

School news with checks: self-critique and a gatekeeper



Guide 7 for the AI Agents workshop

LeX Consultancy B.V. · 20 September 2026

In guide 3 you built School news: one core message becomes a piece for the website, the parent app and social media. The texts were good, but they went out unchecked. This guide adds two things that the *Design patterns* appendix calls pattern 4 and 5:

Addition	Pattern	What it does
Checker (LLM)	Self-critique	Reads the parent app message once more against the style guide and improves it
Gatekeeper (Code) plus IF/ELSE	Gatekeeper	Checks hard rules with code: word count, forbidden words, hashtag, missing information. If everything is right, out it goes; otherwise to the communications officer with the reasons

The difference between those two is the heart of this guide. The Checker is a model reviewing a model: good for form and tone, but it can talk itself into things. The Gatekeeper is six lines of Python that cannot be talked into anything: 81 words is too many, full stop. Together they deliver what a communications officer wants: less checking work, and certainty that the rules that really matter really apply.

This guide follows the cloud version of Dify of 20 September 2026. If your screen differs, follow what you see.

How to read this guide

Form	Meaning
1. 2. 3.	What you do: click, choose, open
Bold	A button, menu or field as it appears on your screen
Table <i>What you type</i>	Values you put in a form field
Grey box <i>Type in</i>	Text you type or paste literally
Orange box	A pitfall you would otherwise discover yourself
Green box <i>Your choice</i>	A place where you can replace our example with your own material, style or prompt

The design card for this app

Before we started clicking, this card was filled in (the blank card is in the Design card appendix). Fill in the same five boxes for your own app; the loose version of this filled-in card is in 00-design-card/examples.

1. Who is it for	2. What goes in
The communications officer who has to write every message three times (website, parent app, social) and has no time to check every version against the style guide.	Three fields: the core message (who, what, where, when), the class or year group, and the tone (informative, celebratory, urgent).
3. What comes out	4. What it must stick to
If approved: three versions ready to post. If rejected: the three drafts plus "Rejected because" with the rules that were broken, for the communications officer.	The style guide from the knowledge base; the Checker (second LLM) rewrites the parent app version; the Gatekeeper (Code + IF/ELSE) measures word count, forbidden words, [FILL IN] and the hashtag. What does not pass the gate does not go out.

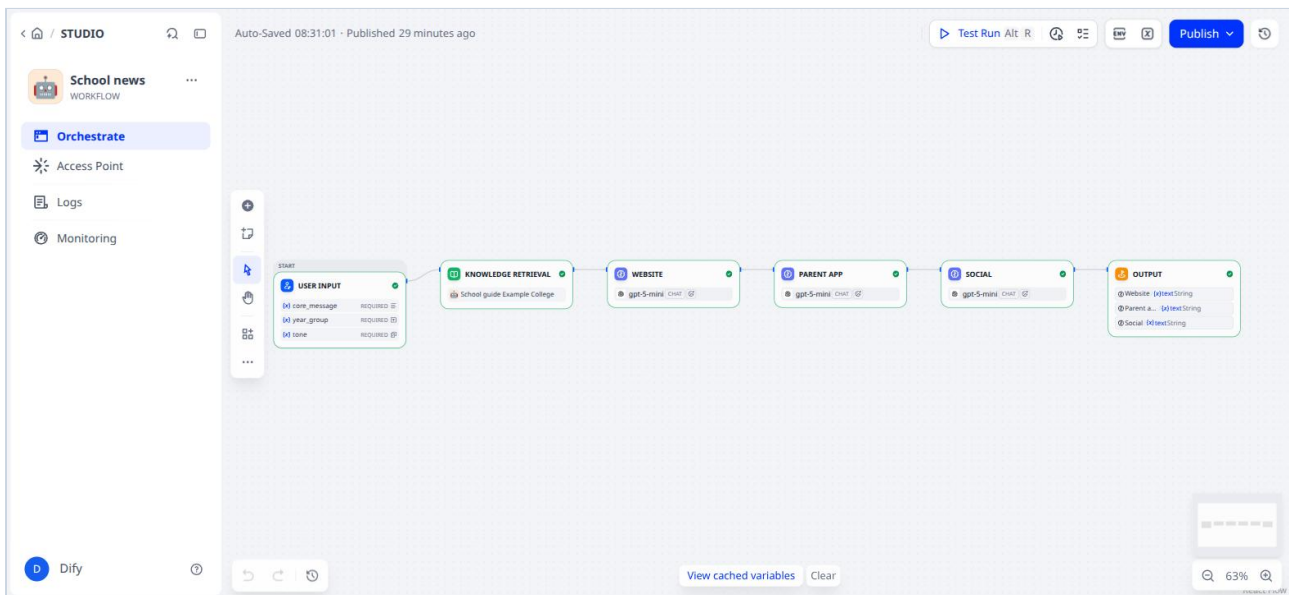
5. How you know it works

Test	What goes in	What must come out
1	A complete message about sports day, celebratory	Approved, three versions within the word limits, with #ExampleCollege
2	A message without a date	Rejected: contains [FILL IN]
3 (the hard one)	A message with "kindly" and "asap"	Rejected with the two forbidden words named

Preparation

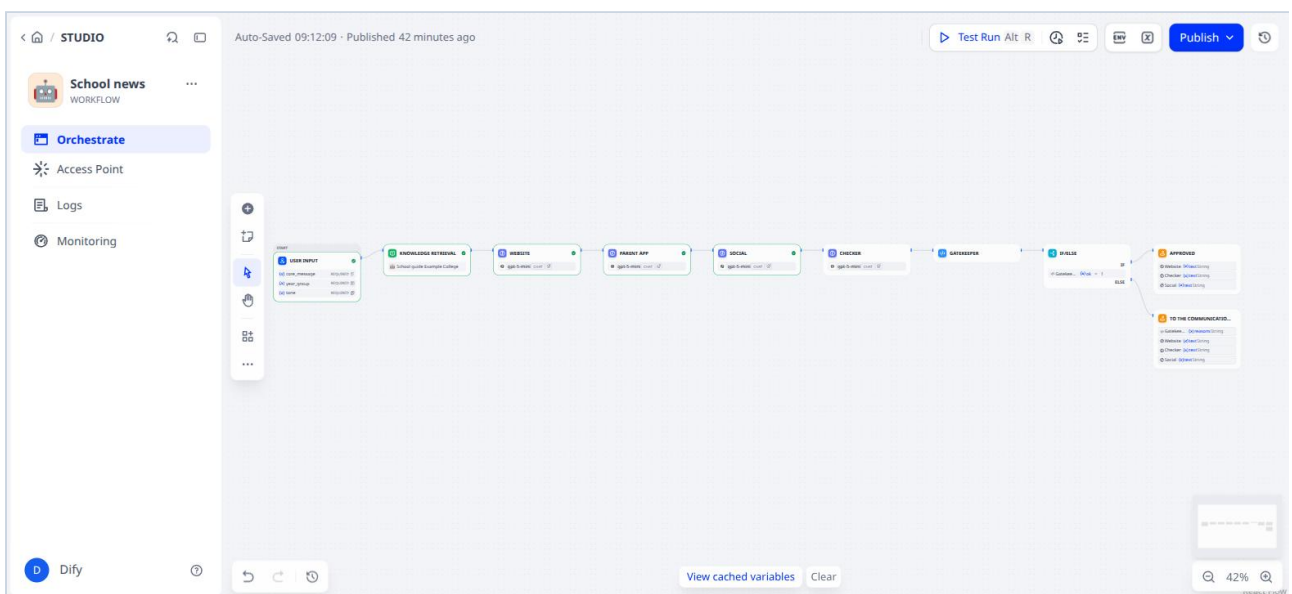
- The **School news** workflow from guide 3 is on your account and works.
- You know what an LLM block and an Output block are (guides 1 and 3).
- You do not need to be able to program. You paste the code; what it does is explained next to it.

This is what School news looks like before you start:



Starting point: the workflow from guide 3

And this is what the tail looks like when you are done:



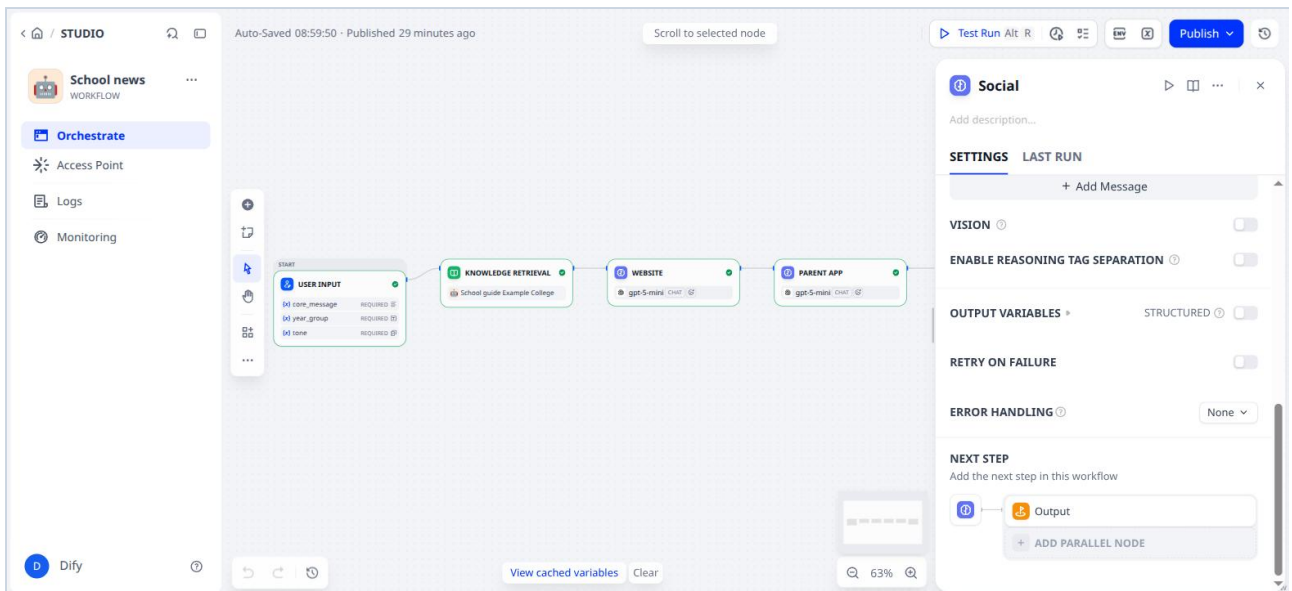
The goal: Checker, Gatekeeper, IF/ELSE and two exits

Part A. The Checker (self-critique)

Step 1. Disconnect the Output from Social

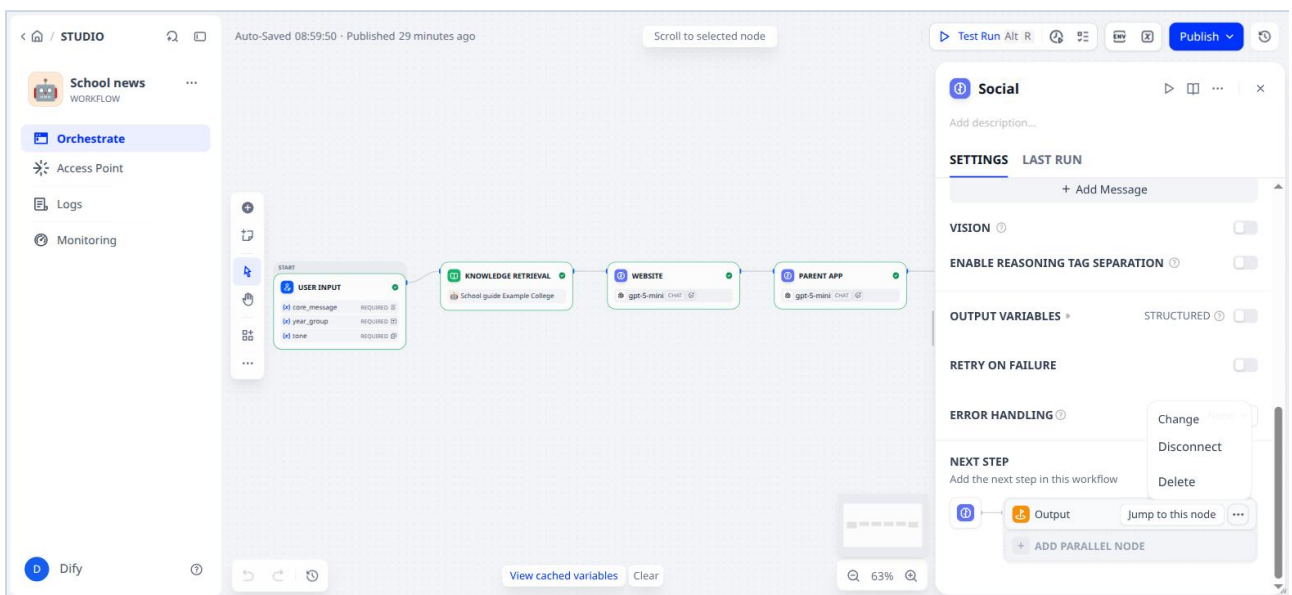
The new blocks go between Social and Output. So you first disconnect Output.

