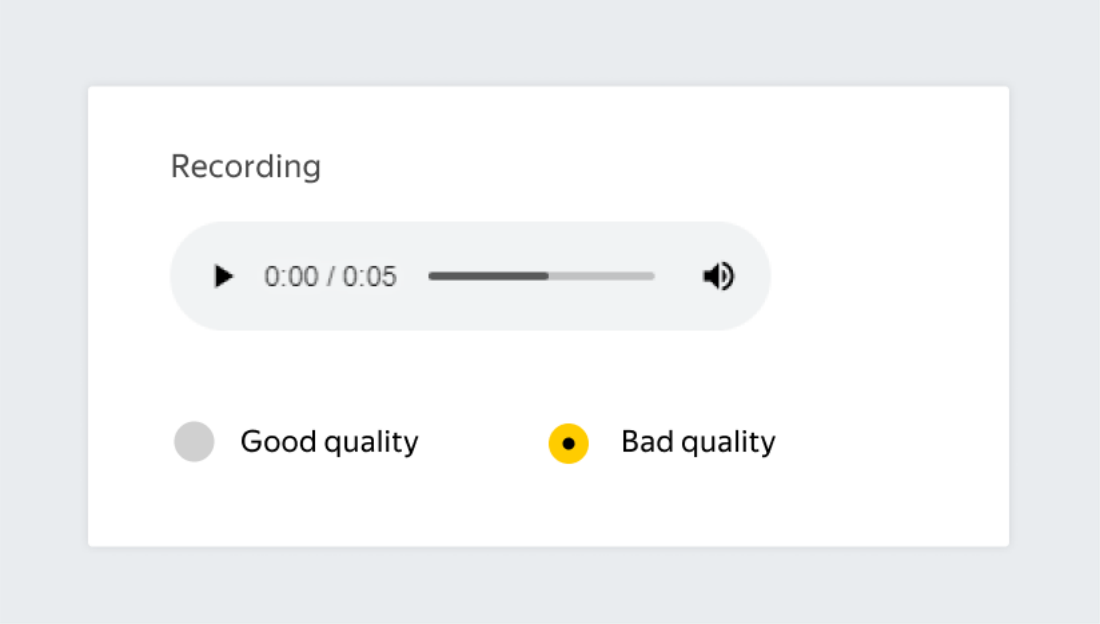
Possible limitations for mobile services:

- Task difficulty
- Media Content, Devices, and Browsers

Rule #1. Cross-platform compatibility

Task: evaluate sound quality in wav audio files

Web version

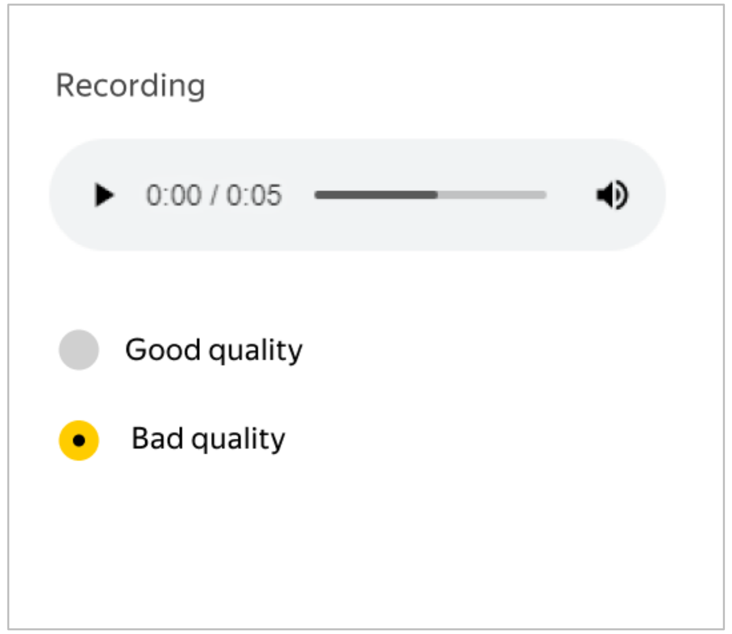


Recording

▶ 0:00 / 0:05 🔊

○ Good quality ● Bad quality

Android App



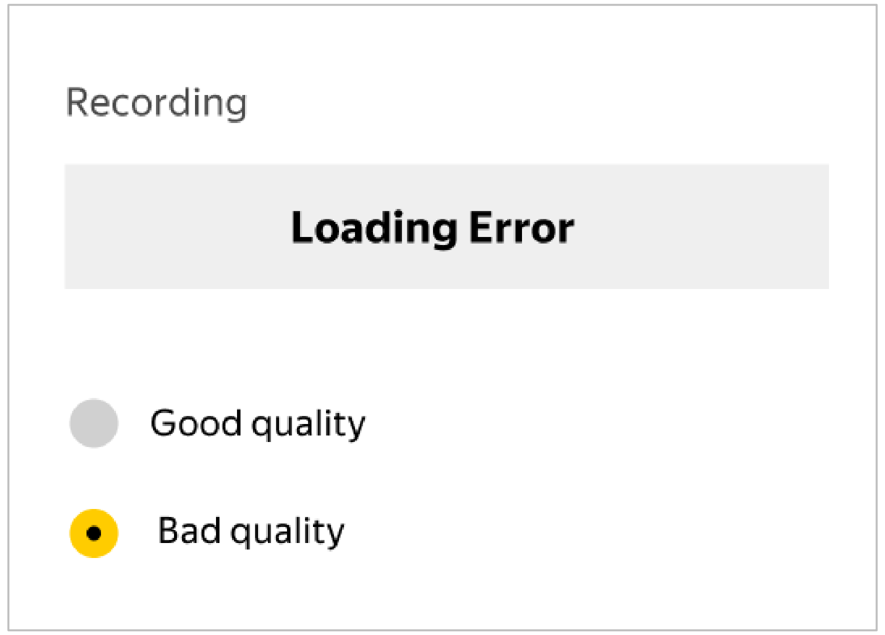
Recording

▶ 0:00 / 0:05 🔊

○ Good quality

● Bad quality

IOS App



Recording

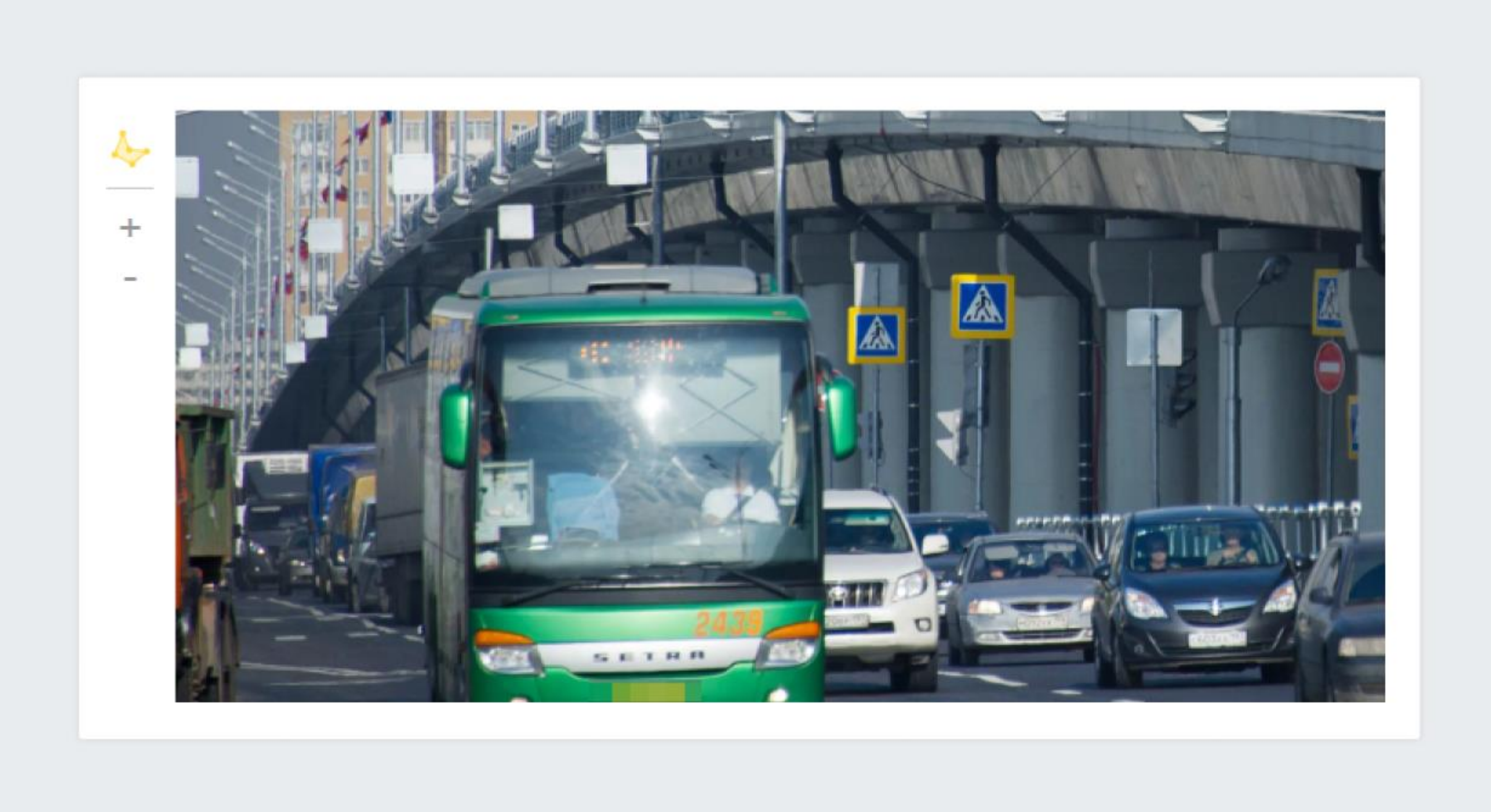
Loading Error

○ Good quality

● Bad quality

