Blog

From Clutter to Clarity: A Guide to Build Triple-branch Workflow

---

*Overview: Learn how to tailor an automated information processing system for your daily work. This guide demonstrates how to use the **Question Classifier** for intelligent pipeline and leverage powerful Agents to handle diverse formats like documents, audio, and images.*

In any modern workplace, we are constantly managing three primary types of information: documents, audio, and images.

"I have over three meetings a day and five reports to check. On top of that, colleagues are sending me pictures and screenshots from morning until night." Do you have the same trouble at your working hours? This daily struggle with "Information Overload" constantly disrupts our focus and cripples efficiency.

So, instead of passively processing information, why not proactively build a system to automate it? -- Try to build a '**Triple-branch Workflow**' with Dify, designed for intelligently classify and process these multi-format inputs automatically.

Part 1: The Architecture: A Workflow with a "Smart-Sorting" Hub

To handle these different information types, I designed the workflow around a central "smart-sorting" hub---the Question Classifier node. The function of **Question Classifier** is to automatically identifies the type of information a user provides and routes it to the correct "assembly line" for processing.

Structural Overview

As we have an ultimate goal to pursue, the first thing to do is to determine the structure of this Workflow:

- **From The START node to Question Classifier:** The Start node allows a user to input information in various formats. The Question Classifier then analyses this input and directs it to the appropriate branch.

- **Branch 1: The Audio Processing Pipeline (Speech-to-Text):** Designed specifically to handle audio inputs like meeting recordings and voice memos.
- **Branch 2: The Document/Text Processing Pipeline:** The core engine for analysing reports, extracting summaries, and crunching data.
- **Branch 3: The Image Processing Pipeline:** Used to recognise and organise content from images, such as text and key elements.

The Ultimate Goal: Regardless of the input format, the objective is to produce structured, easy-to-read text outputs (like meeting minutes, data reports, or image-content summaries), to drastically boost efficiency and streamline information management.

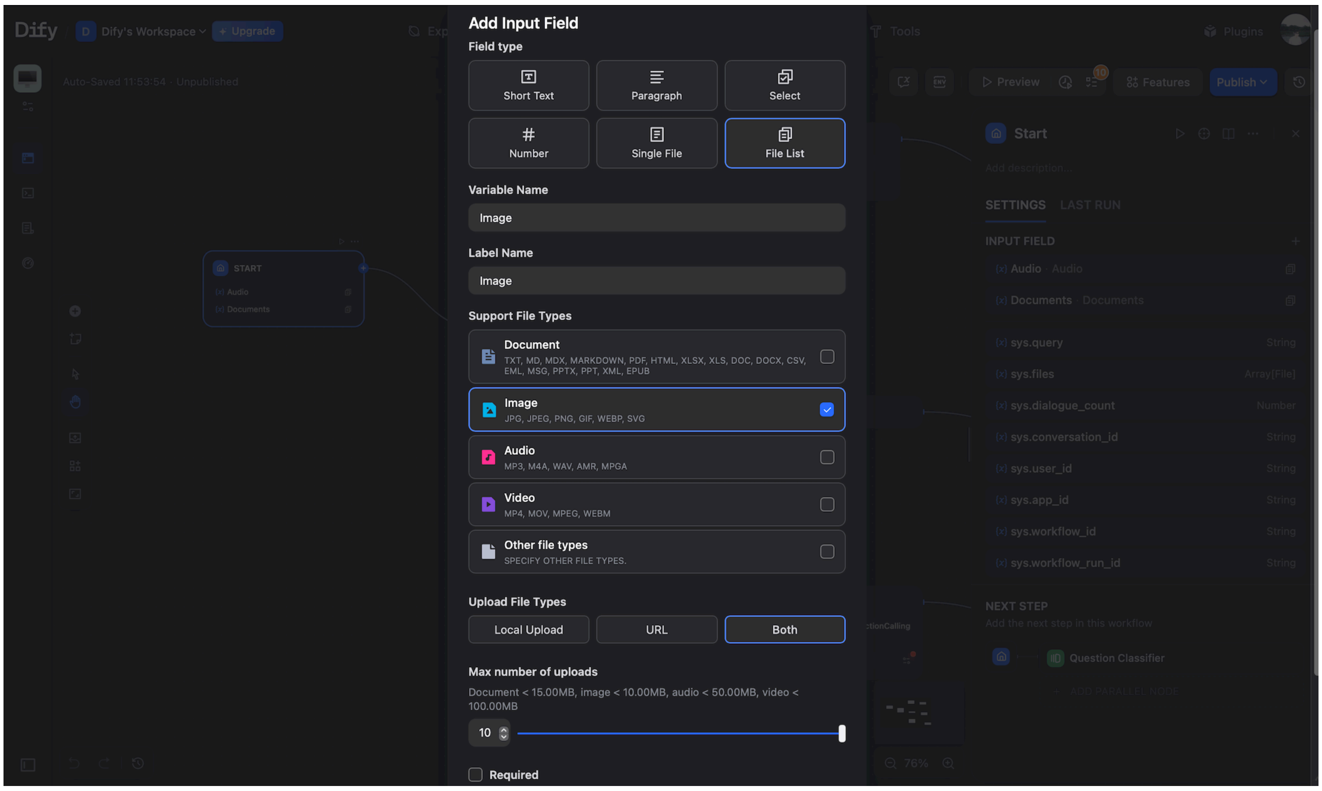
Part 2: The Building Journey — From Idea to Reality

2.1 Mastering the Core Components: Nodes & Tools

To construct this **Triple-branch Workflow**, it's essential to understand its fundamental building blocks: the **Nodes**. The official Dify documentation is an indispensable resource in this process, providing clear explanations for editing both the nodes and the built-in tools.