1. Open **School news** in **Studio** and click the **Social** block.
2. In the right-hand panel scroll to **NEXT STEP**. It says **Output**.

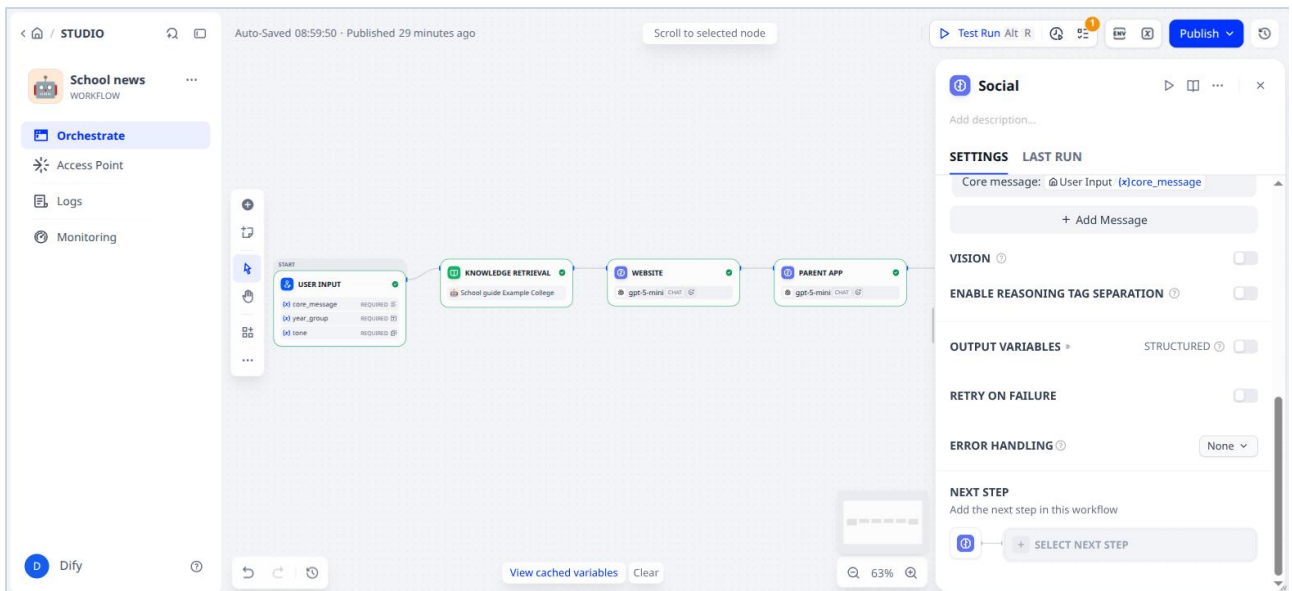


Next Step of Social: Output

- Click the three dots behind Output and choose **Disconnect**. (Not **Delete**: the block may stay, only the line goes.)



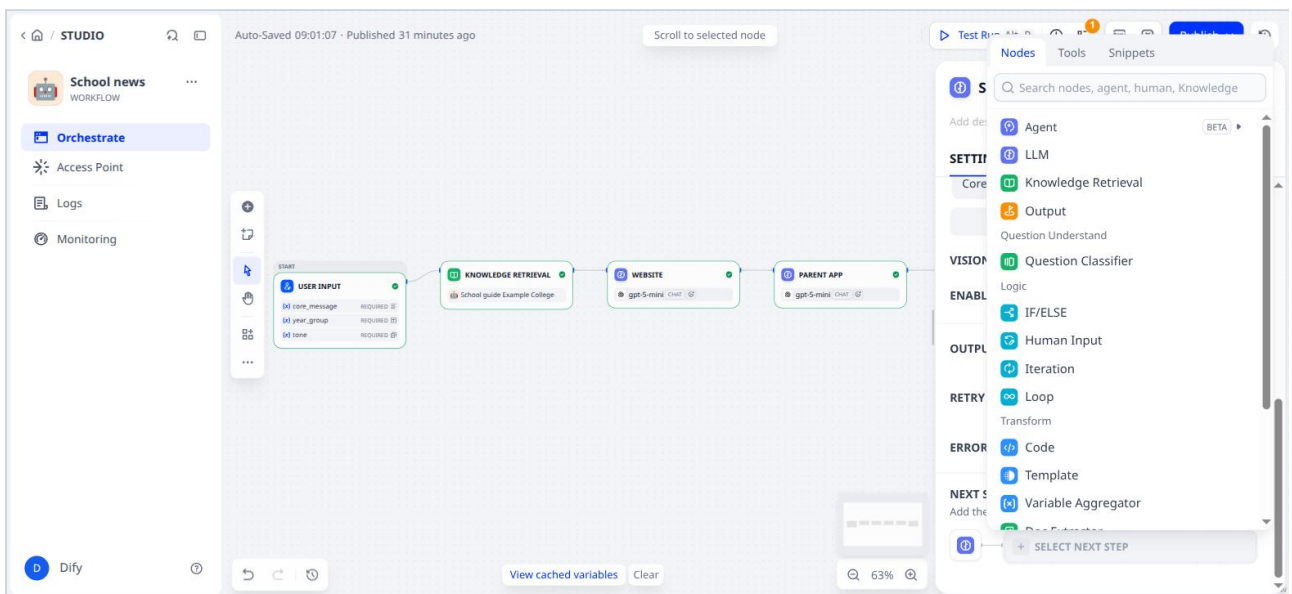
Change, Disconnect, Delete



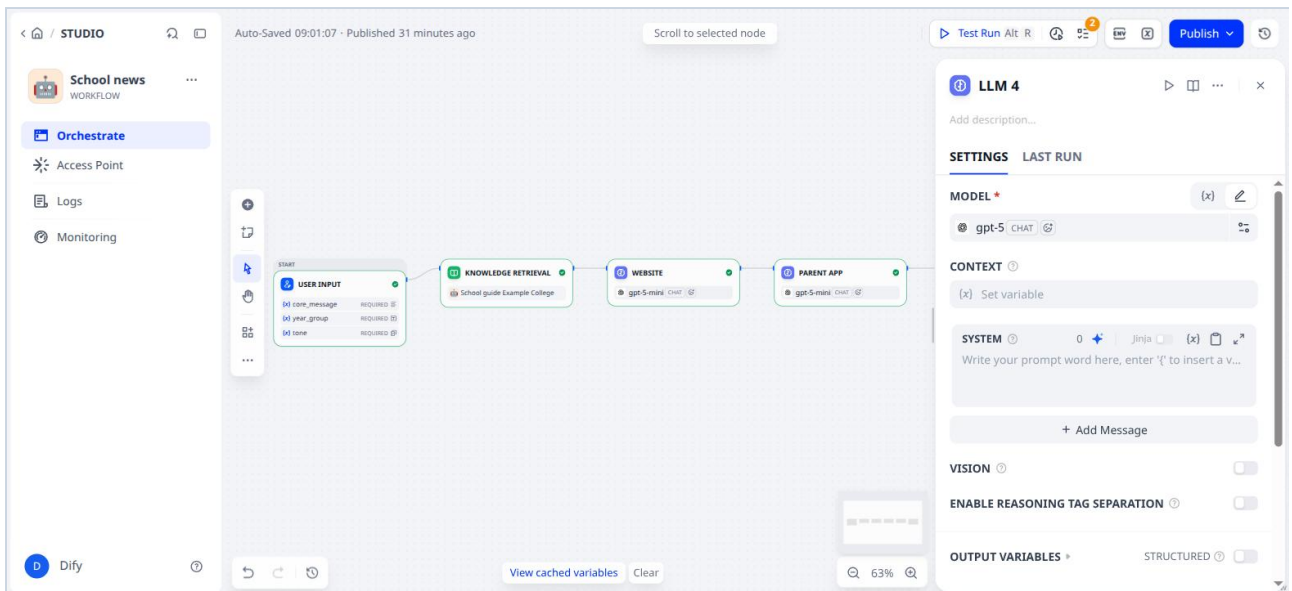
Social now has no next step

Step 2. Add the Checker

1. Under **NEXT STEP** click **Select next step** and choose **LLM**.



The list of blocks



The new, empty LLM block

2. **MODEL:** gpt-5-mini. **CONTEXT:** Knowledge Retrieval / result (the style guide).
3. Paste the prompt below into **SYSTEM**. For `{Context}` type a forward slash and choose **Context**; for `{Parent app/text}` choose the item **text** under **PARENT APP**.

TYPE IN

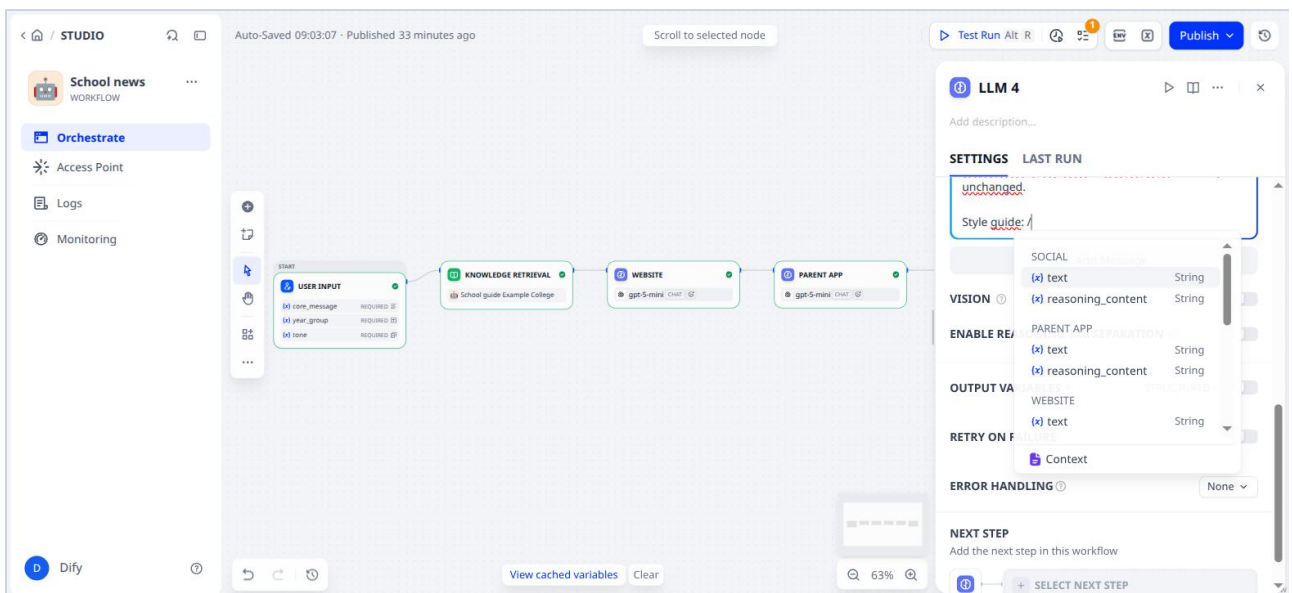
You are the checker of Example College. Below is a draft message for the parent app. Assess it on these points from the style guide: at most 80 words; parents are addressed formally; one message, one date and one action; no abbreviations and no words from the list of words to avoid; no names of students; reading level B1 (short sentences, everyday words).