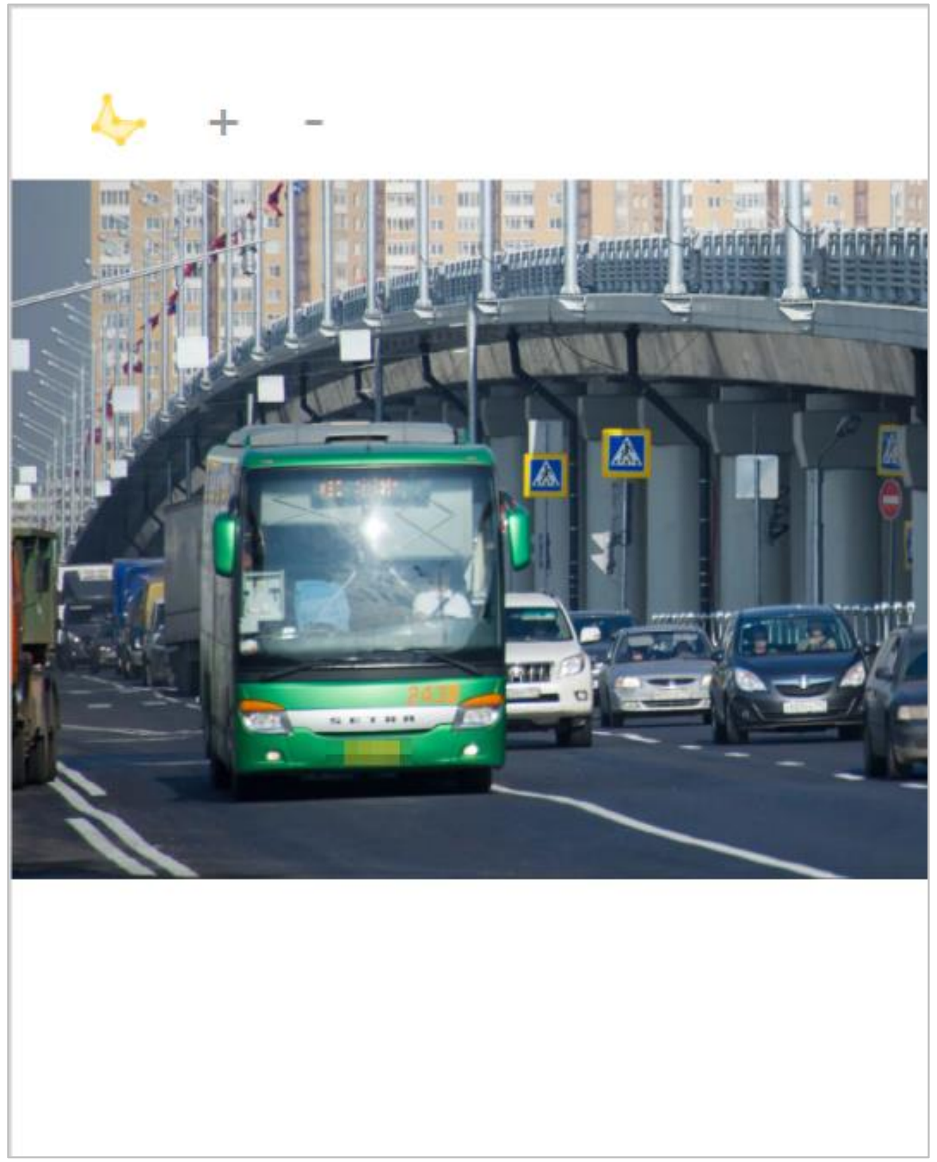
Rule #1. Cross-platform compatibility

Task: draw a polygon around every road sign



Rule #1. Cross-platform compatibility

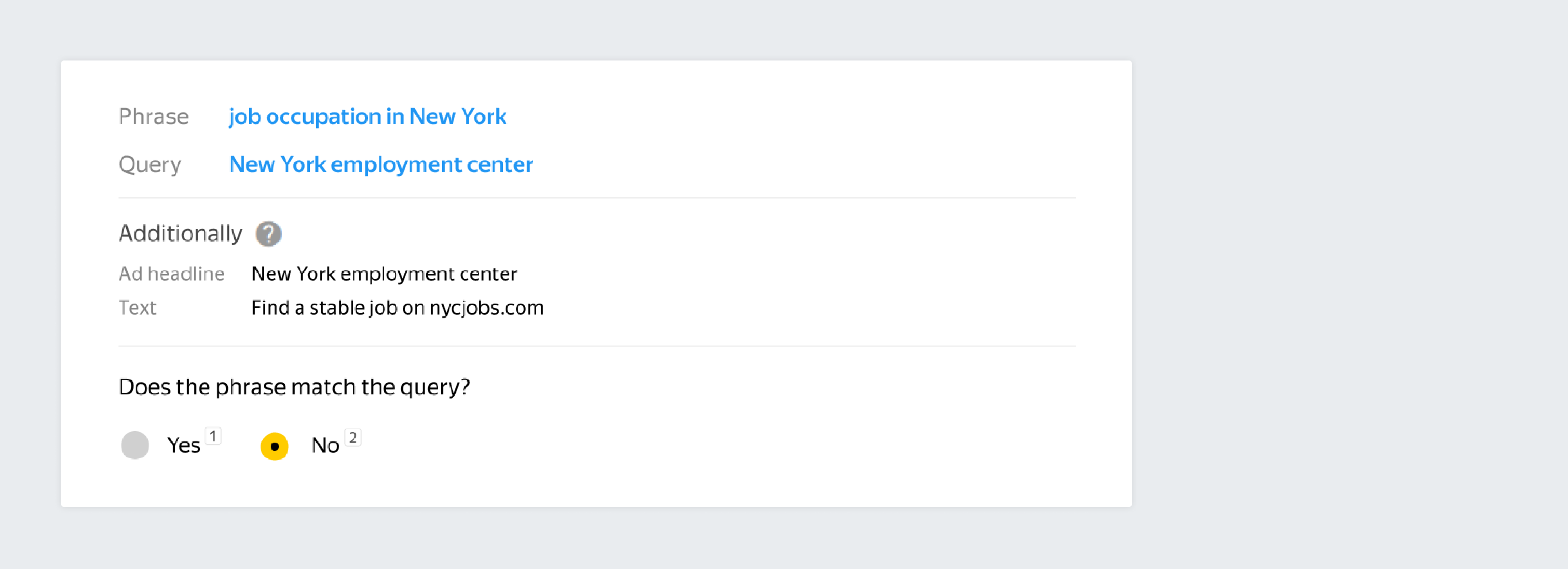
Task: draw a polygon around every road sign



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

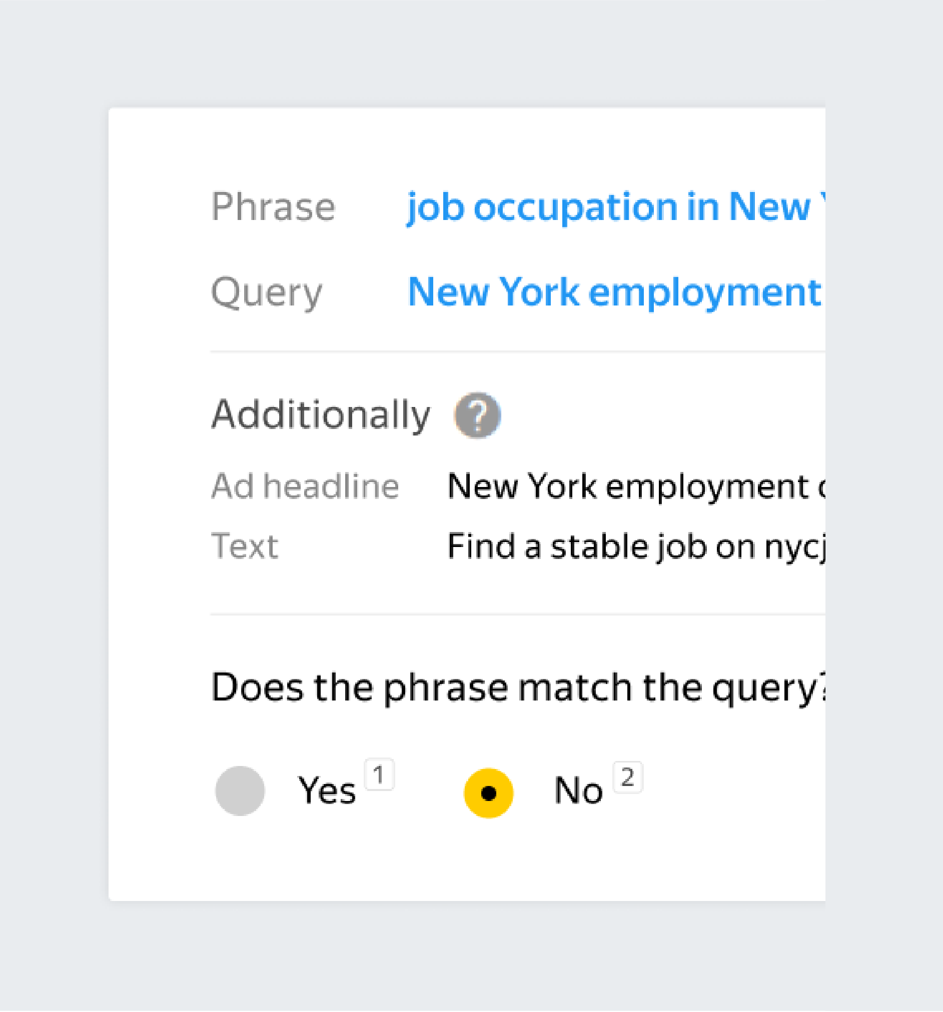
Text Find a stable job on nycjobs.com

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 Y

Query New York employment

Additionally ?