Furthermore, the Dify Marketplace is a great place to discover and integrate free, third-party plugins. My initial step was to identify the crucial nodes required to process the different material types: the **Question Classifier**, an **Audio processor (Speech to text), a** Document Extractor, and an Agent node.

After familiarising with the function of each necessary node, we proceeded with the construction of the prototype. I followed the left-to-right construction sequence, starting with the **START** node. Here, I configured the initial input field to accept file uploads, as shown below:


Add Input Field

Field type: Short Text | Paragraph | Select | Number | Single File | [✓ File List]

Variable Name: Image

Label Name: Image

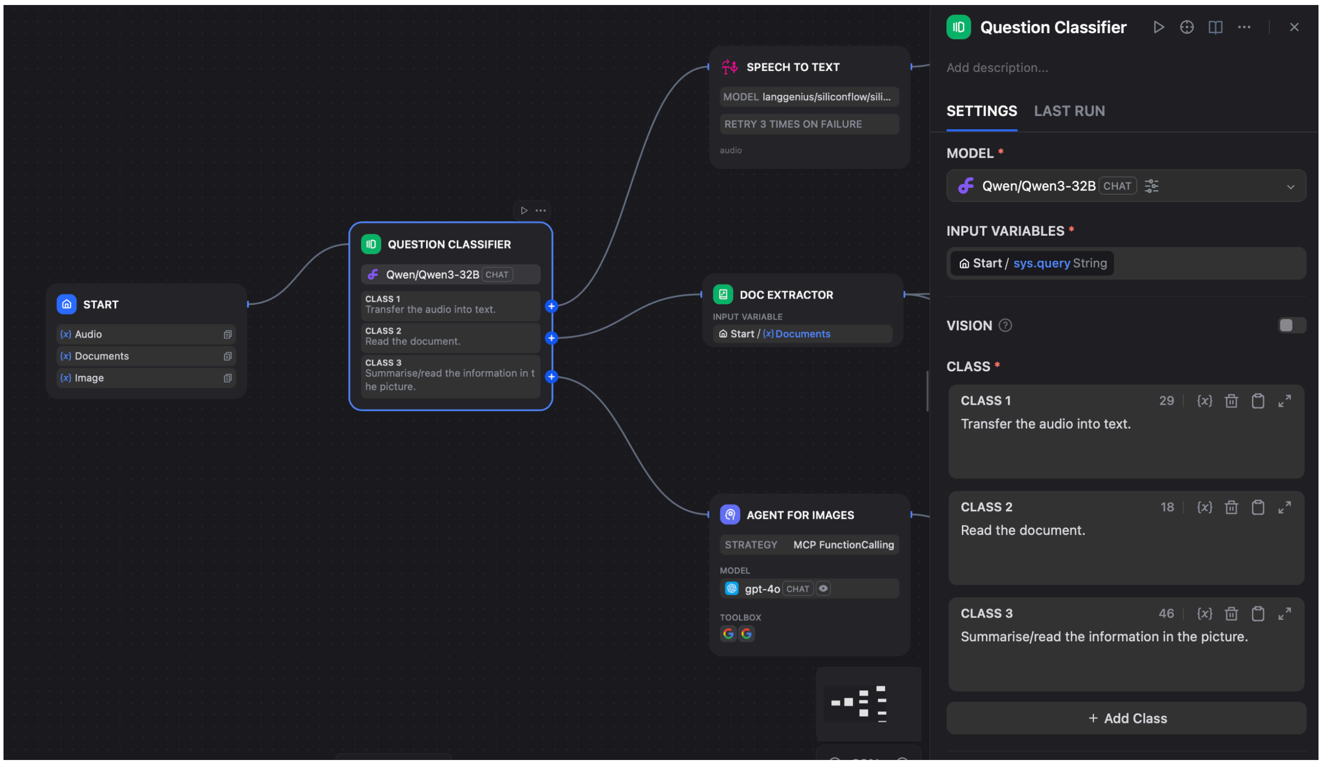
Support File Types
- Document — TXT, MD, MDX, MARKDOWN, PDF, HTML, XLSX, XLS, DOC, DOCX, CSV, EML, MSG, PPTX, PPT, XML, EPUB ☐
- Image — JPG, JPEG, PNG, GIF, WEBP, SVG ☑
- Audio — MP3, M4A, WAV, AMR, MPGA ☐
- Video — MP4, MOV, MPEG, WEBM ☐
- Other file types — SPECIFY OTHER FILE TYPES. ☐

Upload File Types: Local Upload | URL | [✓ Both]

Max number of uploads
Document < 15.00MB, image < 10.00MB, audio < 50.00MB, video < 100.00MB
10

☐ Required

The next node is **Question Classifier**. After check the Doc, I selected an LLM and wrote a simple prompt to guide its classification process:



CLASS 1
Transfer the audio into text.

CLASS 2
Read the document.

CLASS 3
Summarise/read the information in t he picture.