Work in two steps. Step 1: briefly name what is wrong (or write: nothing). Step 2: give the improved version. Invent no facts; what is not in the draft stays [FILL IN]. Do not remove facts, dates or times that are in the draft; a deadline for signing up belongs to the action and stays. Do not change digits into words.

As your answer give **ONLY** the improved text, without step 1, without a heading and without explanation. If the draft is already good, return it unchanged.

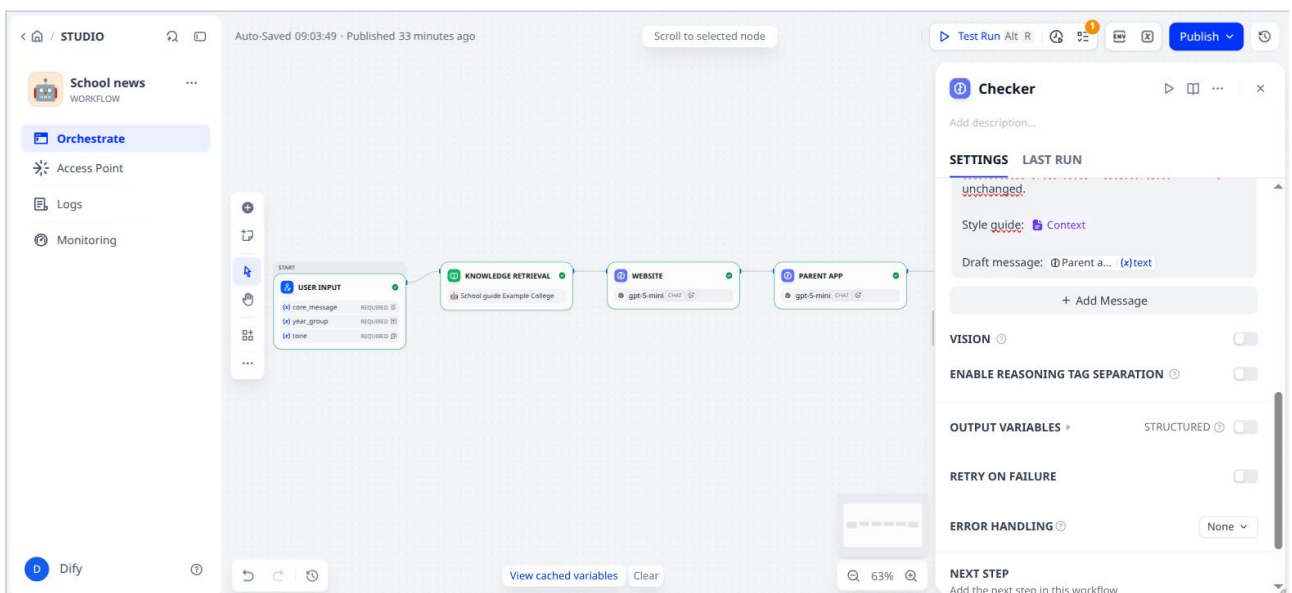
Style guide: `{Context}`

Draft message: `{Parent app/text}`



The prompt, with the menu after a forward slash

4. Rename the block to **Checker**.



Checker ready: context and draft message are connected

“Work in two steps, but give only step 2.” That is not a contradiction but a trick: the model assesses first (which improves the result) and you still get clean text back that you can pass on directly. If you leave step 1 in the answer, that explanation ends up in the parent app too.

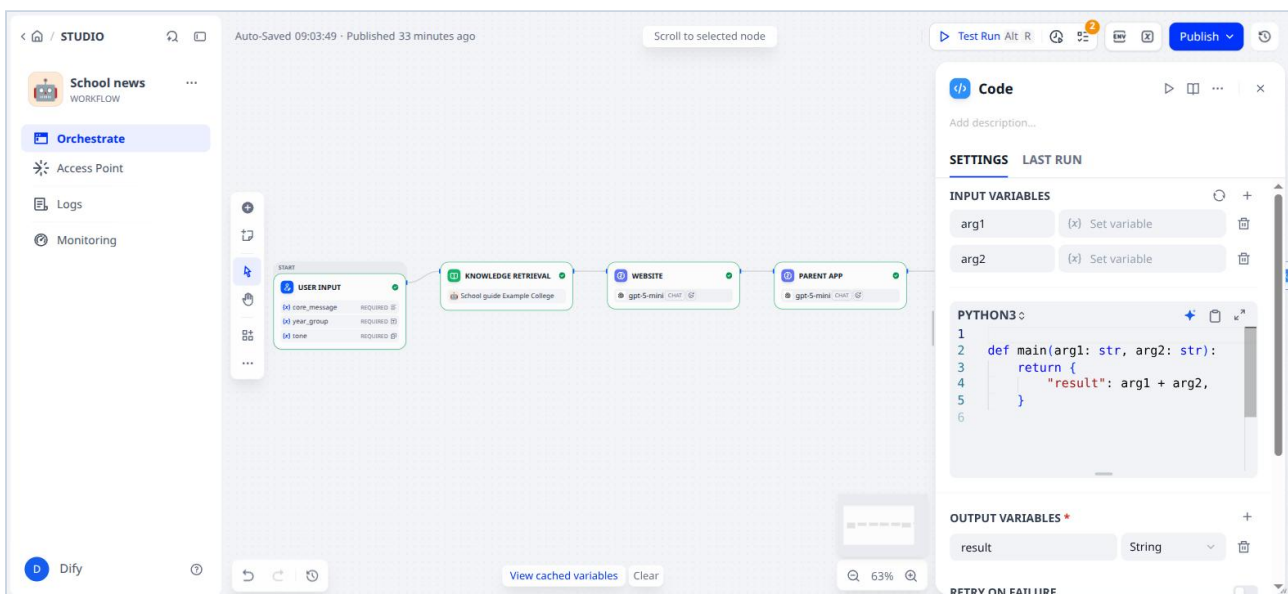
In our first test the Checker did exactly what we asked, and that was too much: it read “one date” as “remove the sign-up deadline”, and “Year 8” became “year eight”. That is why the two sentences about facts and digits are in the prompt. Self-critique improves form; what may change in content is for you to limit.

Your choice: what the Checker checks. The first paragraph of the prompt is your checklist. Also want to check “is there a way to get in touch”? Or the tone per channel? Add it. Want the Checker on the website piece too? Copy the block (three dots on the block, **Duplicate**), connect Website/text as the draft and adjust the rules (200 words, eight-word headline).

Part B. The Gatekeeper (code)

Step 3. Add a Code block

1. At Checker: **NEXT STEP** → **Select next step** → under Transform: **Code**.



The Code block with sample code

2. Click in the editor, select everything (Ctrl+A) and paste the code below.

TYPE IN

```
def main(website: str, parent_app: str, social: str) -> dict:
    reasons = []
    forbidden = ["by means of", "with regard to", "asap", "w.r.t.", "kindly",
"parent(s)/carer(s)", "utilise"]
    texts = (("website", website, 230), ("parent app", parent_app, 90),
("social", social, 70))
    for name, text, maximum in texts:
        count = len(text.split())
        if count > maximum:
            reasons.append(f"{name}: {count} words, maximum {maximum}")
        if "[FILL IN]" in text:
            reasons.append(f"{name}: contains [FILL IN], information is
missing")
        for word in forbidden:
            if word in text.lower():
                reasons.append(f"{name}: contains the word to avoid
'{word.strip()}'")
        if "#examplecollege" not in social.lower():
            reasons.append("social: #ExampleCollege is missing")
    return {
        "ok": 0 if reasons else 1,
        "reasons": "\n".join(reasons) if reasons else "All rules satisfied.",
    }
```

The screenshot displays the Dify Studio interface. On the left, a sidebar shows the 'School news' workflow, with options for 'Orchestrate', 'Access Point', 'Logs', and 'Monitoring'. The main workspace shows a workflow diagram with four nodes: 'START', 'USER INPUT', 'KNOWLEDGE RETRIEVAL', 'WEBSITE', and 'PARENT APP'. The 'USER INPUT' node is expanded, showing required variables: 'core_message', 'year_group', and 'none'. On the right, a 'Code' editor is open, displaying the Python code for the 'PARENT APP' node. The code defines a 'main' function that checks for forbidden words and missing information, returning a dictionary with 'ok' and 'reasons' keys. The 'SETTINGS' panel on the right shows 'INPUT VARIABLES' (arg1, arg2) and 'OUTPUT VARIABLES' (result).

The code is in place

What the code does, line by line in plain words:

Line	Meaning
<code>forbidden = [...]</code>	The list of words that must never be in a text. This is the list from the style guide, made hard here
<code>texts = (...)</code>	The three texts with their maximum in words. The maxima are slightly above the style guide (200, 80, 60), because the headline and the label WEBSITE count too
<code>if count > maximum</code>	Too long: note a reason
<code>if "[FILL IN]" in text</code>	The model did not know something: note a reason. This is the most important rule: a message with a gap never goes out
<code>for word in forbidden</code>	If a forbidden word is in the text: note a reason
<code>if "#examplecollege" not in ...</code>	The social post must have the school hashtag
<code>return {...}</code>	Return two things: <code>ok</code> (1 = good, 0 = wrong) and <code>reasons</code> (the list, or "All rules satisfied.")

Mind the indentation: in Python every space at the start of a line counts. Pasting with Ctrl+V works fine. Typing it over almost always goes wrong; do not do that.