Ad headline New York employment c

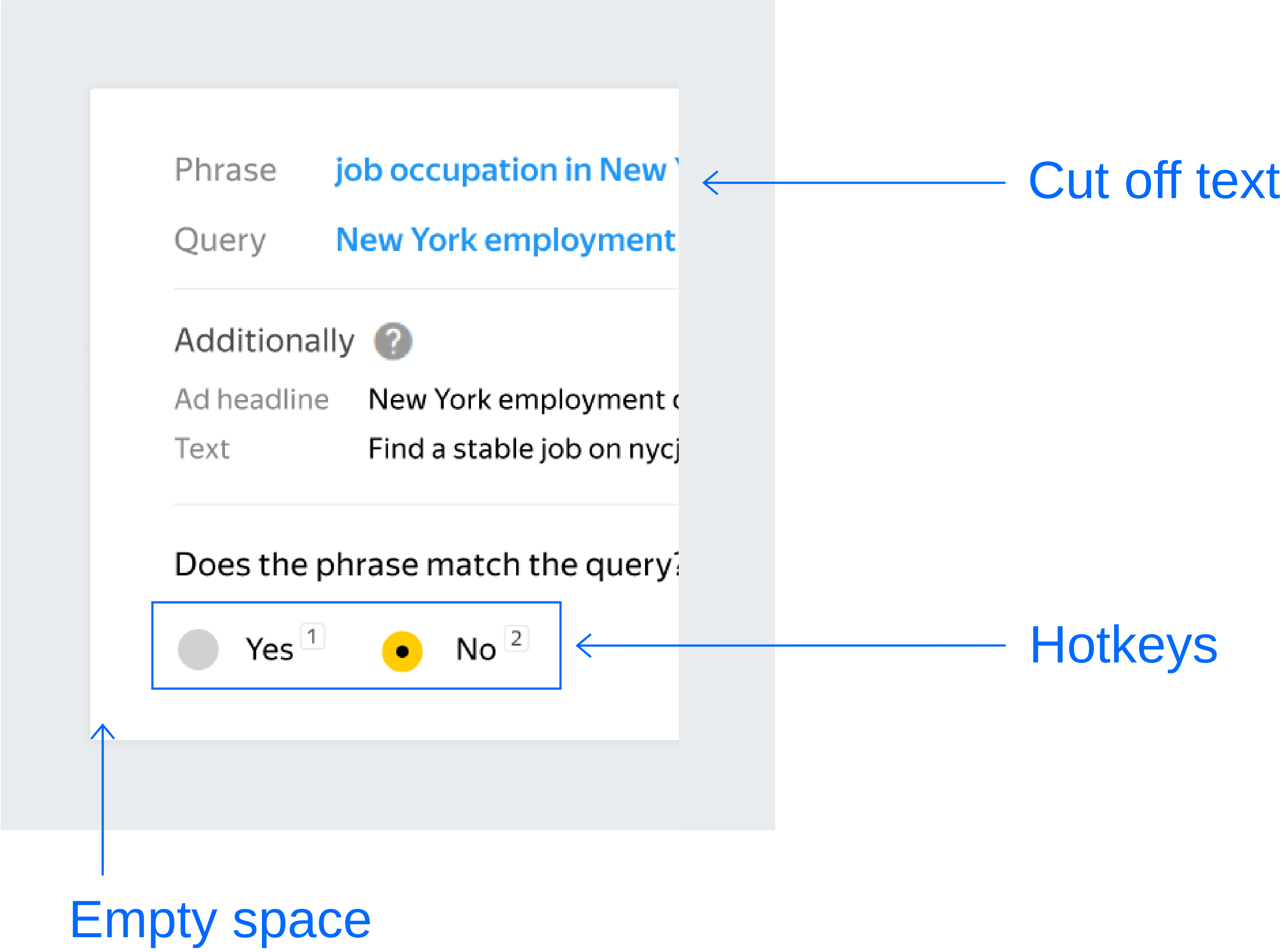
Text Find a stable job on nycj

Does the phrase match the query?

○ Yes [1] ● No [2]

Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match



Phrase job occupation in New

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

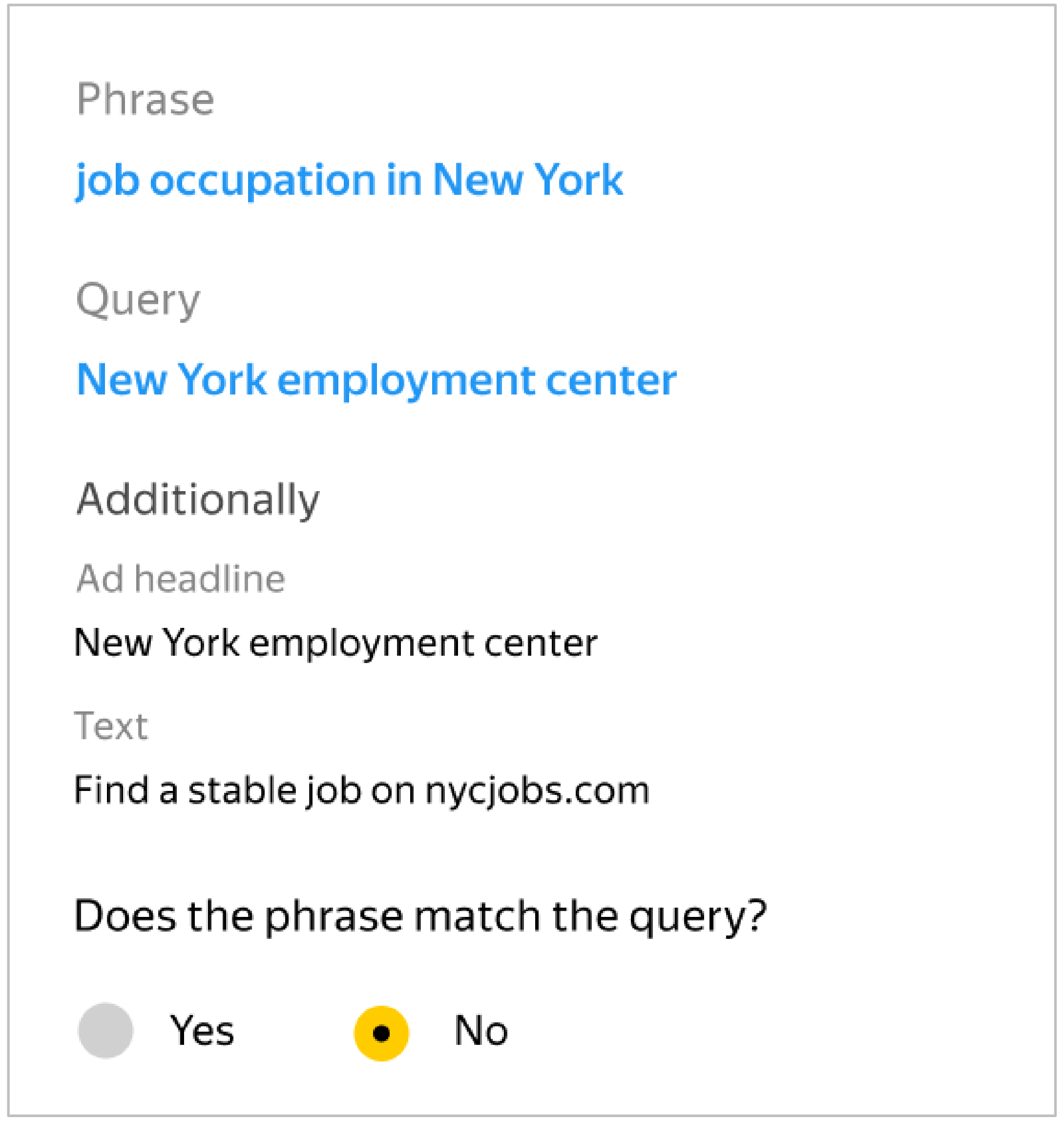
Cut off text

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

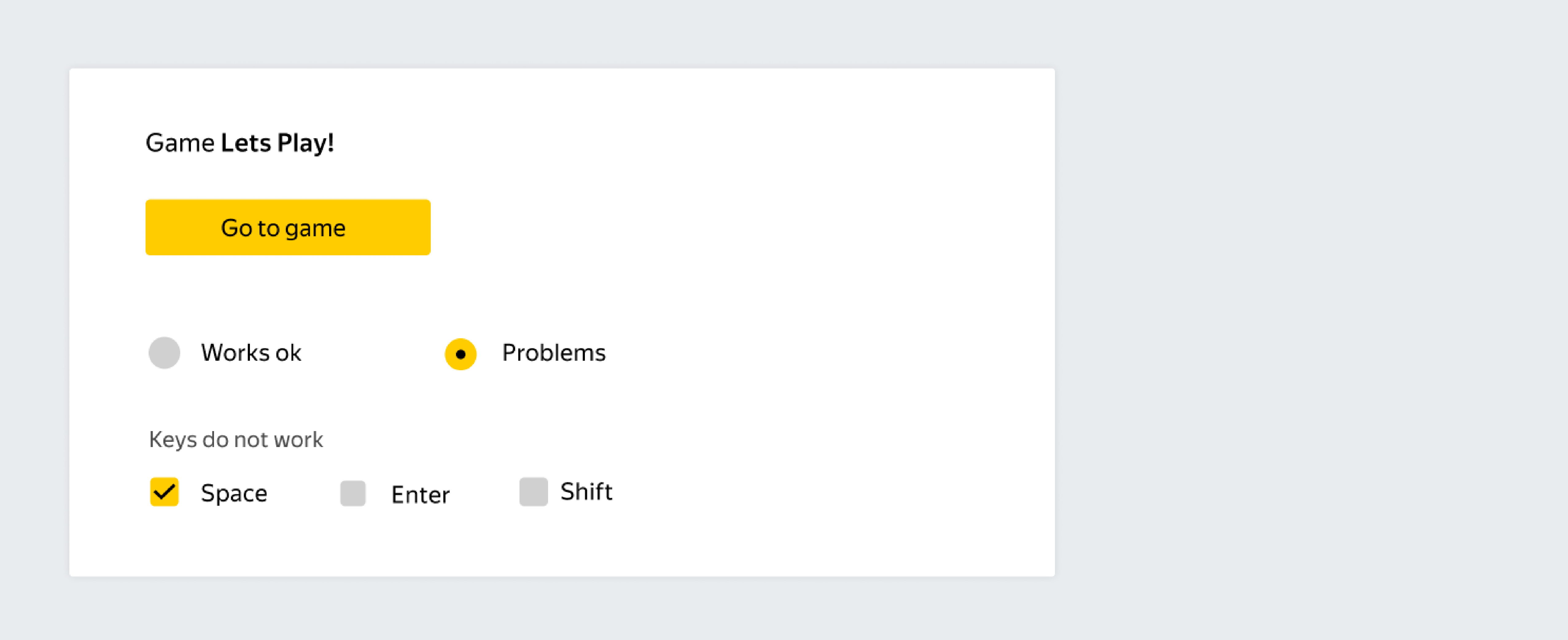
Rule #2. Hotkeys

- 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)