MODEL langgenius/siliconflow/sili...
RETRY 3 TIMES ON FAILURE

INPUT VARIABLE
Start / Documents

STRATEGY MCP FunctionCalling
MODEL
gpt-4o CHAT

Question Classifier

MODEL *
Qwen/Qwen3-32B CHAT

INPUT VARIABLES *
Start / sys.query String

VISION

CLASS *

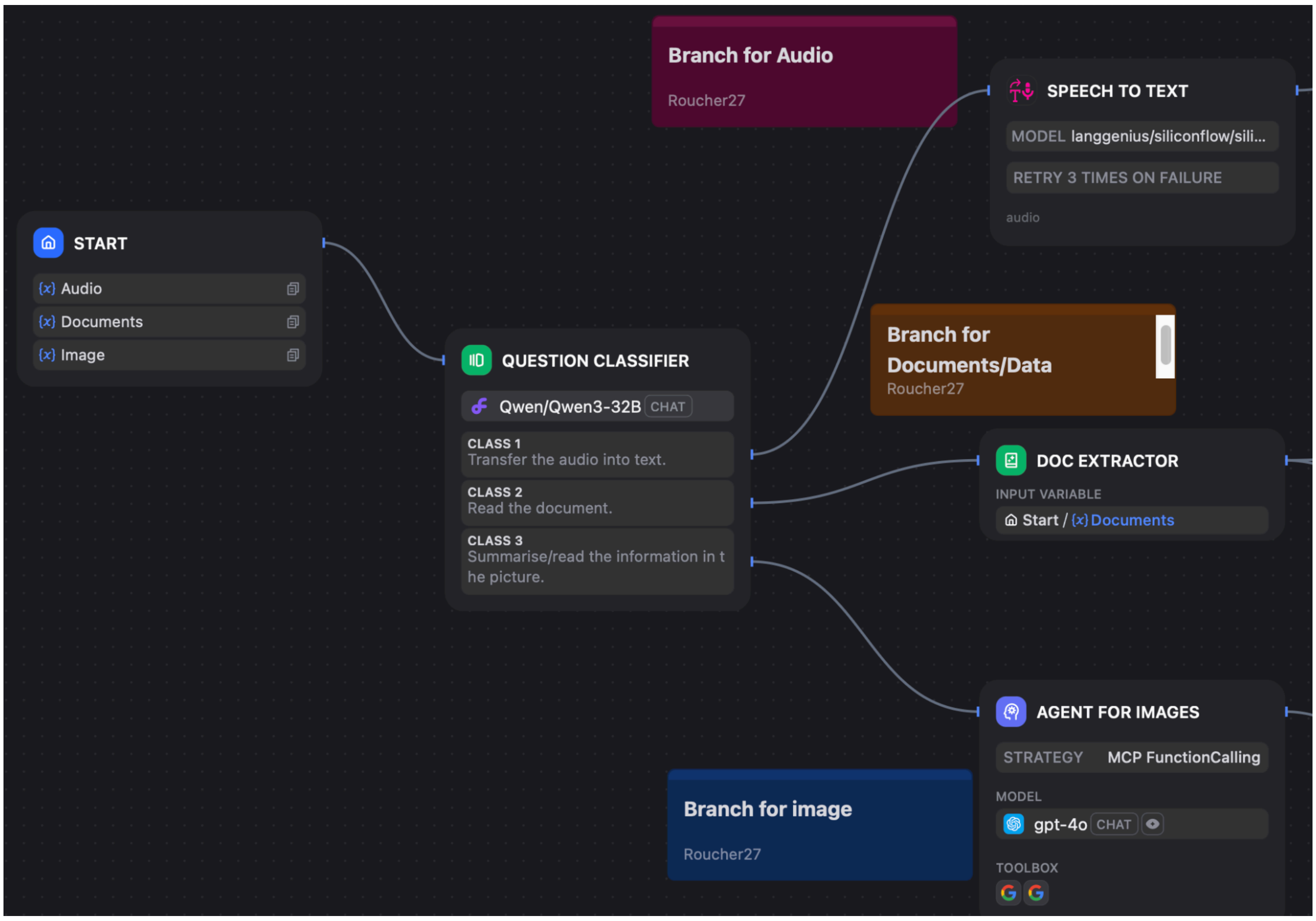
CLASS 1 29
Transfer the audio into text.

CLASS 2 18
Read the document.

CLASS 3 46
Summarise/read the information in the picture.

Following the **Question Classifier**, I established three parallel branches to perform the initial processing for each of the three input types.

The corresponding nodes for these branches—for audio, documents, and images—are readily available from the main '**Add Node**' menu. I positioned these three nodes and connected them to the respective outputs of the **Question Classifier**, resulting in the structure shown in the image below:



MODEL langgenius/siliconflow/sili...
RETRY 3 TIMES ON FAILURE
audio

{x} Audio
{x} Documents
{x} Image

Qwen/Qwen3-32B CHAT
CLASS 1
Transfer the audio into text.
CLASS 2
Read the document.
CLASS 3
Summarise/read the information in t he picture.

INPUT VARIABLE
Start / {x} Documents

STRATEGY MCP FunctionCalling
MODEL
gpt-4o CHAT

The next step involves configuring each of these three nodes to handle their specific tasks. With this, the foundational architecture of the workflow is complete. The task now shifts from high-level design to developing the specific processing logic within each individual branch.

2.2 The Role of LLM & Agent

The Agent serves as a crucial bridge between the LLM and real-world tasks. It became a primary component in the Triple-branch workflow because it not only handles multi-modal information and various input formats but can also invoke different tools and generate files—capabilities that standard LLM nodes lack.

However, Agents primarily fall into two categories: Function Calling and ReAct. Understanding their differences is key to choosing the right one for the job.

FunctionCalling

With **FunctionCalling**, the model comprehends the user's intent and directly determines which **Tool** to use. It then outputs a structured command, typically in JSON format, to invoke that tool. The entire process is more of a single, decisive action.

Core Characteristics:

- **Direct and Efficient:** Ideal for clear, single-step problems. Its efficiency comes from requiring fewer LLM calls, resulting in faster execution.

- **Structured Output:** The output is a precise function call format, which ensures high reliability and predictable results.
- **Opaque Reasoning:** The model's decision-making process is not visible. You don't see how it decided to call a specific function, it acts more like a black box.
- **Best Suited For:** Simple tasks that map directly to a single, specialized tool.