Step 4. Connect the input and the output

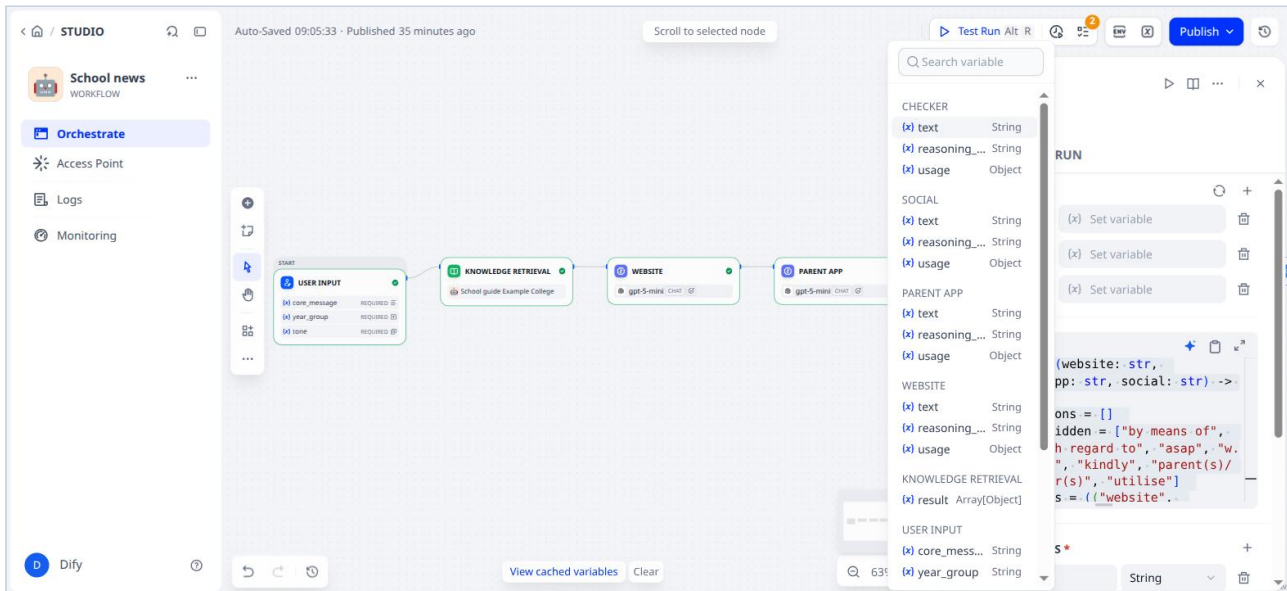
1. Under **INPUT VARIABLES** rename `arg1` to `website` and `arg2` to `parent_app`. Click + for a third variable and call it `social`. The names must be exactly the same as the names in the first line of the code.

The screenshot shows the Studio interface with a workflow on the left and a code editor on the right. The workflow consists of four nodes: **USER INPUT**, **KNOWLEDGE RETRIEVAL**, **WEBSITE**, and **PARENT APP**. The **USER INPUT** node has three input variables: `core_message`, `year_group`, and `tone`. The **KNOWLEDGE RETRIEVAL** node is connected to the **WEBSITE** node, which is connected to the **PARENT APP** node. The **WEBSITE** node has a `get-5-mins` action. The **PARENT APP** node has a `get-5-mins` action. The code editor on the right shows the Python code for the **PARENT APP** node. The code defines a `main` function that takes `website`, `parent_app`, and `social` as arguments. It initializes `reasons` as an empty list and `forbidden` as a list of words. It then checks if the `website` variable is in the `forbidden` list and if the `social` variable is not in the `forbidden` list. If both conditions are met, it returns `ok` as 1 and `reasons` as an empty list. Otherwise, it returns `ok` as 0 and `reasons` as a list containing the reason for failure.

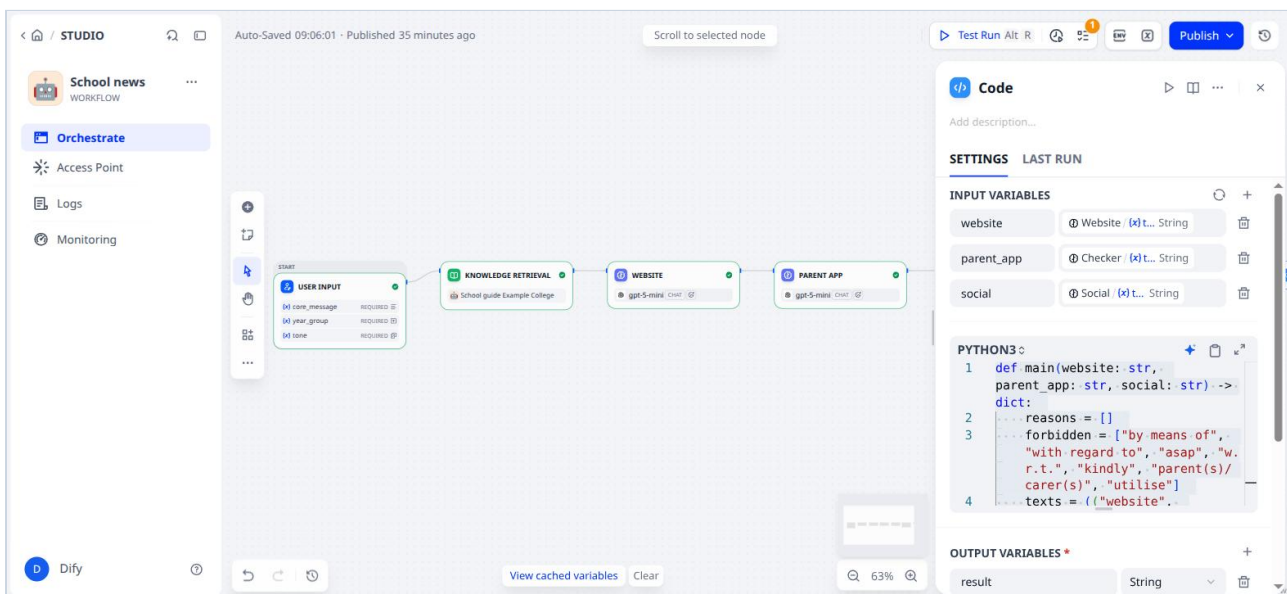
Input variables renamed, third one added

2. For each variable click **Set variable** and choose:

Variable	Choose
website	WEBSITE / text
parent_app	CHECKER / text (the improved version, not the raw one)
social	SOCIAL / text

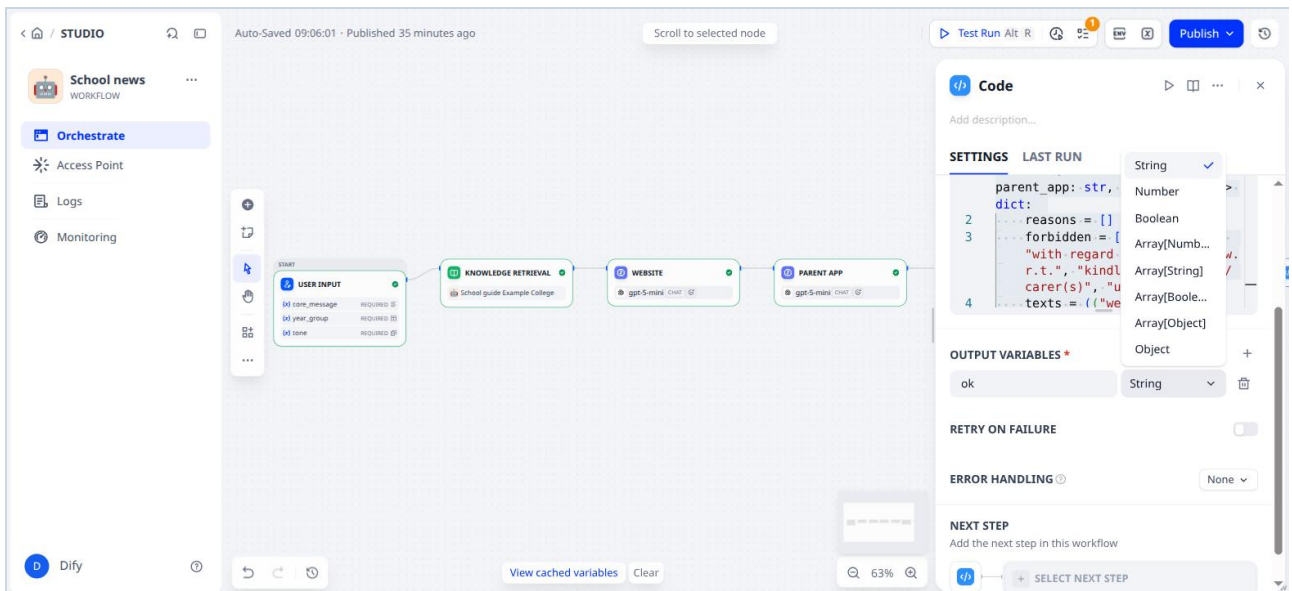


The variable picker



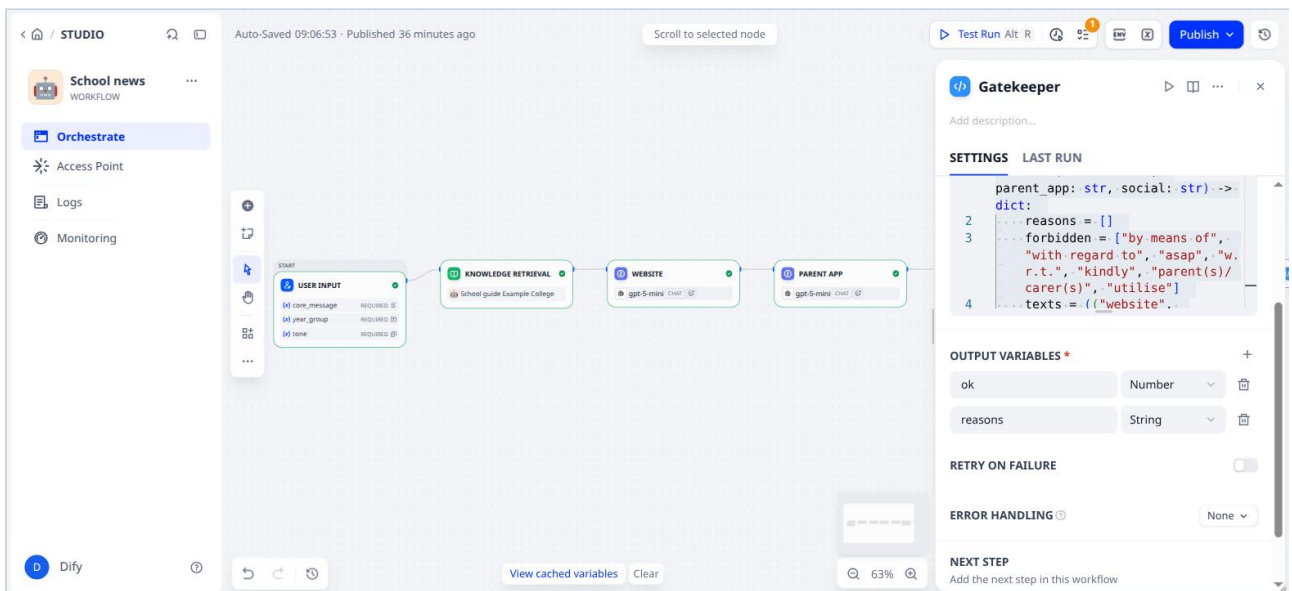
All three connected

- Under **OUTPUT VARIABLES**: rename `result` to `ok` and set the type to **Number**. Click **+** and add `reasons`, type **String**.