Game **Lets Play!**

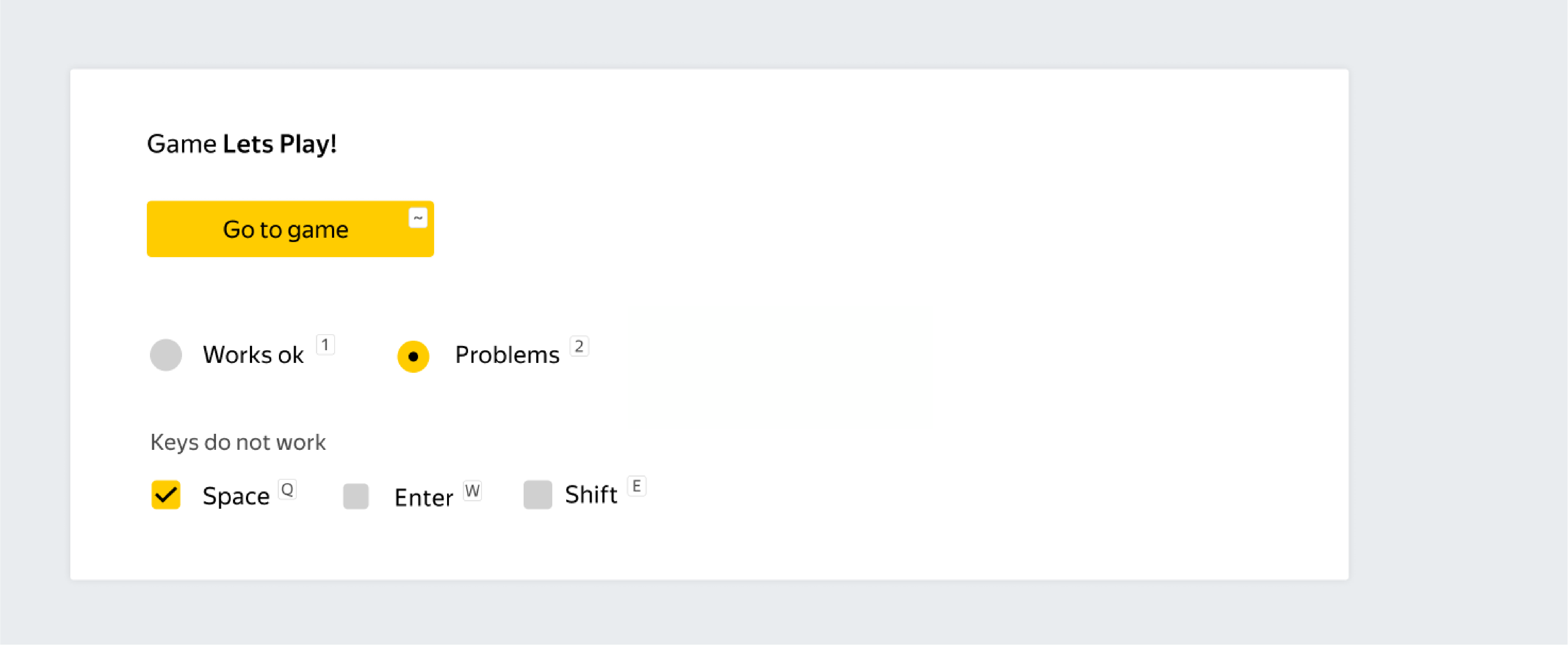
○ Works ok ● Problems

Keys do not work

☑ Space □ Enter □ Shift

Rule #2. Hotkeys

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



Game **Lets Play!**

Go to game ~

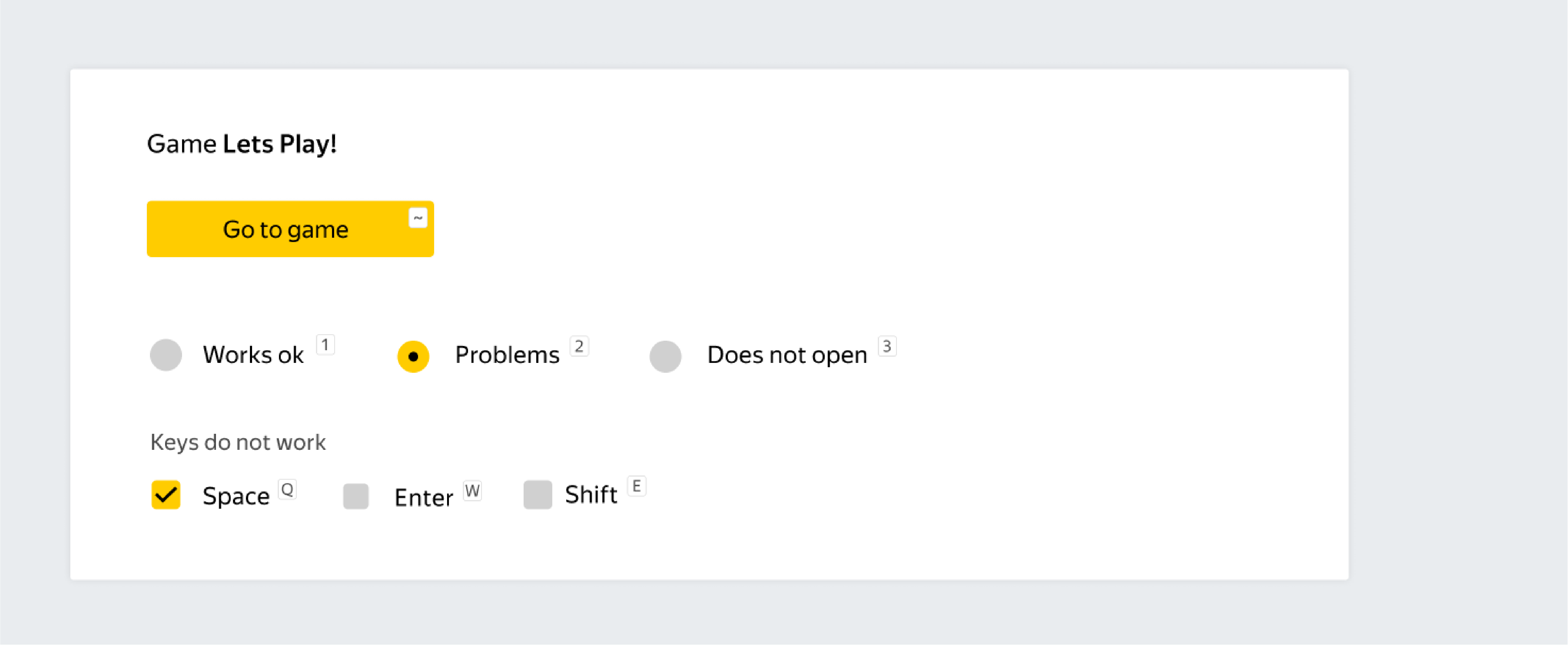
○ Works ok 1   ● Problems 2

Keys do not work

☑ Space Q   □ Enter W   □ Shift E

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

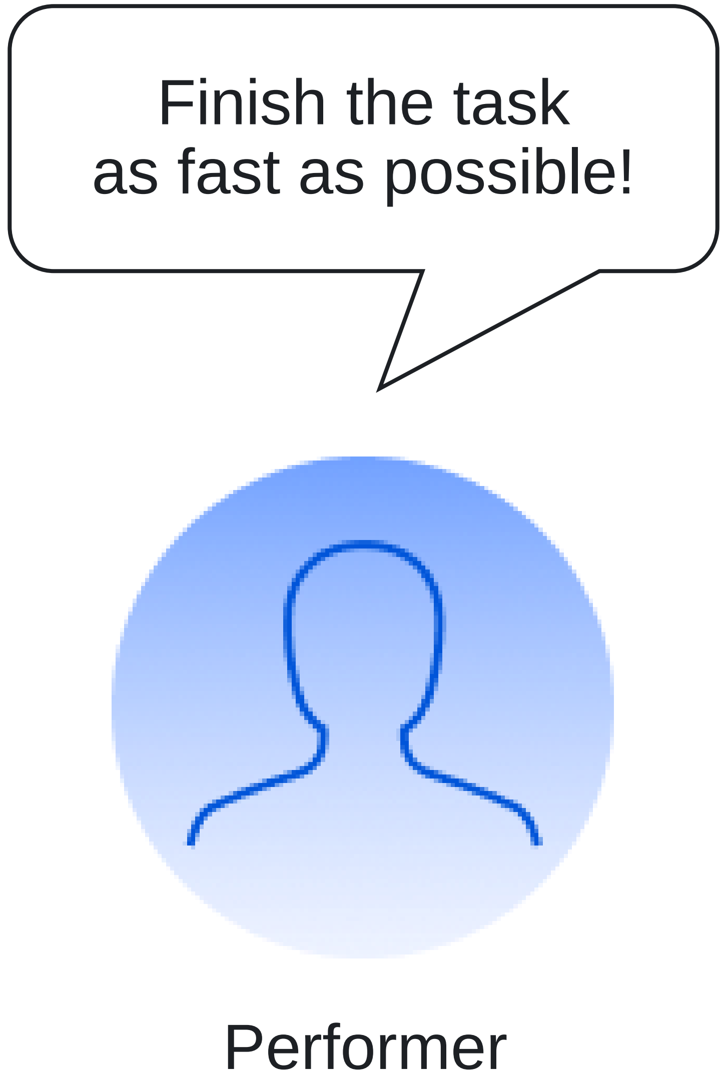
Keys do not work

☑ Space Q   □ Enter W   □ Shift E

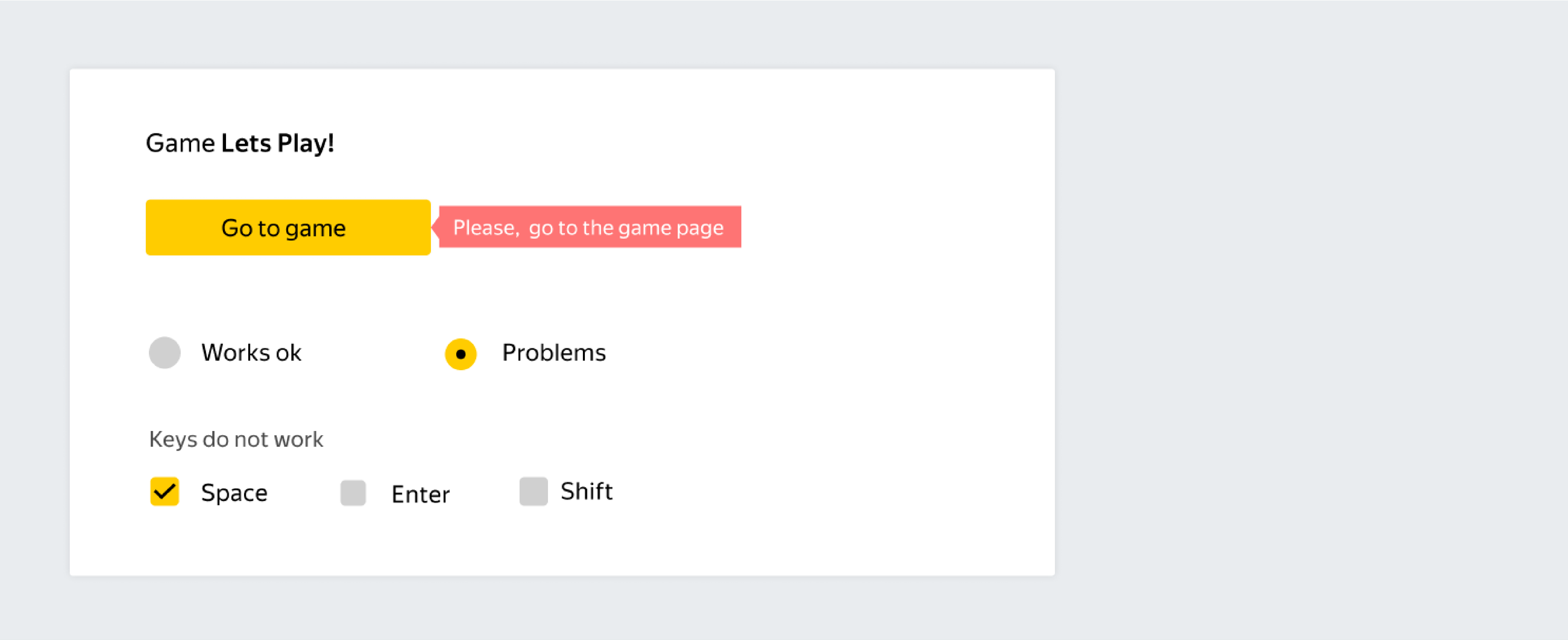
Rule #3. Action and data check

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



Rule #3. Action and data check



Game **Lets Play!**

Please, go to the game page

○ Works ok ● Problems

Keys do not work

☑ Space □ Enter □ Shift

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

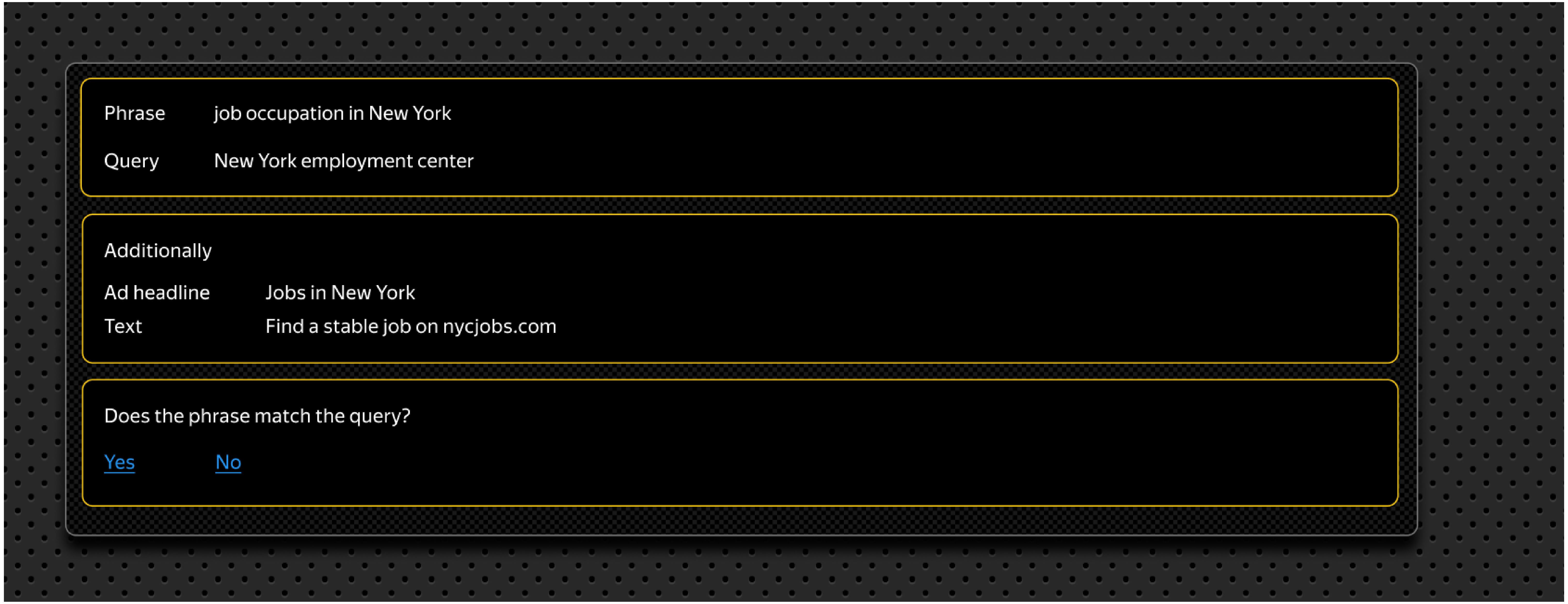
Idea: show screenshots instead of the links