ReAct (Reason + Act)

ReAct is a more powerful reasoning framework that emulates human problem-solving: **Thought** -> **Action** -> **Observation**. The Agent engages in a series of these "thought-action-observation" cycles until it determines it has gathered enough information to provide a final answer.

Core Characteristics:

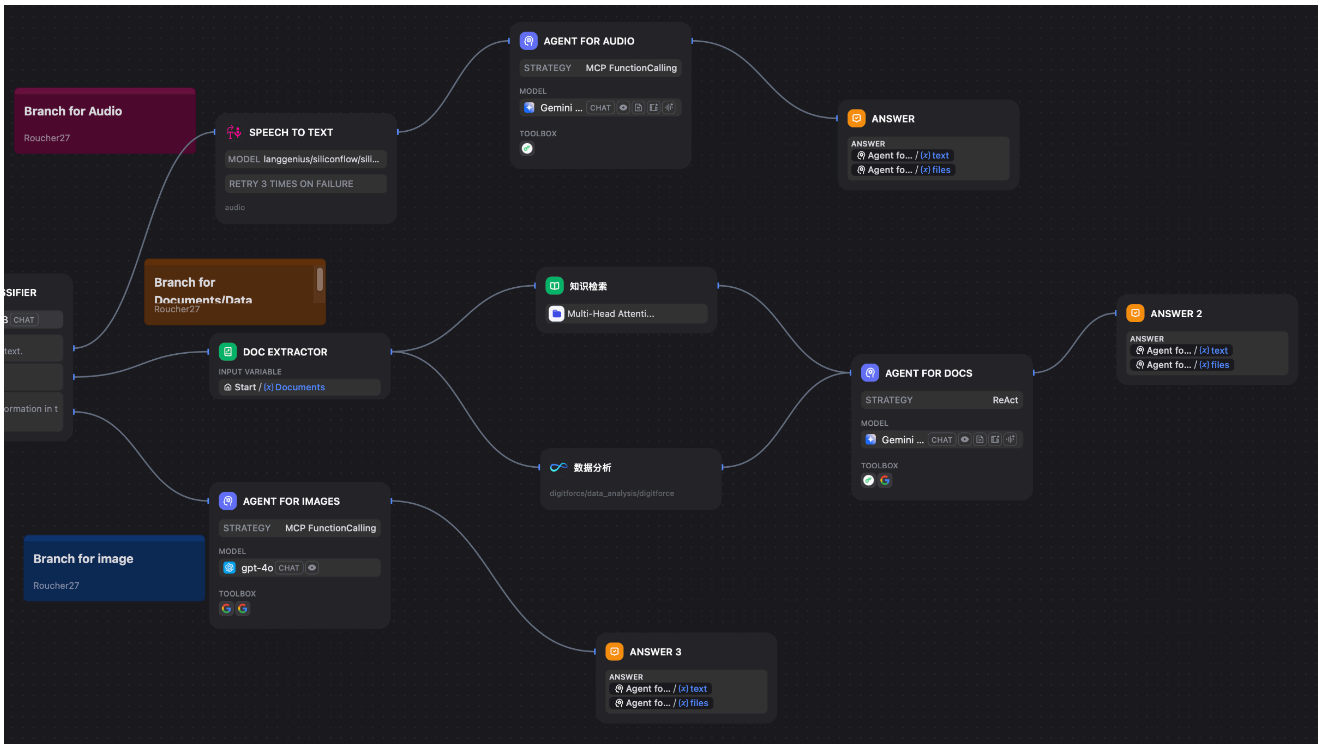
- **Powerful and Flexible:** Capable of handling complex problems that require multiple steps or a combination of tools to solve.
- **Transparent Reasoning:** Its greatest advantage is the visibility of the Agent's **step-by-step "thought"** process. This transparency is crucial for debugging and understanding the Agent's behavior.
- **Robust and Adaptive:** If a **tool** fails or an action does not yield the expected result, the Agent can self-correct and adjust its plan in the next thought cycle.
- **Potentially Higher Overhead:** Due to its iterative nature involving multiple LLM calls, it can be slower and consume more tokens.

Initially, I attempted to use **FunctionCalling** Agents for all branches. However, I soon realised that tasks involving audio and images are typically well-defined and single-purposed, such as "Identify objects in this image" or "Transcribe this audio file." The model doesn't need complex reasoning, it just needs to accurately identify the intent and invoke the appropriate specialized tool.

However, text-based tasks are often more complex, open-ended, and require logical reasoning. For example, "Summarise this report and compare it with last month's data," or "Analyse the sentiment of these user comments and identify the top three complaints." This is where ReAct shines. Its iterative reasoning and self-correction capabilities are perfectly suited to ensure high-quality outcomes for complex, multi-step textual tasks. Therefore, I switched the **Agent for Doc** to **ReAct** mode to handle this complexity.

Furthermore, the choice of LLM is critical to the workflow's final performance. I recommend users gain a basic understanding of different models' capabilities, such as their proficiency in long-context processing, multi-modal handling, image recognition, or reasoning. This knowledge significantly improves efficiency of Workflow building and guarantees a higher quality output. Alternatively, one can continuously test and swap different LLM models during the tuning phase to find the optimal fit, the choice is entirely up to the user's specific needs.