Choosing the type: Number

4. Rename the block to Gatekeeper .



Gatekeeper ready

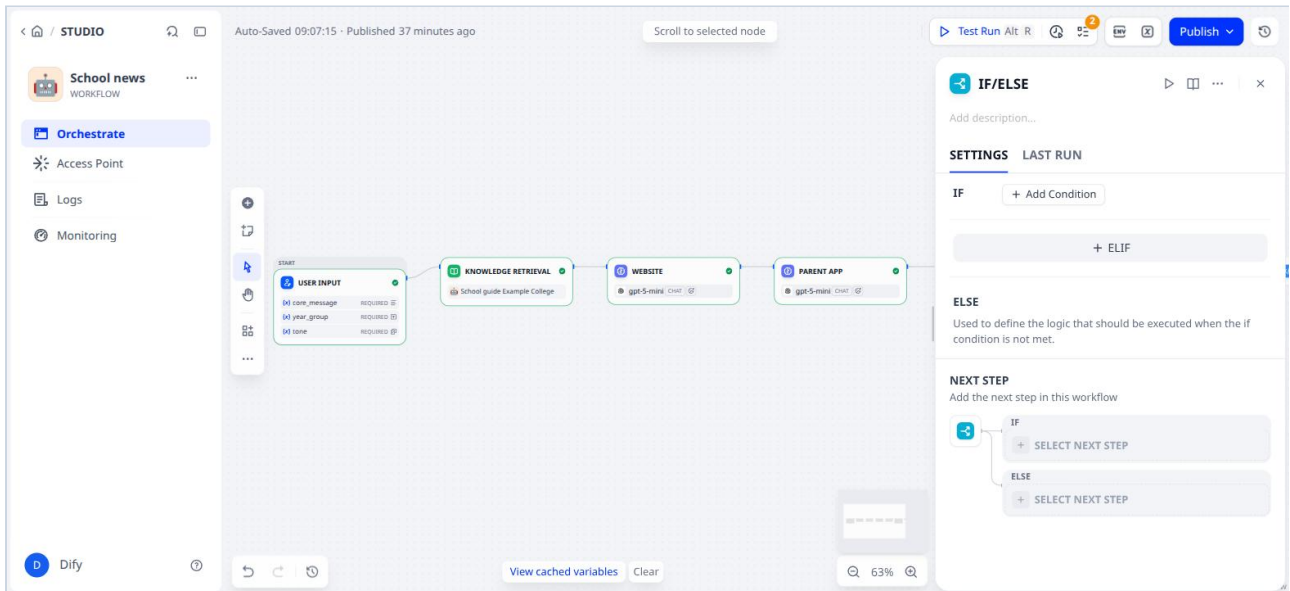
The output names (ok , reasons) must literally match the names in the return line of the code. One letter different and the workflow stops with an error about a missing output.

Your choice: the hard rules. Everything you can check with certainty belongs here and not in a prompt: length, forbidden words, required parts (hashtag, contact line, a date in the future), no personal email addresses, no phone numbers except reception's. Ask your communications officer: "Which mistake must never happen again?" That is your first rule.

Part C. The gate: IF/ELSE and two exits

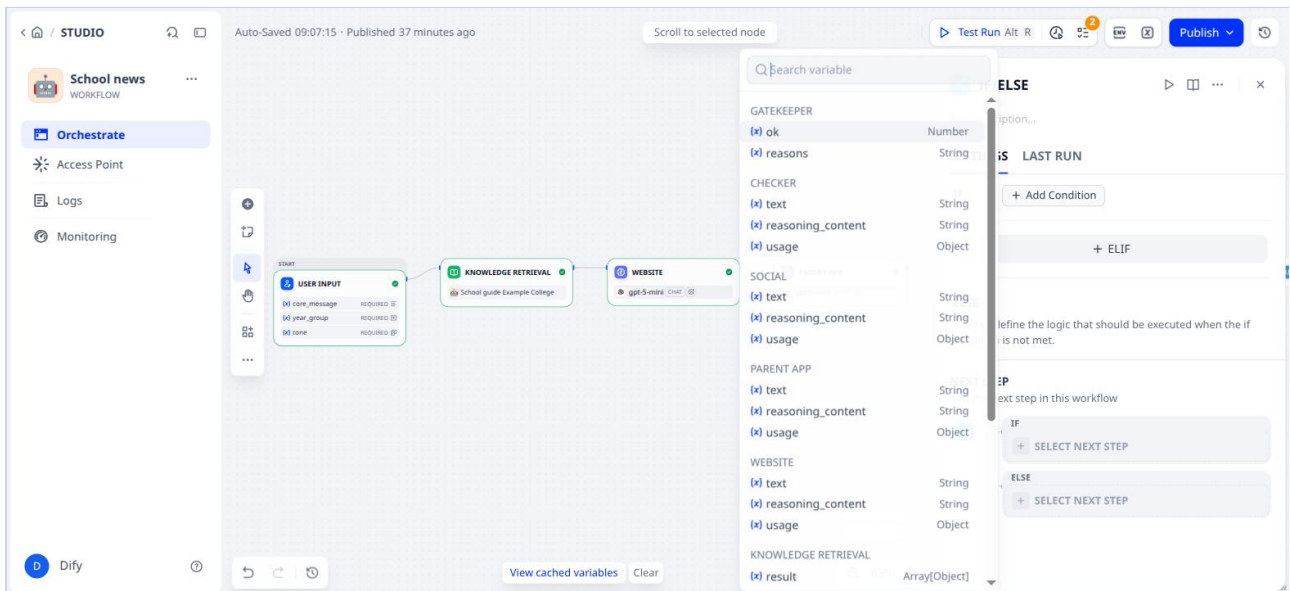
Step 5. IF/ELSE

1. At Gatekeeper: **Select next step** → under Logic: **IF/ELSE**.



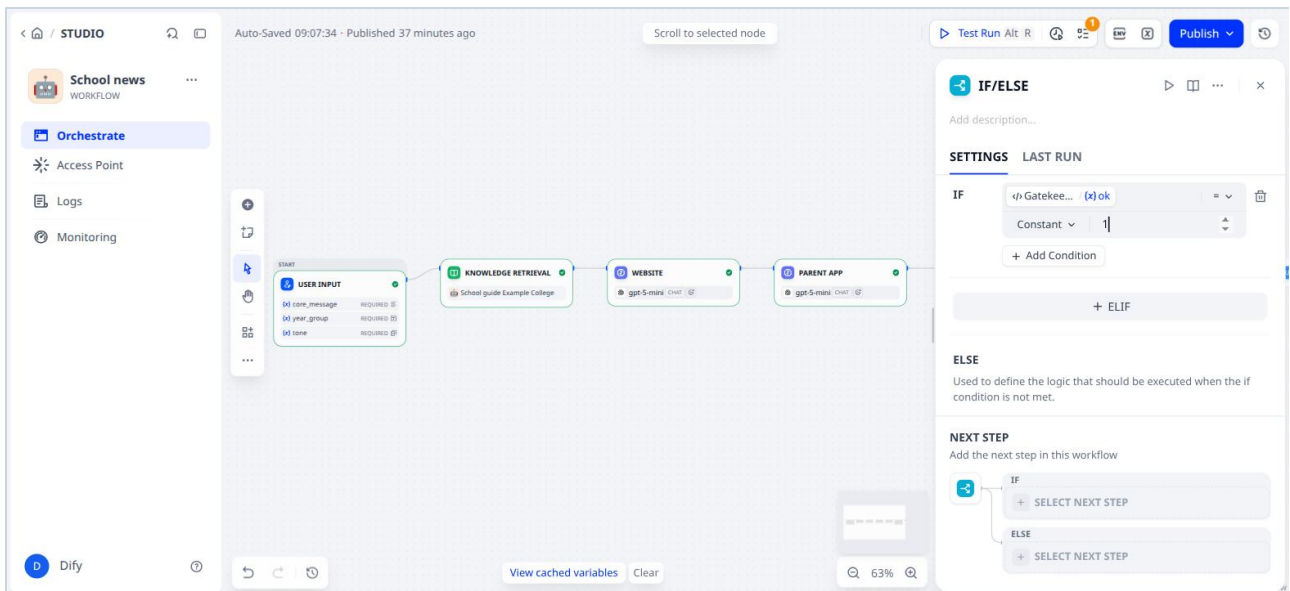
The empty IF/ELSE block

2. Under **IF** click **+ Add Condition** and choose **GATEKEEPER / ok**.



The variable for the condition

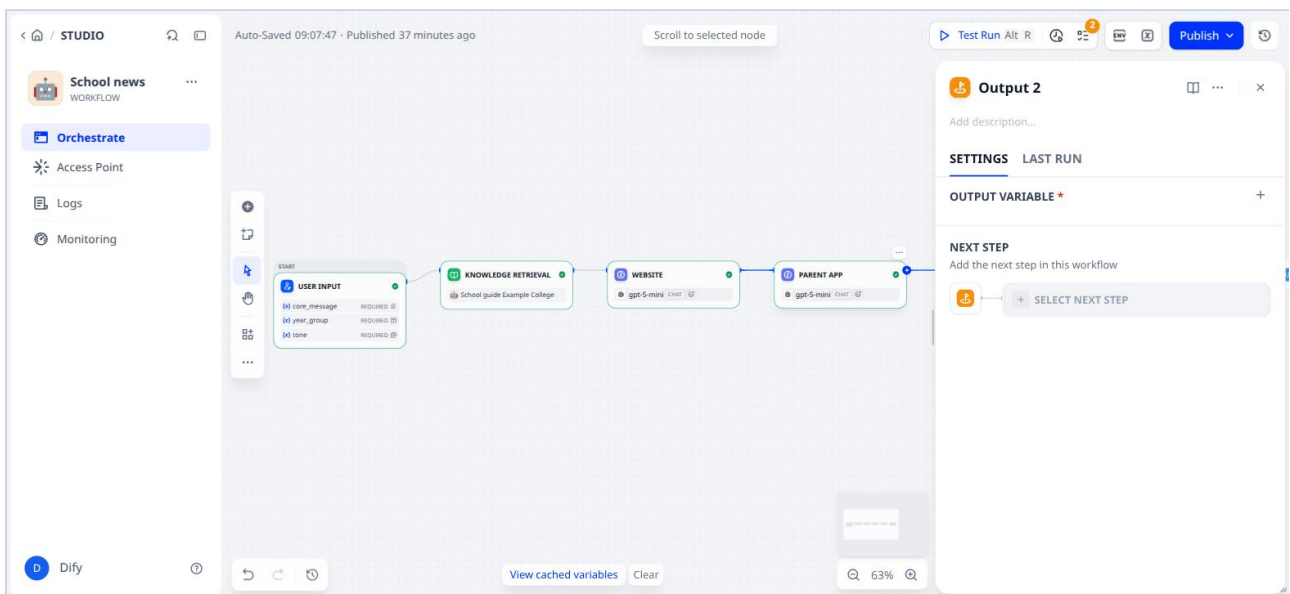
3. Leave the comparison on = and type **1** as the value.