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



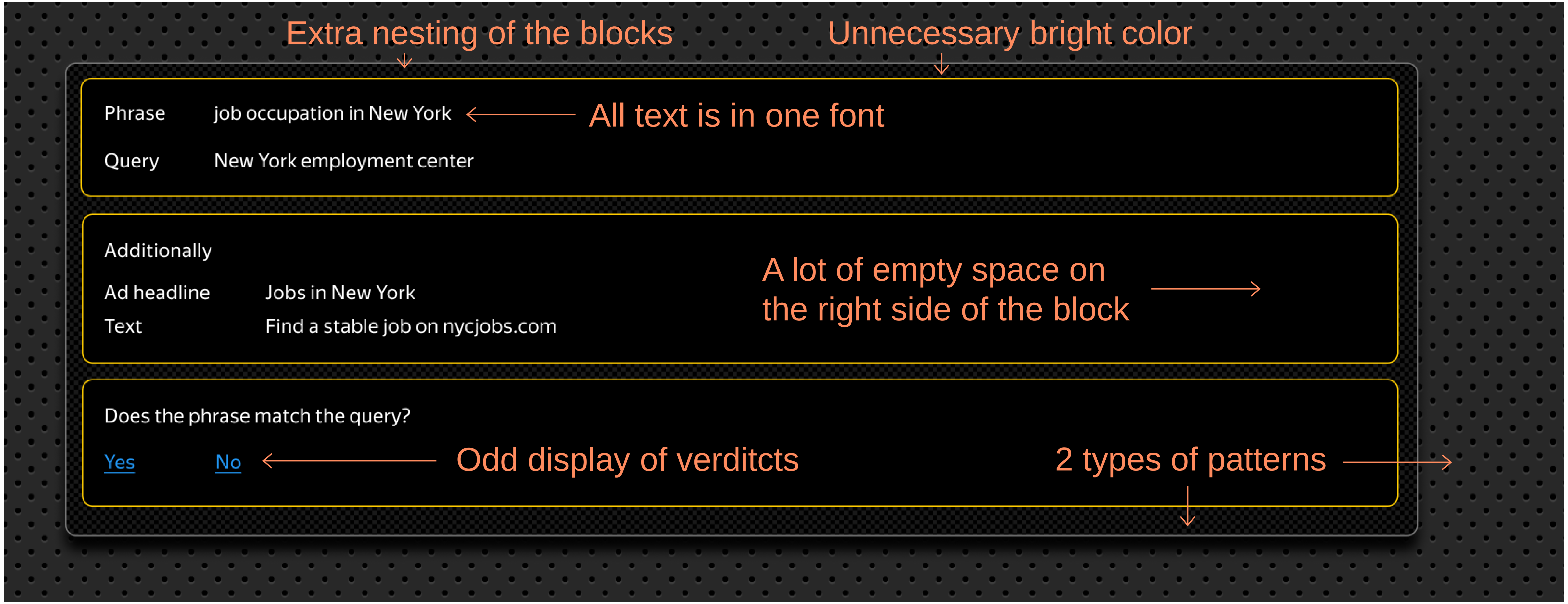
Phrase	job occupation in New York
Query	New York employment center

Additionally

Ad headline	Jobs in New York
Text	Find a stable job on nycjobs.com

Does the phrase match the query?

Rule #6. Avoid experimental design



Phrase job occupation in New York

Query New York employment center

Additionally

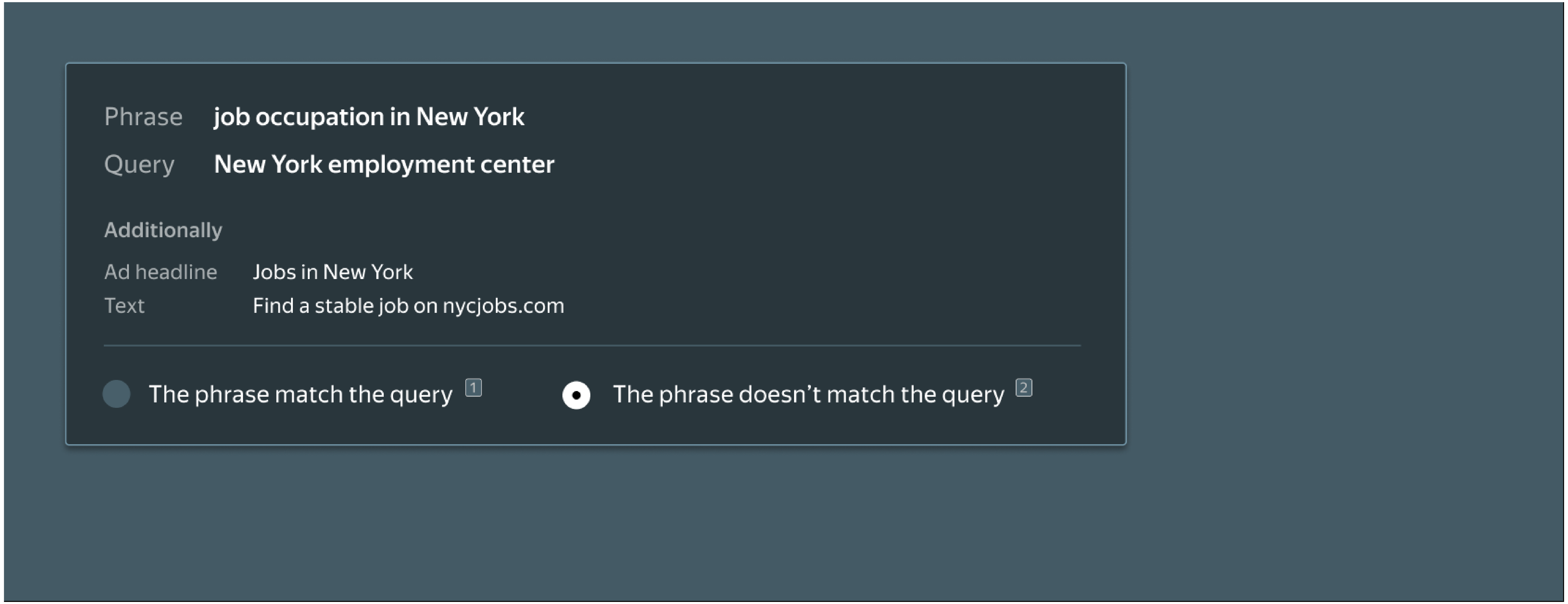
Ad headline Jobs in New York

Text Find a stable job on nycjobs.com

Does the phrase match the query?

Yes No

Rule #6. Avoid experimental design



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

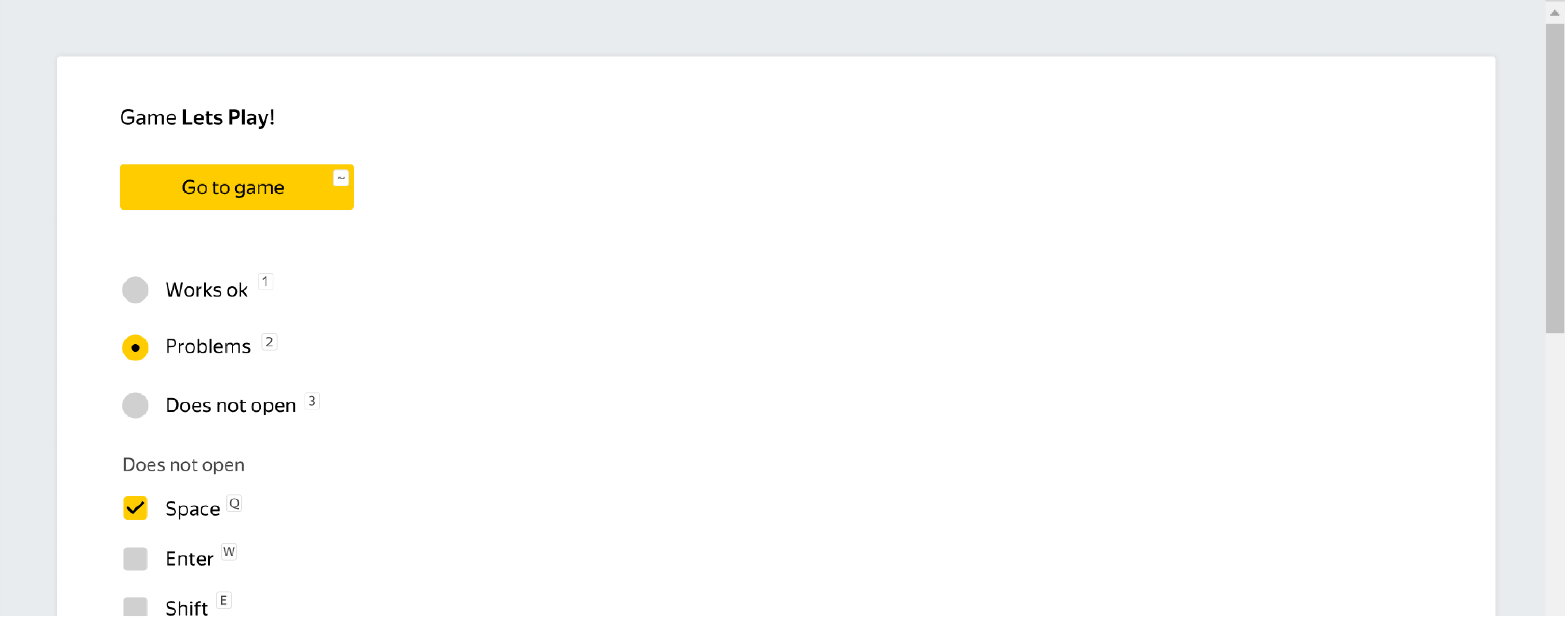
○ The phrase match the query 1 ◉ The phrase doesn’t match the query 2