During this process, I also leveraged AI tools to help generate and refine the prompts for each Agent.
Below are examples of the final three configured agents :



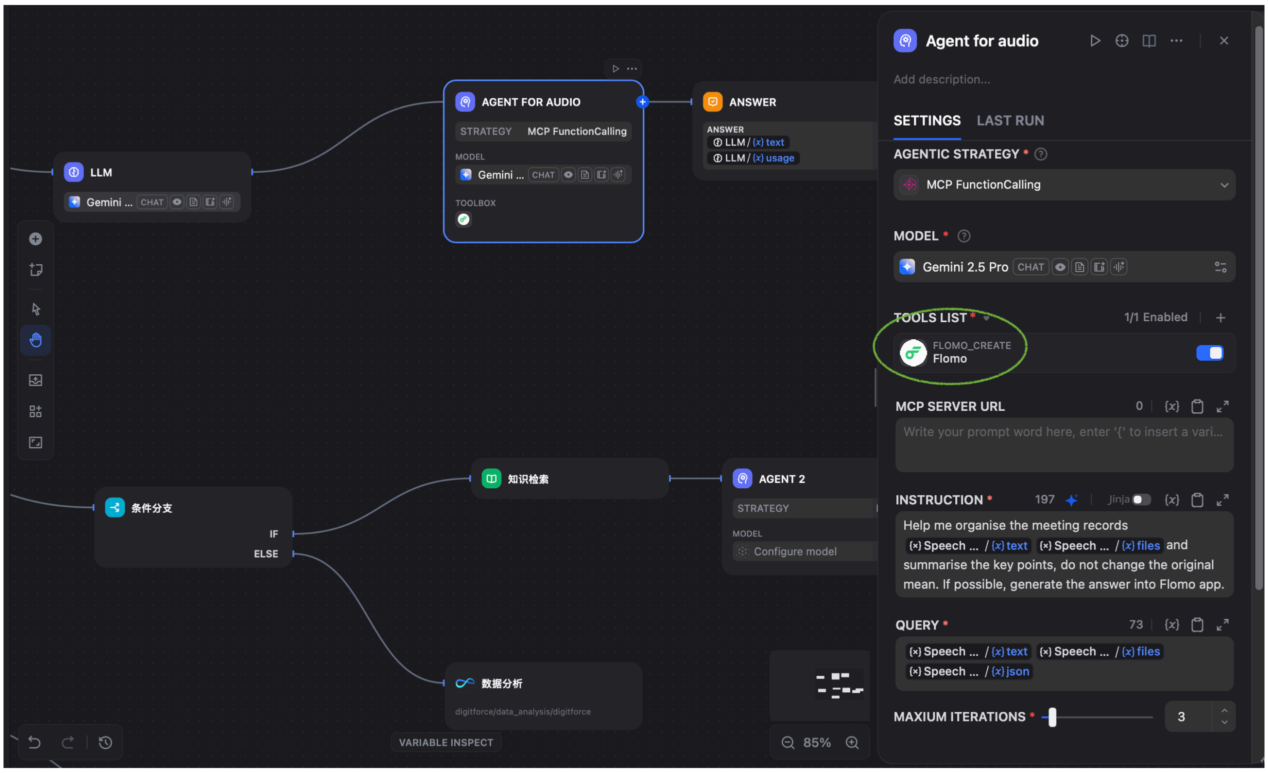
2.3 Tuning and Iteration

As the Workflow's structure solidifies, it can be further customised to integrate with personal work habits and external tools.

Examples:

1. As a long-time user of the note-taking app **Flomo**, I integrated it into my workflow. Summarised meeting minutes can now be automatically uploaded to Flomo by simply adding it as a tool within the Agent. This allows for easy access to notes directly on my

phone.



Agent for audio

Add description...

SETTINGS LAST RUN

AGENTIC STRATEGY *
MCP FunctionCalling

MODEL *
Gemini 2.5 Pro CHAT

TOOLS LIST * 1/1 Enabled
FLOMO_CREATE
Flomo

MCP SERVER URL 0
Write your prompt word here, enter '{' to insert a vari...

INSTRUCTION * 197 Jinja
Help me organise the meeting records {x} Speech ... / {x} text {x} Speech ... / {x} files and summarise the key points, do not change the original mean. If possible, generate the answer into Flomo app.

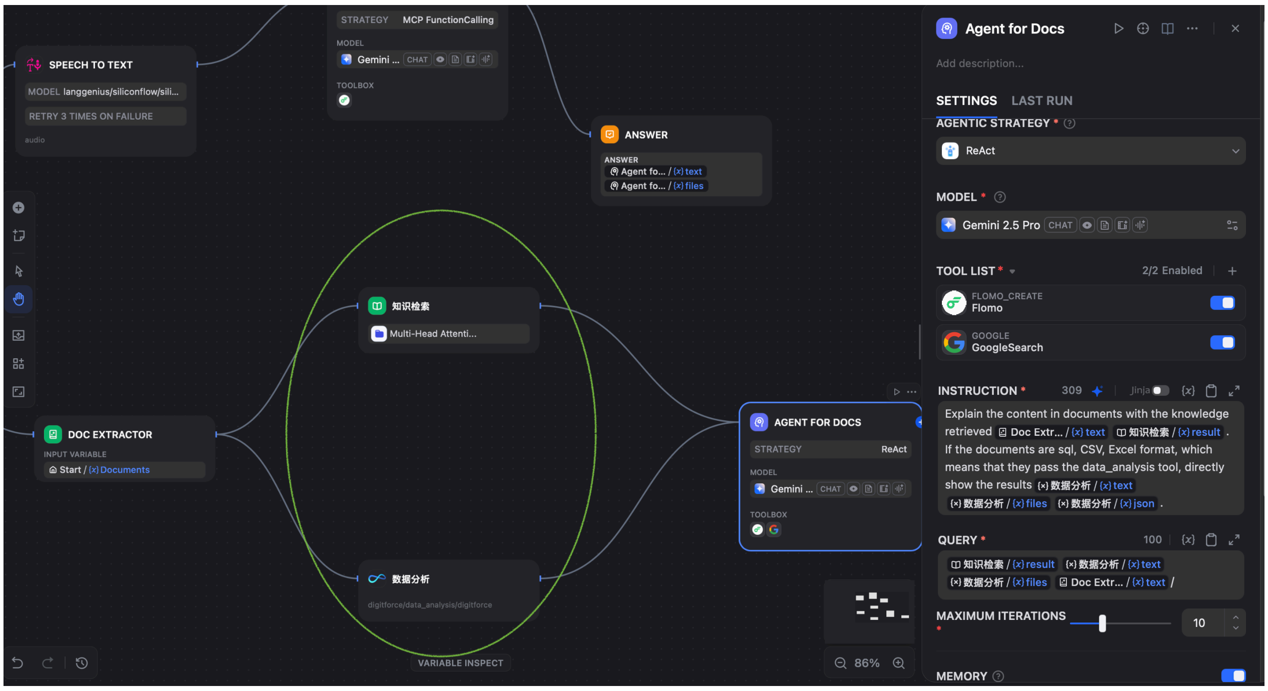
QUERY * 73
{x} Speech ... / {x} text {x} Speech ... / {x} files {x} Speech ... / {x} json