The condition: ok = 1

Step 6. Exit 1: Approved

1. Under **NEXT STEP** under **IF**: Select next step → Output.

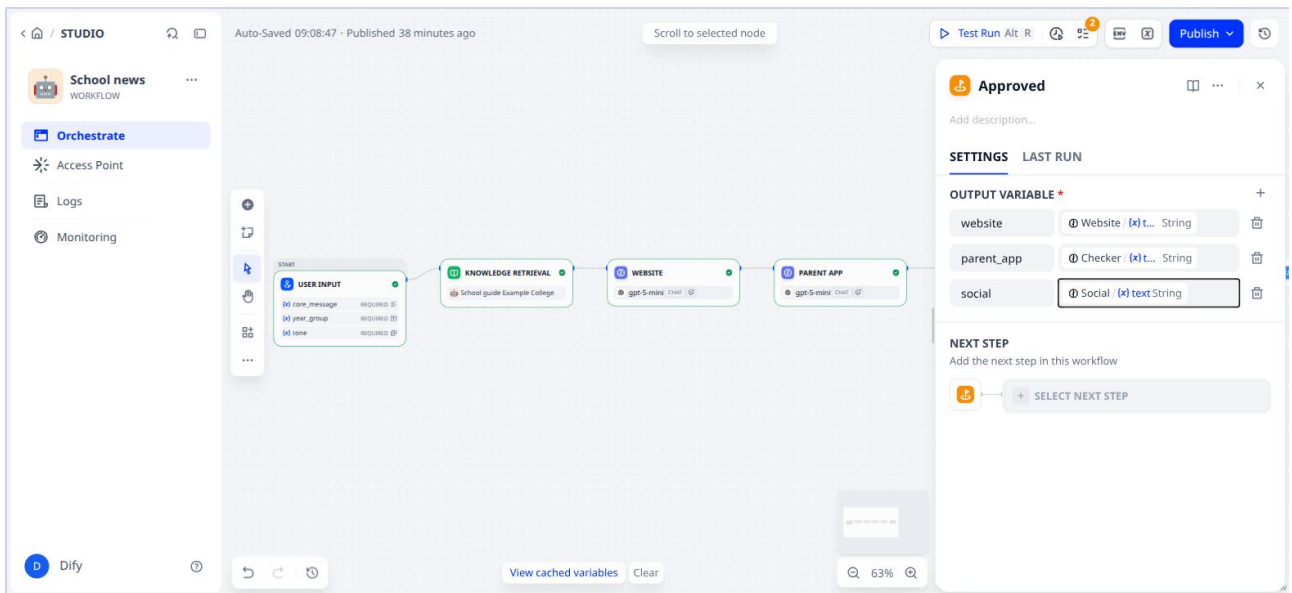


The new Output block

2. Add three output variables:

Variable name	Set variable
website	WEBSITE / text
parent_app	CHECKER / text
social	SOCIAL / text

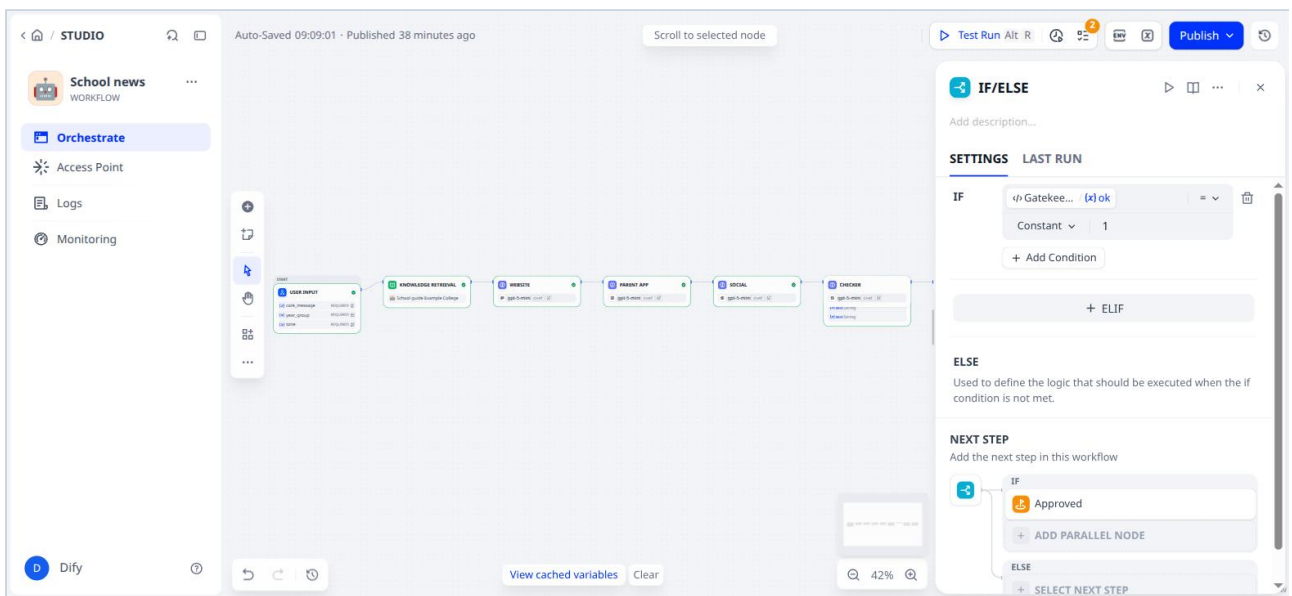
3. Rename the block to Approved .



Approved

Step 7. Exit 2: to the communications officer

1. Click the **IF/ELSE** block. Under **ELSE** it still says **Select next step**; choose **Output**.

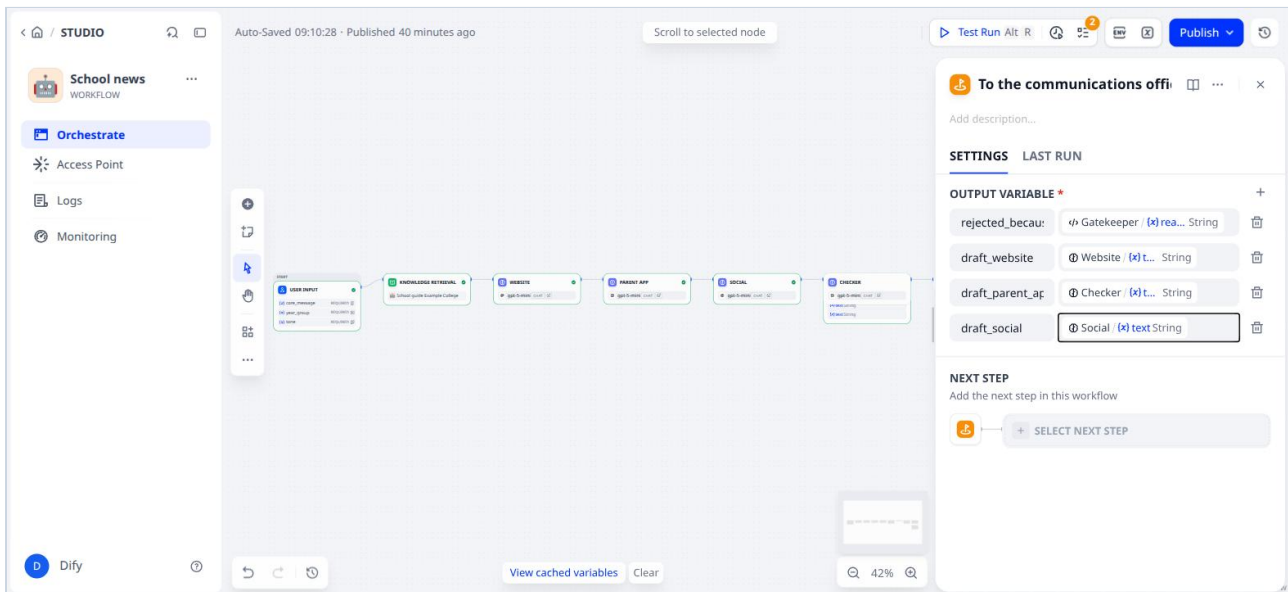


IF has Approved, ELSE nothing yet

2. Add four output variables. The reasons go at the top, so the officer immediately sees what is wrong.

Variable name	Set variable
rejected_because	GATEKEEPER / reasons
draft_website	WEBSITE / text
draft_parent_app	CHECKER / text
draft_social	SOCIAL / text

3. Rename the block to To the communications officer.

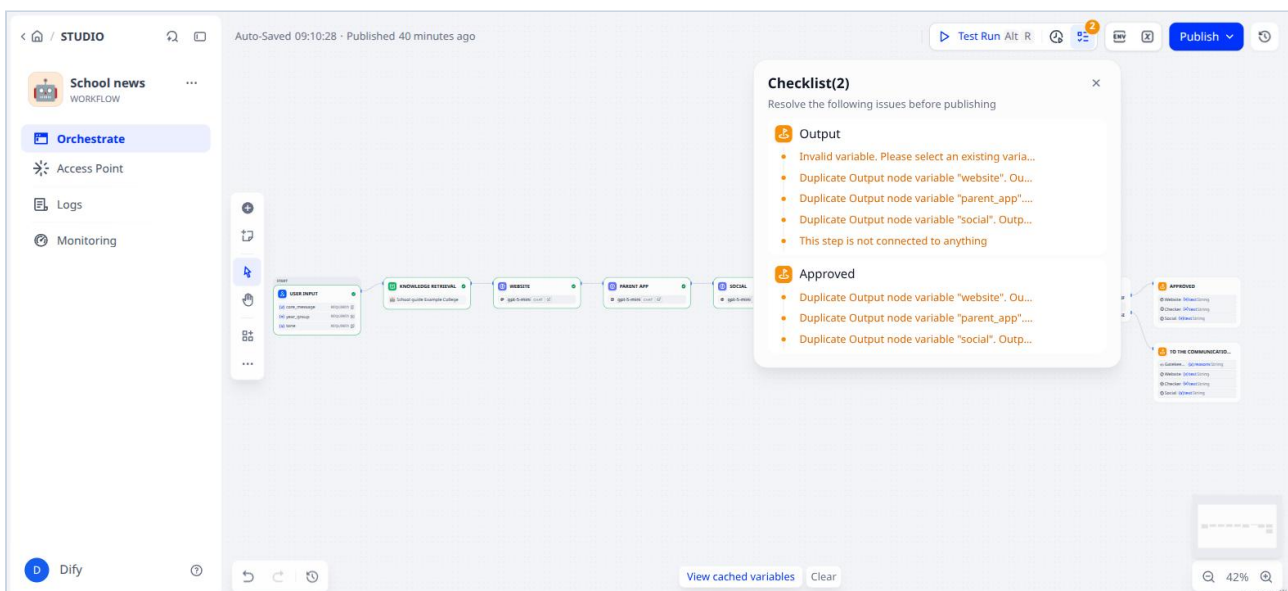


The second exit

Two Output blocks may not have variables with the same name. If you call them `website` in both, Dify puts an orange “Duplicate variable” in the checklist and you cannot publish. Hence `draft_` in the second exit.