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



Game **Lets Play!**

○ Works ok 1

● Problems 2

○ Does not open 3

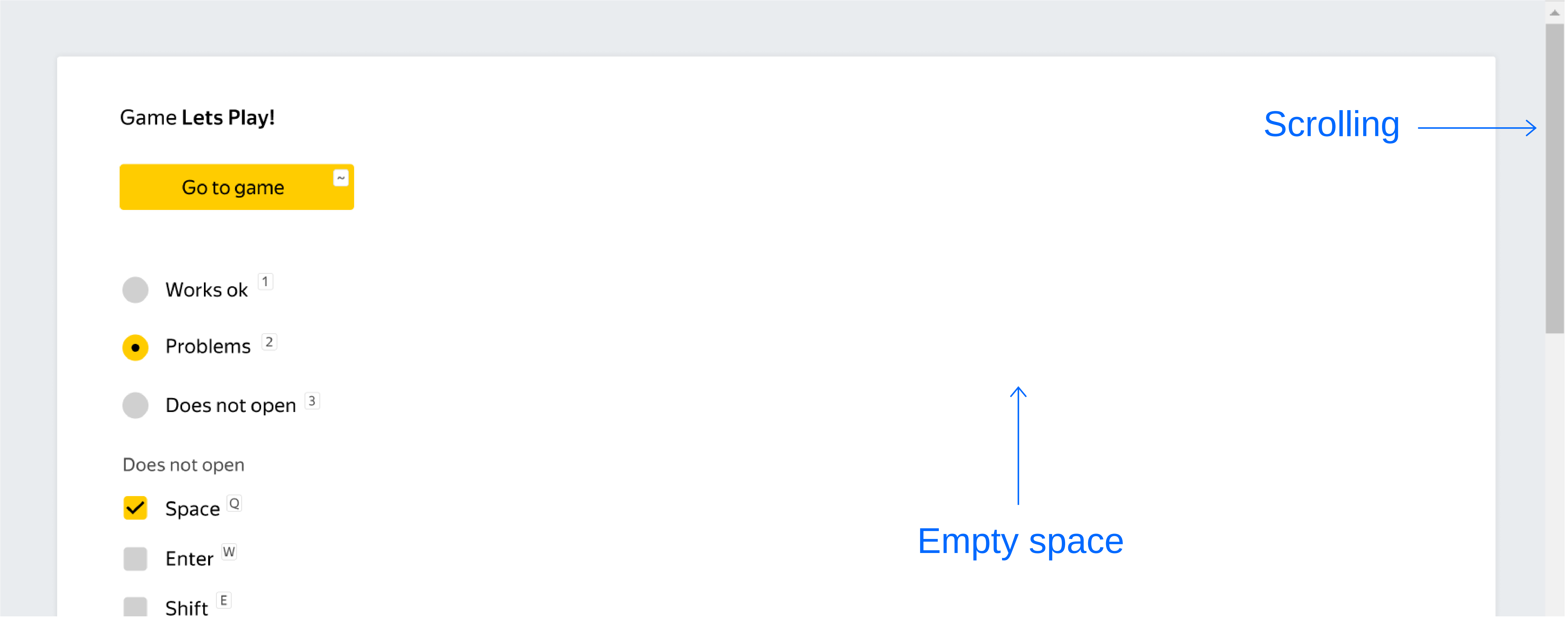
Does not open

☑ Space Q

□ Enter W

□ Shift E

Rule #7. Efficient space usage



Game **Lets Play!**

○ Works ok 1

● Problems 2

○ Does not open 3

Does not open

☑ Space Q

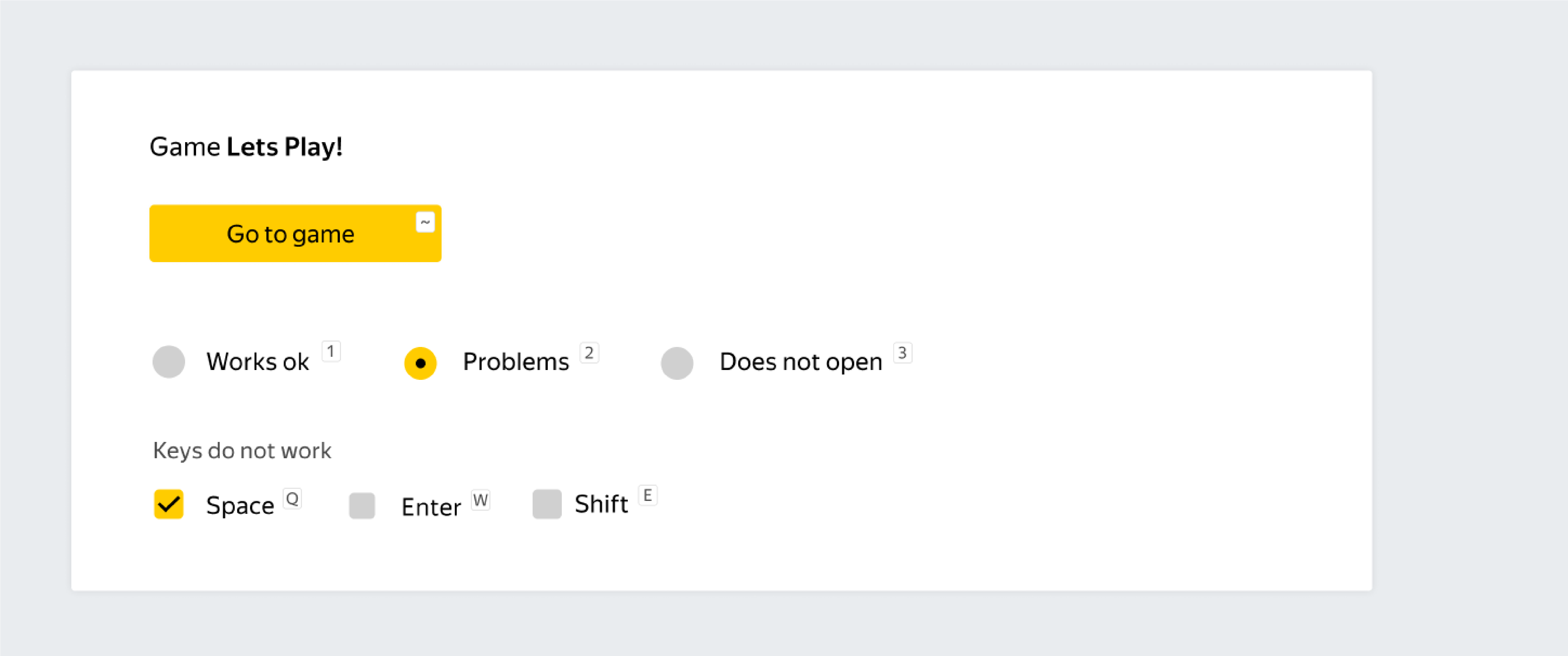
□ Enter W

□ Shift E

Scrolling

Empty space

Rule #7. Efficient space usage



Game **Lets Play!**

○ Works ok 1 ● Problems 2 ○ Does not open 3

Keys do not work

☑ Space Q □ Enter W □ Shift E

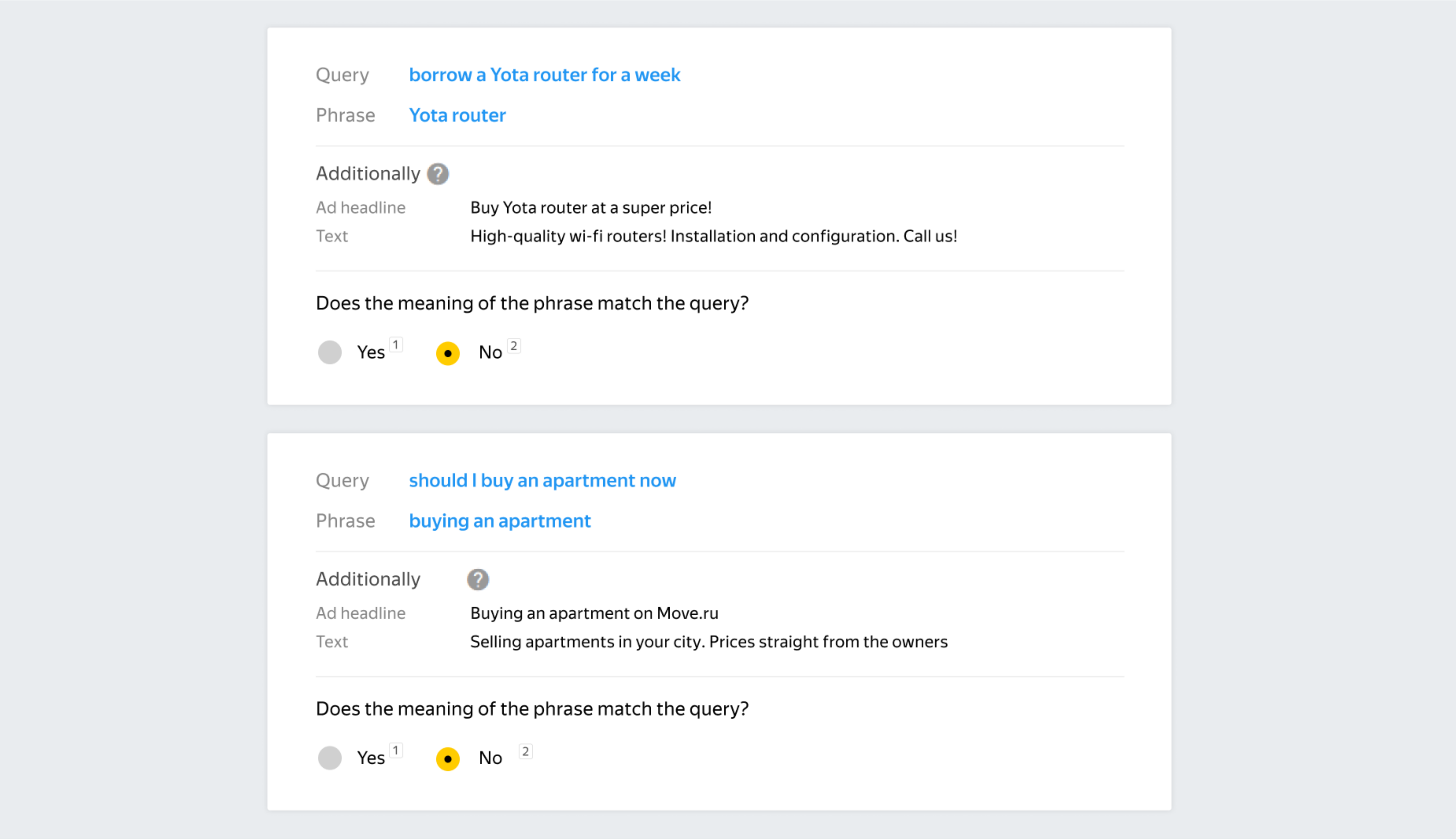
Rule #8. Constructing task suit

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

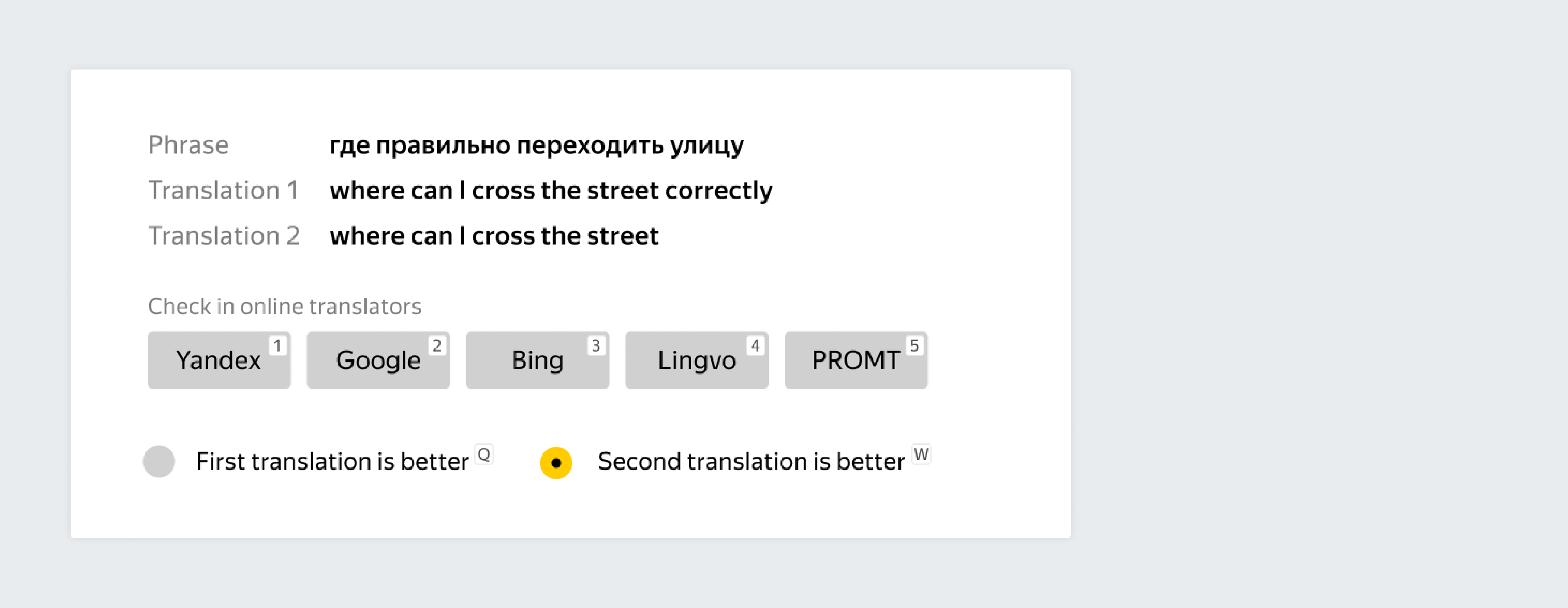
- Buttons
- Links
- Images
- Other elements, that with a particular function

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

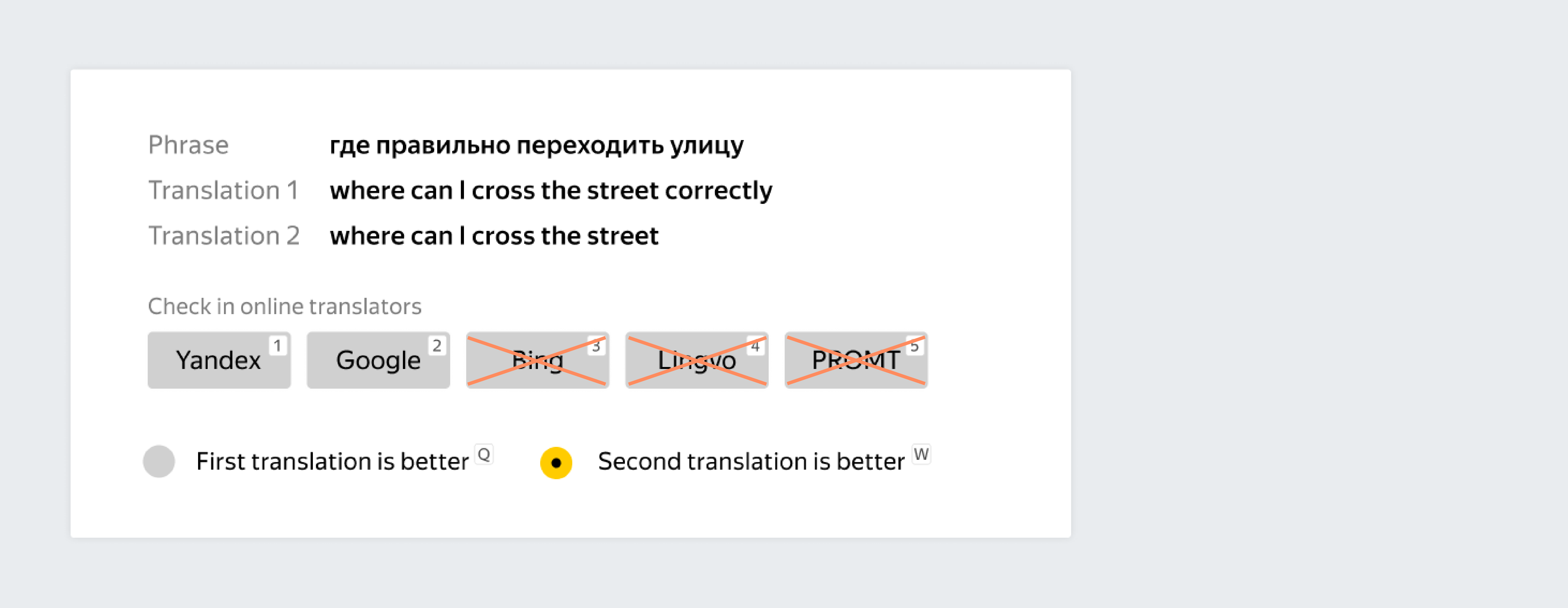
Check in online translators

○ 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

Yandex 1 | Google 2 | ~~Bing~~ 3 | ~~Lingvo~~ 4 | ~~PROMT~~ 5

○ 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