MAXIUM ITERATIONS * 3

2. For users requiring higher precision, domain-specific expertise, or enhanced privacy for document analysis, creating a custom Knowledge Base is a powerful option. Compared to direct analysis by an LLM via a prompt, applying Dify's built-in **RAG** (Retrieval-Augmented Generation) capabilities significantly improves the accuracy and specialization of text and data analysis. The Knowledge Base creation process also offers a chance to explore advanced features like Dify's new parent-child chunking and various retrieval strategies
3. **Advanced Document Handling with Parallel Processing.** To enhance the Workflow's versatility with diverse document types, I implemented a parallel processing structure within the **document processing** branch, as shown in the image. This design features two nodes working in tandem: Knowledge Retrieval and Data Analysis.

   The purpose of this parallel setup is to intelligently route different file formats to the most appropriate tool for initial processing. The **Data_analysis** tool is specifically designed for structured data files, such as CSV or Excel, and outputs a clean JSON format for the agent to use. Conversely, the **Knowledge Retrieval** node excels with unstructured text from documents like PDFs or Word files, which the **Data_analysis** tool cannot handle.

   By connecting them in parallel, the workflow automatically diverts different formats for specialized processing before both paths converge at the final **AGENT FOR DOCS**. This agent then performs the conclusive text processing on the unified, pre-analyzed information, creating a highly flexible and efficient system for handling any type of document.



Agent for Docs

Add description...

SETTINGS LAST RUN

AGENTIC STRATEGY * ReAct

MODEL * Gemini 2.5 Pro CHAT

TOOL LIST * 2/2 Enabled
FLOMO_CREATE Flomo
GOOGLE GoogleSearch

INSTRUCTION * 309

Explain the content in documents with the knowledge retrieved Doc Extr... / {x} text 知识检索 / {x} result . If the documents are sql, CSV, Excel format, which means that they pass the data_analysis tool, directly show the results {x} 数据分析 / {x} text {x} 数据分析 / {x} files {x} 数据分析 / {x} json .

QUERY * 100

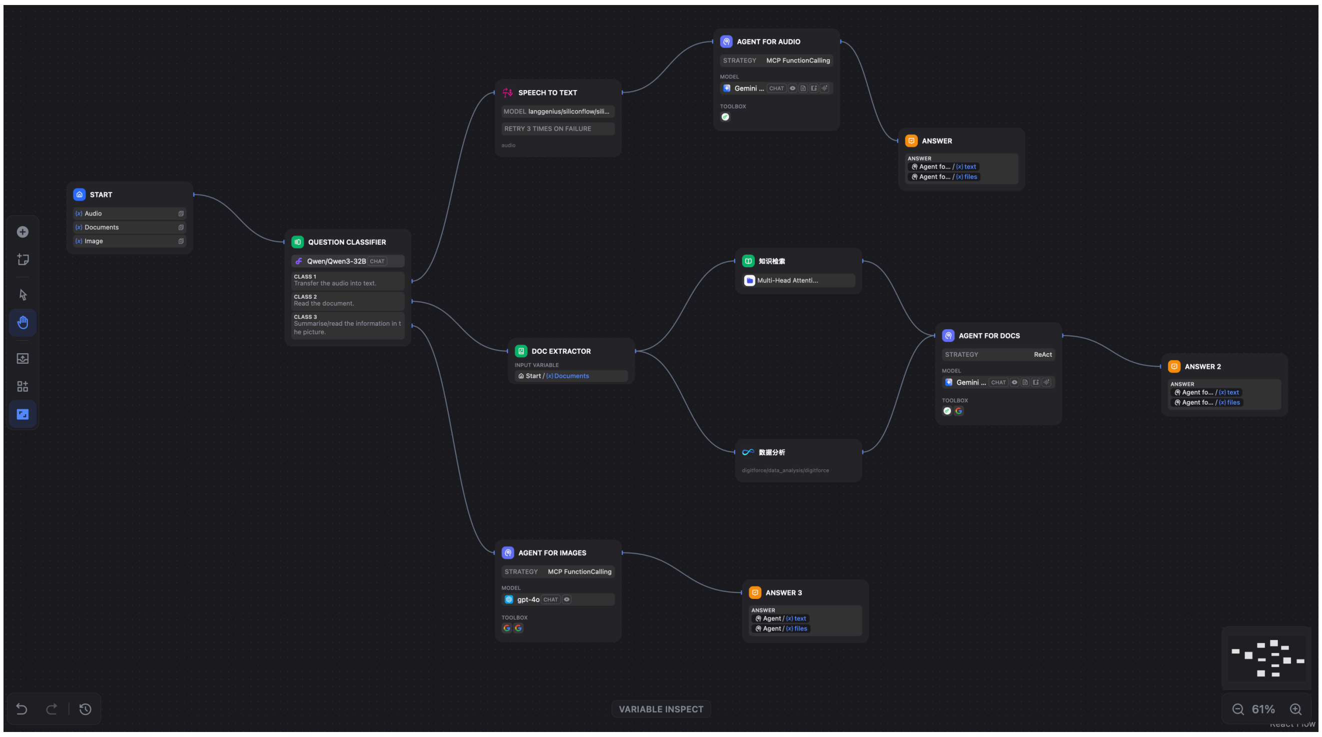
知识检索 / {x} result {x} 数据分析 / {x} text {x} 数据分析 / {x} files Doc Extr... / {x} text /