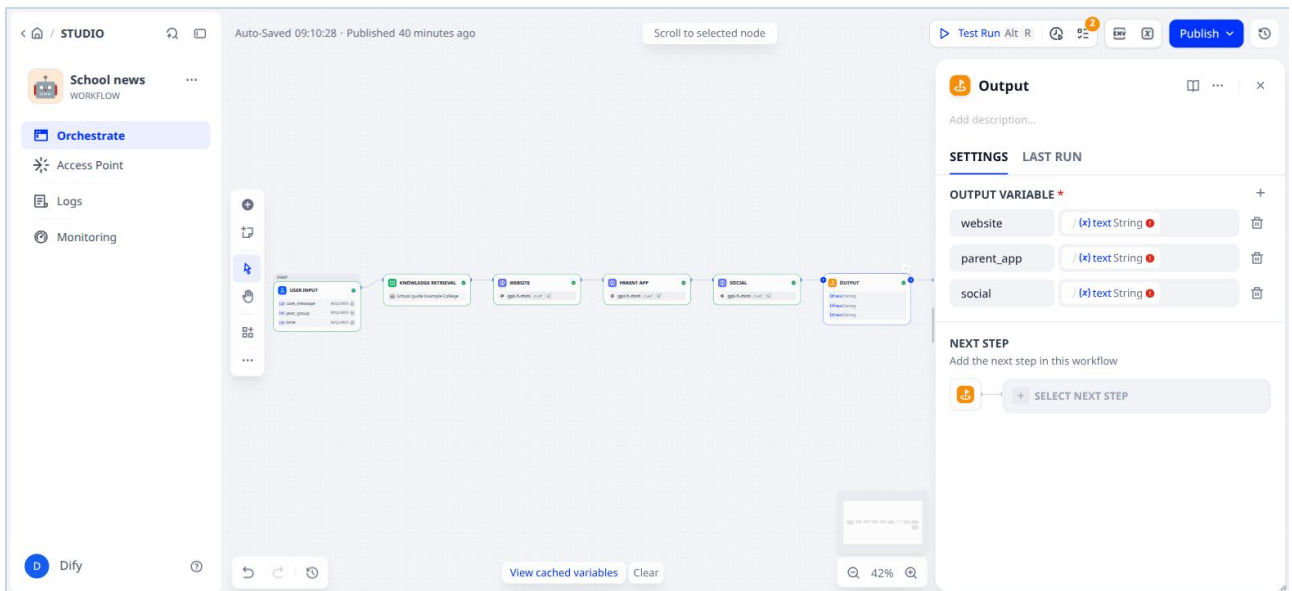
Step 8. Clean up the old Output

1. Click the checklist icon at the top right (next to Test Run). You see which blocks are not right yet.



The checklist: the old Output is connected to nothing

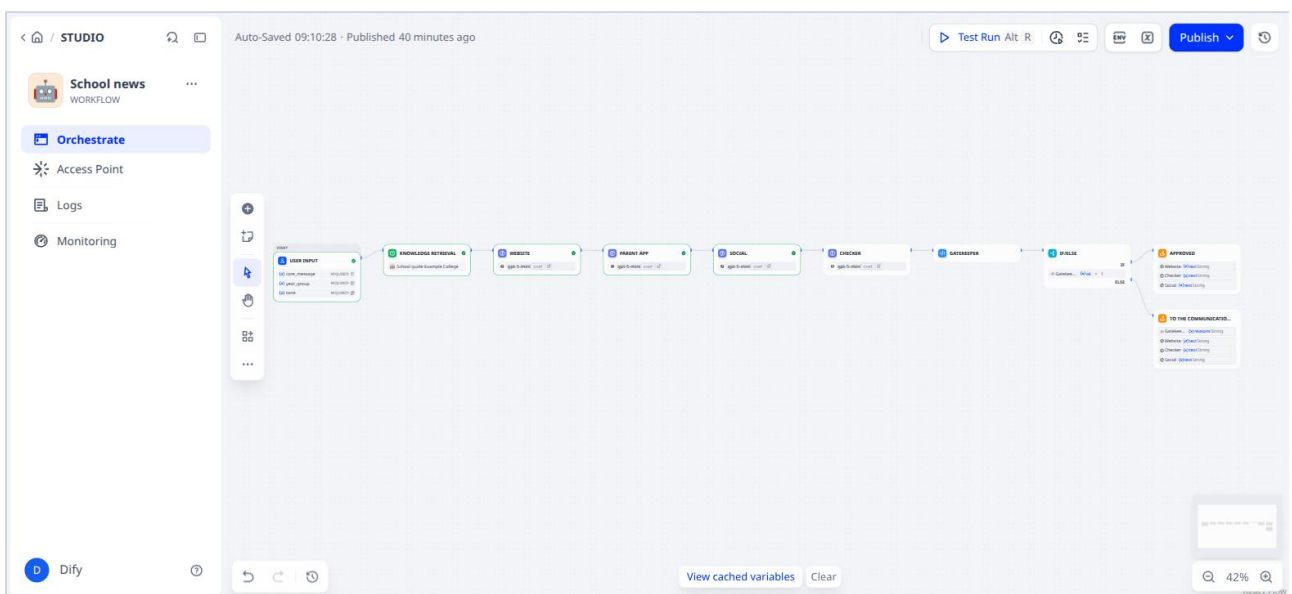
2. Click the first line under **Output** in the checklist. The old block is selected and the panel opens.



The old Output, selected from the checklist

3. In the panel click the three dots and then **Delete**.

4. Check that the checklist is empty (no number on the icon any more).



The new tail of the workflow

Part D. Testing: once right, once wrong

You always test a gatekeeper both ways. Testing only the good side proves nothing.

Step 9. Test 1: a complete message

1. **Test Run**. Fill in:

Field on screen	What you type
What has happened...	On Thursday 8 October at 19.30 there is a parents' evening for Year 8 in the main hall. Form tutors will talk about subject choices and the test week timetable. Parents can sign up until 5 October through the parent portal. Coffee and tea from 19.15.
For which class or year group?	Year 8
Which tone fits this message?	informative

2. **Start Run.** After about a minute the path via **IF** to **Approved** turns green.

The screenshot shows the Studio interface with a workflow titled 'School news' in the left sidebar. The main canvas displays a workflow with several steps: 'WEBHOOK', 'KNOWLEDGE RETRIEVAL', 'RETRIEVE', 'RECENT APP', 'SOCIAL', and 'DECISION'. A 'Test Run' window is open on the right, showing the 'RESULT' tab. The 'WEBSITE' section displays the generated text for the parents' evening, including the date, time, location, and sign-up information.

Test 1: the path runs via IF to Approved

3. Click **DETAIL**: status SUCCESS, run time and tokens.

This screenshot shows the 'DETAIL' tab of the 'Test Run' window. It displays the status as 'SUCCESS', the elapsed time as 57.386s, and the total tokens as 7457. The 'INPUT' and 'OUTPUT' sections show the JSON data used in the test run. The 'OUTPUT' section shows the generated text for the parents' evening, which matches the text in the previous screenshot.

Detail of the run

4. Click **TRACING** and expand **GATEKEEPER**. Under OUTPUT is what the code returned.

The screenshot shows the Dify Studio interface. On the left, a sidebar contains 'School news WORKFLOW', 'Orchestrate', 'Access Point', 'Logs', and 'Monitoring'. The main canvas displays a workflow with blocks: USER INPUT, KNOWLEDGE RETRIEVAL, WEBSITE, PARENT APP, SOCIAL, and CHECKER. On the right, a 'Test Run (09:13:59)' panel is open, showing a table of execution results for each block.

Block	Time	Status
USER INPUT	0.078 ms	✓
KNOWLEDGE RETRIEVAL	3.583 s	✓
WEBSITE	1.828K tokens - 15.756 s	✓
PARENT APP	2.082K tokens - 16.292 s	✓
SOCIAL	1.438K tokens - 8.132 s	✓
CHECKER	2.109K tokens - 12.851 s	✓
GATEKEEPER	94.849 ms	✓
IF/ELSE	0.126 ms	✓
APPROVED	0.091 ms	✓

Tracing: every block with its time and tokens

This screenshot shows the same workflow as the previous image, but the 'Test Run' panel is expanded to show the 'OUTPUT' of the 'GATEKEEPER' block. The output is a JSON object indicating that all rules are satisfied.

```
1 {
2   "ok": 1,
3   "reasons": "All rules satisfied."
4 }
```

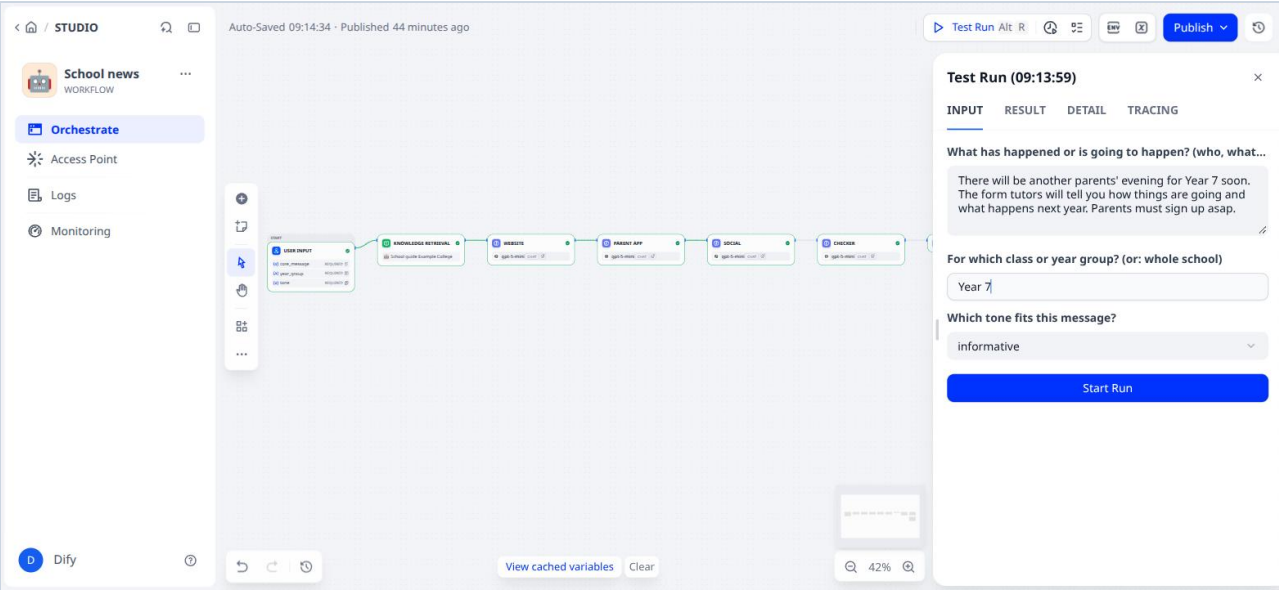
The Gatekeeper: ok 1, "All rules satisfied."

Tracing is your friend. For every question “why did this come out?” you open the block here and see exactly what went in and what came out. This is also the screen on which you saw that the Checker had removed the sign-up deadline (see the orange box in step 2).

Step 10. Test 2: a message with gaps

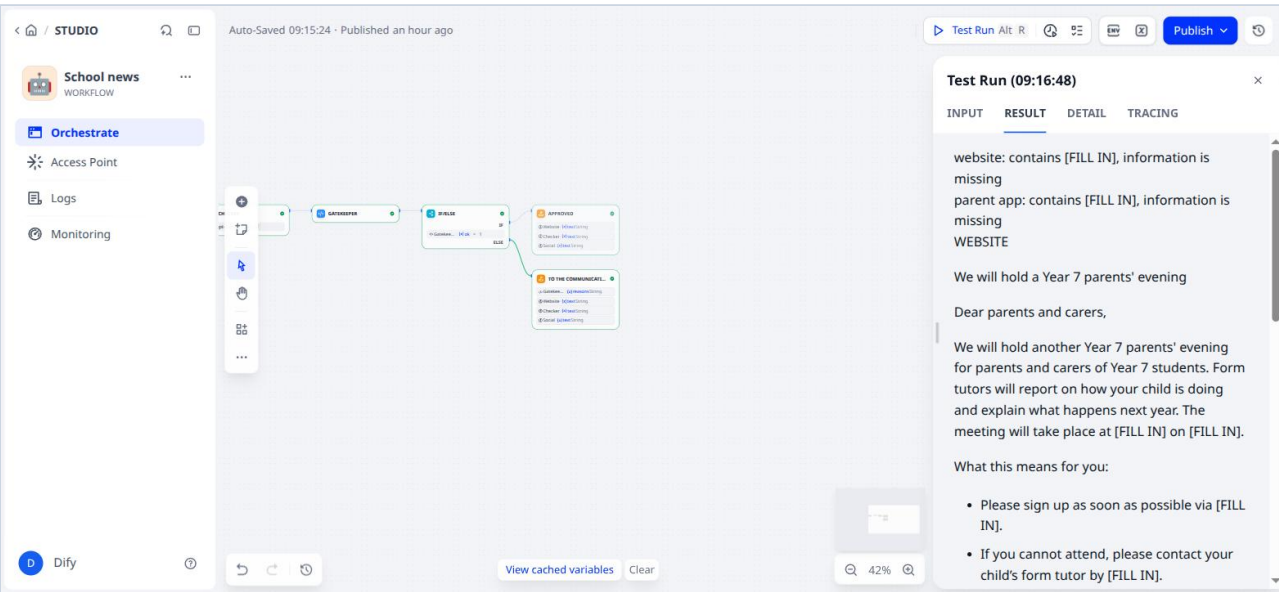
1. **Test Run** again, now with a message without a date, without a time and with an abbreviation:

Field on screen	What you type
What has happened...	There will be another parents' evening for Year 7 soon. The form tutors will tell you how things are going and what happens next year. Parents must sign up asap.
For which class or year group?	Year 7
Which tone fits this message?	informative



Test 2: the input

2. **Start Run.** Now the path runs via **ELSE** to **To the communications officer**, and the reasons are at the top of the result.