MAXIMUM ITERATIONS * 10

MEMORY

Part 3: Core Use Cases: The Workflow in Action

A theoretical design is only as good as its practical application. Here is how this **Triple-branch** Workflow performs in three common, real-world business scenarios, transforming tedious tasks into automated, efficient processes.



Scenario 1: From Dense Documents to Core Insights — Report & Content Analysis

In any role that requires data-driven decisions, we are often inundated with lengthy reports, market analyses, or articles. Instead of spending hours manually sifting through them, I can now simply upload a PDF or CSV document, or even paste a wall of text, into the Workflow.

The "Document Processing" branch is automatically triggered. The **ReAct** Agent extracts the full text and, following the predefined instructions (Prompt), performs a series of actions: it generates a concise summary, extracts key data points like KPIs and growth rates, performs calculations for metrics like **ROI** or **GMV**, and identifies the core arguments or conclusions.

Scenario 2: From Long Recordings to Actionable Minutes — Meeting Summarisation

Post-meeting work can be a significant bottleneck, especially when it involves transcribing and summarising long discussions. This Workflow elegantly solves that problem and here is the process:

A user upload a meeting's audio file (e.g., MP3, WAV) to the system. The "Audio Processing" branch activates, where the **Speech-to-Text** node first creates an accurate transcript. This text is then passed to an LLM agent, which has been instructed to refine the raw output. It automatically cleans up colloquialisms and filler words, distinguishes between different speakers (where possible), distills the conversation into key topics and decisions, and—most importantly—compiles a clear list of **Action Items**. Finally, applying the tool - **Flomo**, the output will automatically upload to user's **Flomo** app on their phones.

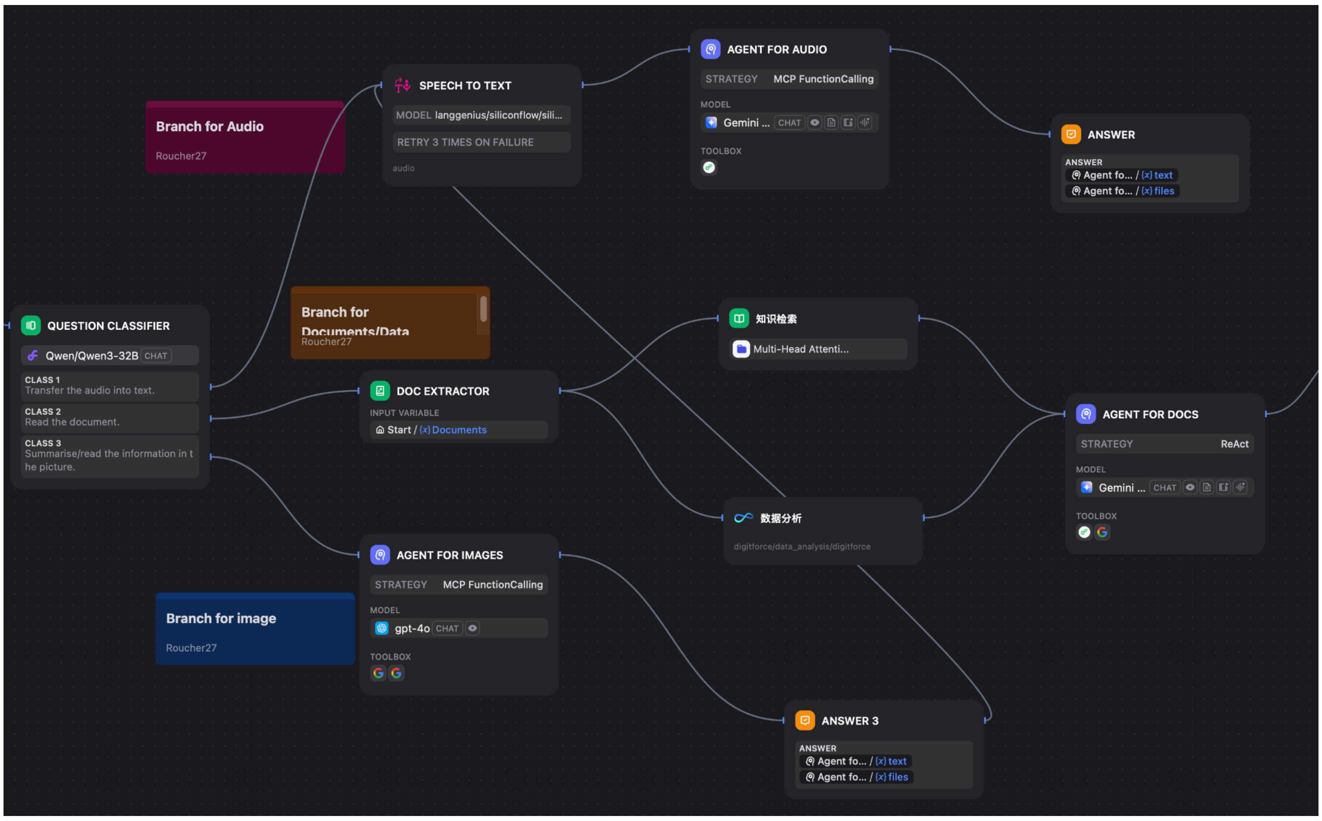
Scenario 3: From Visual Data to Editable Text — Image Content Organization

We often capture valuable information in images: a photo of a whiteboard after a brainstorming session, a screenshot of a presentation slide, or an event poster. This workflow seamlessly converts that static visual data into a dynamic, usable format.