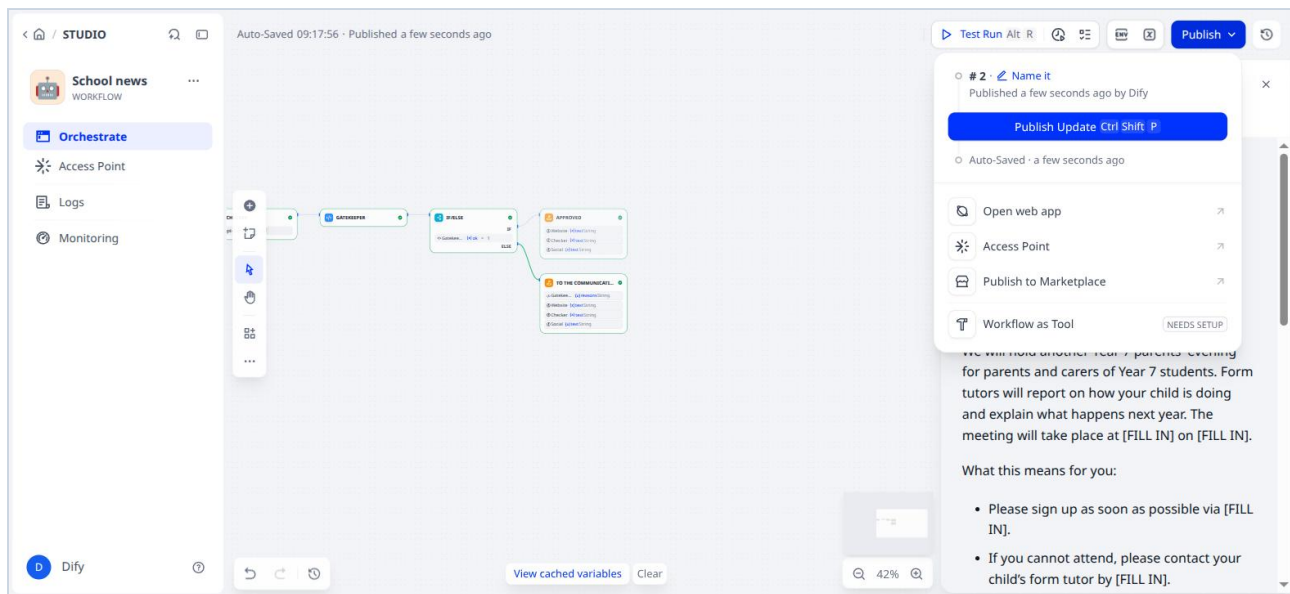


Test 2: rejected, with the reasons at the top

Read the reasons: all three texts contain [FILL IN] (the models did not know the date and neatly did not invent one). In our Dutch run the social post also contained the abbreviation from the core message; in this run the model left it out. That difference is instructive: the Social prompt asked to follow the style guide, and whether the abbreviation slips through depends on the run. A prompt asks; the gatekeeper enforces, every time.

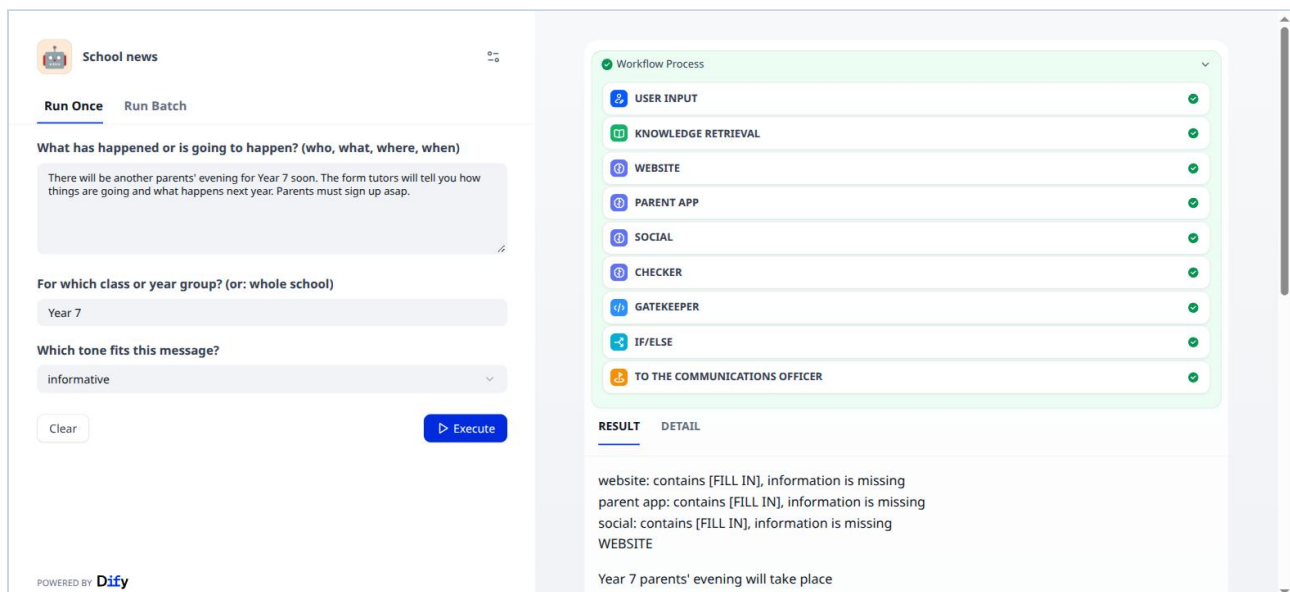
Step 11. Publish

1. **Publish → Publish Update.** Dify creates version 2 of the app; with **Name** you can give a version a name, and via the clock icon at the top right you can always go back to version 1.



Published as version 2

2. Open the webapp and repeat test 2. The colleague sees the reasons at the top and below them the drafts to complete.



The webapp: rejected, with the reasons and the drafts

The link has stayed the same: <https://udify.app/workflow/E2CHtVPKe3PLg7GF>.

Checking that it worked

- The checklist at the top right shows no number
 - Test 1 ends in **Approved**, test 2 in **To the communications officer**
 - In **TRACING** Gatekeeper gives `ok: 1` for test 1 and `ok: 0` with at least one reason for test 2
 - The parent app message in Approved comes from Checker (visible in the Output settings), not from Parent app
 - The webapp shows the reasons at the top for a rejected message
-

When things go wrong

What you see	What is going on
Code block: error about <code>IndentationError</code> or <code>SyntaxError</code>	The indentation is damaged. Select everything, delete, paste again
Code block: error about a missing output variable	The names under OUTPUT VARIABLES differ from <code>ok</code> and <code>reasons</code> in the <code>return</code> line
Code block: <code>main()</code> missing argument	An input variable is named differently from the first line of the code, or is not connected
IF/ELSE always goes to ELSE	The type of <code>ok</code> is String instead of Number, or you compare with <code>"1"</code> instead of <code>1</code>
Checklist: "Duplicate variable of the output node"	Two Output blocks use the same variable name. Rename one (step 7)
Checklist: "This step is not connected to anything"	The old Output block still exists. Delete it (step 8)
The Checker removes information or changes digits	The prompt does not limit what may change in content. See the two sentences in step 2
Approved shows the raw parent app message	The variable <code>parent_app</code> in Approved points to Parent app/text instead of Checker/text
The run takes more than two minutes	Four LLM blocks in a row; 60 to 90 seconds is normal. If it takes longer, look in Tracing which block hangs

Housekeeping and costs

- **One run** now costs four model calls (Website, Parent app, Social, Checker) and about 7,500 tokens; with your own OpenAI key about one and a half cents, on Dify credits about

13. The Code block and IF/ELSE cost nothing. A sandbox account (200 credits) manages about fifteen runs; a Professional account (5,000 credits a month) is ample for a team.

- **The forbidden list** lives in two places: in the style guide (for the models) and in the code (for the gate). If the style guide changes, change the code too. Forget that and you get rejections the models do not understand.
- **Versions.** Every Publish Update is a new version. If a change goes wrong, you restore the previous version via the clock icon at the top right without rebuilding.
- **Logs** also show which messages were rejected and why. That is exactly the information you need to improve the prompts of Website, Parent app and Social: every reason that occurs often is a sentence missing from a prompt.

Make it yours

From easy to hard.

Change	Where	Difficulty	What you get for it
Other forbidden words	Code, line <code>forbidden</code>	easy	The real list from your style guide is enforced
Other word limits	Code, line <code>texts</code>	easy	Limits that fit your channels
An extra check point for the Checker	Step 2, first paragraph of the prompt	easy	More checking work off your plate
A Checker for the website piece	Duplicate the block, different input and rules	medium	The longest text checked too
A rule “the date must be in the future”	Code, with <code>datetime</code>	medium	No more messages about yesterday
A third exit: “doubtful” for minor violations	A second condition in IF/ELSE (ELIF) and a second counter in the code	medium	Small things through automatically, big ones to a person
A human in the loop: post the message only after a click	Human Input block before Approved	hard	The most honest gatekeeper there is
Self-critique in a loop until the gate opens (at most two rounds)	Loop block around Checker and Gatekeeper	hard	Fewer rejections, but double the cost

Three assignments

1. **Fill the list.** Ask your communications team for the five words or phrases they cross out most often. Put them in `forbidden` and run test 1 and test 2 again. Is test 1 now rejected

too? Then your list is too broad (a word like “just” appears everywhere) and you need to be more precise.

2. **Break the gate.** Write a core message that passes the gate but is still wrong: a wrong date, an invented student name, a tone that does not fit. What does the Checker catch, what does the code catch, and what does nobody catch? For that last one you write a new rule, or you decide that a person keeps doing that.
3. **Measure it.** Run ten real messages from the past month through the app. Count: how many went straight through, how many were rejected, and how many of the rejections were justified? Those three numbers are your conversation with the leadership about whether this is worth it.

Checklist before you use your own material

- [] The forbidden list and the limits in the code match the style guide in the knowledge base
- [] You have tested both sides: a message that must pass the gate, and one that must be stopped
- [] It is agreed who picks up the rejected messages and within what time
- [] The core messages colleagues enter contain no student names; the gate does not check for that
- [] Someone also spot-checks approved messages; the gate checks form, not truth
- [] You still have the previous version of the app (version history), in case the new one breaks something

Licence: CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>). You may copy, adapt and reuse this document, including at your own school, provided you credit the source (Guido van Dijk, LeX Consultancy B.V., <https://git.lexconsultancy.nl/guido/dify-guides>) and share your adaptation under the same licence. The LeX Consultancy logo is excluded; product names are trademarks of their owners. Schools, people and documents in the examples are fictional.

Owner and contact: Guido van Dijk, LeX Consultancy B.V. · guido@l-e-x.nl