By simply uploading an image, the "Image Processing" branch is activated. The Image Vision node recognizes and extracts all the text and understands its basic layout. This raw text is then sent to an LLM agent that logically organizes the content, applying proper formatting and structure.

Highlight: a Closed-Loop System
What makes this workflow particularly powerful is its recursive, **closed-loop** capability. The document generated from an image in **Scenario 3** doesn't have to be the final output.
This is achieved by creating a new connection from the output of the image processing branch (in this case, **Answer 3**) to the **"Document Processing"** branch for a much deeper analysis—such as summarisation, translation, or data extraction.This creates a powerful, end-to-end information processing cycle, where data seamlessly transforms from one format

to another and is refined at every step.

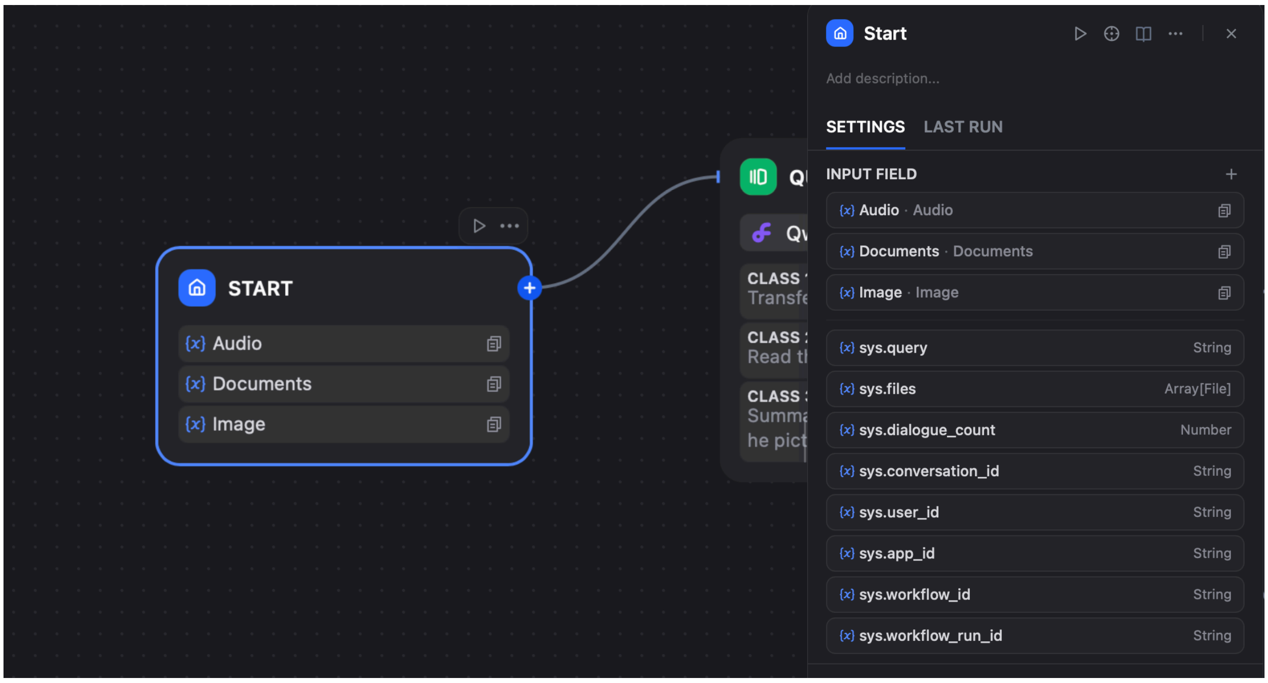


Part 4: From User's Perspective: Suggestions for Improvement

After extensive use of the Dify platform, I identified a few minor areas where enhancements could significantly improve the user experience, particularly regarding Workflow construction and navigation.

1. **Reordering Inputs in the START Node:** Currently, the inputs within the **START** node cannot be reordered using drag-and-drop after they are created. To change their sequence, one must delete the existing fields and recreate them in the desired order. Implementing a simple drag-and-drop functionality here would make layout management

more intuitive and efficient for users.



Start

Add description...

SETTINGS LAST RUN

INPUT FIELD

{x} Audio · Audio	
{x} Documents · Documents	
{x} Image · Image	
{x} sys.query	String
{x} sys.files	Array[File]
{x} sys.dialogue_count	Number
{x} sys.conversation_id	String
{x} sys.user_id	String
{x} sys.app_id	String
{x} sys.workflow_id	String
{x} sys.workflow_run_id	String

2. **Distinct Icons for the App and Documentation:** For new users especially, having the official documentation open for quick reference is a common practice. However, the documentation pages and the main app currently share the same tab icon. When multiple tabs are open, it becomes difficult to distinguish between them. So, introducing another distinct icon for the documentation would greatly enhance navigational clarity and create a smoother user experience.

Conclusion

This concludes my overview of the design, construction, and practical application of my custom **Dify** Workflow. This project was developed from the ground up, without relying on pre-existing templates. It was born from a direct analysis of real-world work scenarios, identifying key pain points, and engineering targeted solutions by integrating the appropriate tools and plugins.

For those interested in diving deeper into building powerful workflows and agents with Dify, I highly recommend reading more templates and exploring the official resources, including the Dify 101 Docs, the developer's repository or the project's GitHub Repository.